TRELLIS DECODING AND APPLICATIONS FOR QUANTUM ERROR CORRECTION

A Dissertation
Presented to
The Academic Faculty

By

Eric Sabo

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Mathematics
College of Sciences

Georgia Institute of Technology

December 2022

© Eric Sabo 2022

TRELLIS DECODING AND APPLICATIONS FOR QUANTUM ERROR CORRECTION

Thesis committee:

Dr. Evans Harrell
School of Mathematics
Georgia Institute of Technology

Dr. Moinuddin Qureshi
School of Computer Science
Georgia Institute of Technology

Dr. Matthew Baker
School of Mathematics
Georgia Institute of Technology

Dr. Kenneth Brown
Pratt School of Engineering
Duke Universityy

Dr. Martin Short
School of Mathematics
Georgia Institute of Technology

Date approved: August 02, 2022

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ABDF almost block-diagonal form

BCH Bose-Chaudhuri-Hocquenghem

BCJR Bahl-Cocke-Jelinek-Raviv

BFS breadth-first search

BZ Brouwer-Zimmermann

CP completely positive

CPTP completely-positive, trace-preserving

CSS Calderbank-Shor-Steane

CWE complete weight enumerator

DFS depth-first search

DQC distributed quantum computing

ET elementary trellis

GNS Gelfand-Naimark-Segal

i.i.d. independently and identically distributed

i.ni.d. independently and non-identically distributed

ML maximum likelihood

MWPM minimum-weight perfect matching

NISQ near-term intermediate scale quantum

NP nondeterministic polynomial time

OQEC operator quantum error correction

QEC quantum error correction

QECC quantum error-correcting code

QRM 15-qubit quantum Reed-Muller

RM Reed-Muller

RS Reed-Solomon

TOF trellis-oriented form

TP trellis product

VBD Vardy-Be'ery decomposition

SUMMARY

Compact, graphical representations of error-correcting codes called trellises are a crucial tool in classical coding theory, establishing both theoretical properties and performance metrics for practical use. The idea was extended to quantum error-correcting codes by Ollivier and Tillich in 2005. Here, we use their foundation to establish a practical decoder able to compute the maximum-likely error for any stabilizer code over a finite field of prime dimension. We define a canonical form for the stabilizer group and use it to classify the internal structure of the graph. Similarities and differences between the classical and quantum theories are discussed throughout. Numerical results are presented which match or outperform current state-of-the-art decoding techniques. New construction techniques for large trellises are developed and practical implementations discussed. We then define a dual trellis and use algebraic graph theory to solve the maximum-likely coset problem for any stabilizer code over a finite field of prime dimension at minimum added cost.

Classical trellis theory makes occasional theoretical use of a graph product called the trellis product. We establish the relationship between the trellis product and the standard graph products and use it to provide a closed form expression for the resulting graph, allowing it to be used in practice. We explore its properties and classify all idempotents. The special structure of the trellis allows us to present a factorization procedure for the product, which is much simpler than that of the standard products.

Finally, we turn to an algorithmic study of the trellis and explore what coding-theoretic information can be extracted assuming no other information about the code is available. In the process, we present a state-of-the-art algorithm for computing the minimum distance for any stabilizer code over a finite field of prime dimension. We also define a new weight enumerator for stabilizer codes over $\mathbb{F}_2$ incorporating the phases of each stabilizer and provide a trellis-based algorithm to compute it.

CHAPTER 1
INTRODUCTION AND BACKGROUND

To maintain a coherent presentation, this document does not include the author's work on computational knot theory [1] or on a new branch of classical coding theory called DNA coding theory. A software package covering topics in classical and quantum coding theory beyond what is discussed here was developed in conjunction with this work and has been publicly released as the beginning of a coding theory library for the Julia programming language [2]. This includes code for the trellis-based algorithms developed in this work.

Quantum error correction (QEC) is fundamentally an analysis-based field at heart, and there are numerous references on quantum computation for mathematicians which do an excellent job explaining the operator theory behind the subject. However, most physicists working in the field take a group-theoretic approach. We too will stick to algebra in this work, and here we discuss both aspects of the subject to reach the largest audience. After going through the basics of the operator theory, we switch to the group approach while attempting to motivate and explain how each choice in the modeling stems back to its analytical core. Undergraduate knowledge of algebra is assumed but all concepts from operator theory will be explained from scratch. No prior background of QEC is assumed. The standard reference for this is [3]; however, this book is now dated and does not contain the material discussed here. A more advanced and recent treatment may be found in [4].

Related to QEC is "classical" coding theory and "classical" error-correcting codes. Standard references for this material are [5, 6, 7]. The notation and terminology used here is aligned with [7]. Knowledge of the classical theory can both be beneficial and harmful to the study of quantum error-correcting code (QECC)s. We compare and contrast the clas-

sical and quantum theories throughout this work for those readers which may be familiar with the classical case. We will return to the majority of the classical concepts introduced in Section 1.1 in Chapter 4.

Section 1.2 begins the review of operator theory and Section 1.3, QEC. Section 1.3.2 introduces the so-called stabilizer (group) formalism. Several QECCs we will refer to in the following chapters are introduced in 1.3.3 and 1.3.4. The quantum circuit model is discussed in Section 1.4. The key concepts of what defines a good QECC begin in Section 1.4.2. Finally, Section 1.5 includes comments on the design, goal and implementation of computer simulations of classical and quantum error-correcting codes.

1.1 Classical Error-Correcting Codes

For simplicity when we connect to quantum coding theory, we restrict to linear codes over $\mathbb{F}_q$ for some prime power $q = p^m$. A (classical) error correcting code $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_q^n$. Elements of $\mathcal{C}$ are called codewords. The number of codewords in $\mathcal{C}$ is denoted $|\mathcal{C}|$. The dimension of $\mathcal{C}$, $\dim(\mathcal{C})$, is defined to be the dimension of $\mathcal{C}$ as a vector space over $\mathbb{F}_q$, i.e., $|\mathcal{C}| = q^{\dim(\mathcal{C})}$. It is customary to denote $\dim(\mathcal{C})$ by k such that $\mathcal{C}$ is an $(n, q^k)_q$ code, or an $[n, k]_q$ code. The notation of choice depends on whether or not it is easier to make an argument about $|\mathcal{C}|$ or $\dim(\mathcal{C})$. The latter is the notation of choice here and is almost exclusively used in quantum coding theory. An $[n, k]_q$ code is written $[n, k]$ when $q = 2$.

A $k \times n$ matrix G is a generator matrix for $\mathcal{C}$ if $\mathcal{C}$ is the row space of G. An $(n-k) \times n$ parity check matrix H for $\mathcal{C}$ is a generator matrix for the row space of the vector space orthogonal to $\mathcal{C}$ in $\mathbb{F}_q^n$ with respect to the standard Euclidean inner product, $\mathcal{C}^\perp$, i.e., $\mathcal{C} = \ker H$. This is called the dual code of $\mathcal{C}$ and the generator and parity-check matrices of $\mathcal{C}$ and $\mathcal{C}^\perp$ are switched. A code is called self-orthogonal if $\mathcal{C} \subseteq \mathcal{C}^\perp$ and self-dual if $\mathcal{C} = \mathcal{C}^\perp$. The orthogonality of $\mathcal{C}$ and $\mathcal{C}^\perp$ gives $G^T H = G H^T = 0$. A generator matrix is said to be in standard form if $G = (I_k \mid A)$, where I_k is the $k \times k$ identity matrix, and a parity-check

matrix is said to be in standard form if $H = (B \mid I_{n-k})$. The relationship between G and H gives $B = -A^T$. By elementary row operations, any linear code is equivalent to a linear code with a generator matrix in standard form. The product Hv is called the syndrome of v and a zero syndrome implies $v \in \mathcal{C}$. Given G or H one may always generate the other in $\mathcal{O}(n^3)$ by simply computing the (right) kernel of the matrix using elementary linear algebra. A subset of $\{1, 2, \ldots, n\}$ of cardinality k is called an information set for $\mathcal{C}$ if the corresponding columns of G are linearly independent.

The (Hamming) weight of $x \in \mathbb{F}_q^n$, $\mathrm{wt}(x)$, is the number of nonzero components in the vector. The (Hamming) distance between $x \in \mathbb{F}_q^n$ and $y \in \mathbb{F}_q^n$, denoted by $d(x, y)$, is defined to be the number of places at which x and y differ, i.e., $d(x, y) = \mathrm{wt}(x - y)$. It is easy to see that this satisfies the properties of a metric. For a code $\mathcal{C}$ with $|\mathcal{C}| \geq 2$, the (minimum) distance of $\mathcal{C}$, denoted by $d = d(\mathcal{C})$, is

$$d(\mathcal{C}) = \min\{d(x, y) \mid x, y \in \mathcal{C}, x \neq y\} = \min\{\mathrm{wt}(c) \mid c \in \mathcal{C}\},$$

where the second equality holds only for the linear codes considered in this work. An $[n, k]_q$ code with minimum weight d is denoted by $[n, k, d]_q$[1]. The weight enumerator of $\mathcal{C}$ is the bivariate polynomial

$$W(\mathcal{C}; x, y) = \sum_{i=0}^{n} A_i x^i y^{n-i}, \tag{1.1}$$

where A_i is the number of elements of $\mathcal{C}$ with weight i. The weight distribution of $\mathcal{C}$ is the ordered sequence $\{A_i\}_{i=0}^n$. The minimum distance is hence the smallest index i such that $A_i \neq 0$. The weight enumerator of $\mathcal{C}$ and $\mathcal{C}^\perp$ are related via the MacWilliams identity

$$W(\mathcal{C}^\perp; x, y) = \frac{1}{|C|} W(\mathcal{C}; y - x, y + x). \tag{1.2}$$

[1]Some authors, especially in the early quantum literature, use the notation $[n, d, k]_q$. It is usually clear after a little thought which notation is being employed if not explicitly specified.

Taking this as a fundamental property of coding theory, the most general ring over which such a formula holds, and hence we can do coding theory, is a Frobenius ring [8].

Information is encoded in $\mathcal{C}$ via $\mathrm{enc} : \mathbb{F}_q^k \to \mathbb{F}_q^n$, $v \mapsto vG$. One may think of the k bits of data in the information set as being embedded, or "smeared out", into the larger, n-dimensional space. The parameter d is related to the error-correcting process called decoding, which should not be confused with "unencoding".

Theorem 1.1.1

An $[n, k, d]_p$ code can correct $t = \lfloor (d-1)/2 \rfloor$ or fewer errors. Conversely, a code which can correct $t = \lfloor (d-1)/2 \rfloor$ or fewer errors has minimum weight d.

The variables n, k, d, and t will be reserved for these quantities throughout this work and will be referred to often.

The proof of this result is elementary and may be found in any book on coding theory. Intuitively, picture each codeword in C as a point in V. If $t = (d-1)/2$, then spheres of radius t centered about each codeword are guaranteed to be disjoint. Now suppose a codeword is transported from one place to another, or in the terminology of information theory, a codeword is sent through a channel. The means of transportation (or the details of the channel) is irrelevant at this point and may be considered to be a black box. We must determine whether the vector at the end of the process is the same codeword or if any errors have occurred. If the vector has t or fewer errors, then it will be contained in the sphere of radius t about the original codeword. One then corrects the errors, or decodes, by projecting the vector back to codeword at the center of the sphere. If the vector has greater than t errors, then, in the best case, it will not be contained by any spheres. This is detectable as $Hc^T \neq 0$, and will signal an error has occurred even if it is unable to be corrected. In the worst case, the vector will be contained by a sphere about another codeword, causing our decoding scheme to fail.

The construction of classical error-correcting codes becomes a game of balancing the parameters n and d given k. If n is chosen too large with d small, the probability of a

vector not being contained in a sphere becomes large. On the other hand, if n is large, one may attempt to find a k-dimensional subspace which only contains vectors of large weight. However, the larger d becomes for a fixed n, the higher the probability that spheres will start to intersect. This may be tolerated by a decoding scheme as long as the probability that a vector is contained in the intersection is low. At the same time, listing every codeword and keeping track of every sphere becomes a daunting task for even for reasonably small n. All things considered, the sphere model of decoding is too flawed to be used in practice. Decades of research in the field have replaced it with increasingly more sophisticated techniques and models; however, all of the basic concepts are captured by this simple model, and the intuition may be used going forward.

The direct sum of codes $\mathcal{C}_1$ and $\mathcal{C}_2$ has generator matrix $G_1 \oplus G_2$. A code is called decomposable if can be written as the direct sum of smaller codes. The most common form of extending a code is to add an extra column to the generator matrix such that the sum of the coordinates of each row is 0. Augmenting a code adjoins rows to the generator matrix. Expurgating a code deletes rows from the generator matrix and then removes any potentially resulting zero columns. Puncturing a code deletes columns from the generator matrix and then removes any potentially resulting zero rows. Shortening is expurgating followed by puncturing. We will only puncture a single column in this work, so we drop the standard notational dependence on column indices and denote a puncture of $\mathcal{C}$ by $\mathcal{C}^*$, which is common. Likewise, we will use the common notation $\overline{\mathcal{C}}$ to denote a shortened code. The most systematic way to shorten $\mathcal{C}$ is to use the theorem that $\overline{C}^{\perp} = (\mathcal{C}^{\perp})^*$ and then use elementary linear algebra to compute the duals.

Let $\mathcal{C}$ be an $[n, k, d]_{p^m}$ code. Then the subfield subcode of $\mathcal{C}$ over a subfield $\mathbb{F} \leq \mathbb{F}_{p^m}$, denoted $\mathcal{C}|_{\mathbb{F}}$, is given by $\mathcal{C} \cap \mathbb{F}^n$, i.e., the collection of codewords of $\mathcal{C}$ whose components lie entirely in $\mathbb{F}$. The code $\mathcal{C}$ is called the supercode of $\mathcal{C}|_{\mathbb{F}}$. If $\mathcal{C}$ has parameters $[n, k, d]_{p^m}$, $\mathcal{C}|_{\mathbb{F}}$ has parameters $[n, k', \geq d]$ over $\mathbb{F}$, where $n - k \leq n - k' \leq \ell(n - k)$ and $\ell = [\mathbb{F}_{p^m} : \mathbb{F}]$ (the index of $\mathbb{F}$ in $\mathbb{F}_{p^m}$). As the codewords of $\mathcal{C}|_{\mathbb{F}}$ are codewords of $\mathcal{C}$, it follows immediately that

the minimum distance of $\mathcal{C}|_{\mathbb{F}}$ is at least the minimum distance of $\mathcal{C}$, and $\mathcal{C}|_{\mathbb{F}}$ can be decoded using the same algorithm as $\mathcal{C}$, although perhaps not efficiently as a native algorithm over $\mathbb{F}$ designed specifically for the subfield subcode.

An $[mn, mk, \geq d]_p$ code may be constructed from an $[n, k, d]_{p^m}$ code by expanding its elements using a basis of $\mathbb{F}_{p^m}/\mathbb{F}_p$. The first code is called the expanded code of the second. To see why the minimum distance of the code could increase, let $\beta = \{\beta_j\}_1^m$ be a basis of $\mathbb{F}_{p^m}/\mathbb{F}_p$ and let $c = (c_1, \ldots, c_n) \in \mathbb{F}_{p^m}^n$ be a minimum weight codeword in an $[n, k, d]_{p^m}$ code. Expressing each c_i with respect to β, $c_i = \sum_j c_{ij}\beta_j$, we can replace each element with its corresponding m-tuple, $(c_{i1}, \ldots, c_{im})$. If $c_i \neq 0$, then the Hamming weight of its expansion is at least one and therefore the Hamming weight of the expansion of c is at least d.

Recall that the inner product over finite fields is given by the trace. In particular, if β is a basis of $\mathbb{F}_{p^m}/\mathbb{F}_p$ such that $x = \sum_{j=1}^m x_j\beta_j$ for $x \in \mathbb{F}_{p^m}$, then $x_j = \mathrm{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_p}(x\beta_j^\perp) \in \mathbb{F}_p$, where $\beta^\perp$ is the unique trace-orthogonal dual of β such that $\mathrm{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_p}(x_i y_j) = \delta_{ij}$ for $x_i \in \beta$ and $y_j \in \beta^\perp$. (The dual basis always exists and is easy to compute given β.) For $c = (c_1, \ldots, c_n) \in \mathbb{F}_{p^m}^n$ denote the expansion with respect to β by the isomorphism $\phi_\beta : \mathbb{F}_{p^m}^n \to \mathbb{F}_p^{nm}$ given by

$$\begin{aligned}\phi_\beta(c) &= (\phi_\beta(c_1), \ldots, \phi_\beta(c_n)) \\ &= (\mathrm{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_p}(c_1\beta_1^\perp), \ldots, \mathrm{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_p}(c_1\beta_m^\perp), \mathrm{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_p}(c_2\beta_1^\perp), \ldots, \mathrm{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_p}(c_n\beta_m^\perp)).\end{aligned}$$

Generator and parity check matrices for expanded codes are given by (e.g. [9])

$$G_\phi = \begin{pmatrix} \phi_\beta(\beta_1 g_1) \\ \vdots \\ \phi_\beta(\beta_m g_1) \\ \phi_\beta(\beta_1 g_2) \\ \vdots \\ \phi_\beta(\beta_m g_k) \end{pmatrix} \quad , \quad H_\phi = \begin{pmatrix} \phi_\beta(\beta_1 h_1) \\ \vdots \\ \phi_\beta(\beta_m h_1) \\ \phi_\beta(\beta_1 h_2) \\ \vdots \\ \phi_\beta(\beta_m h_{n-k}) \end{pmatrix}. \tag{1.3}$$

In general, an expanded code loses the properties of its parent code and different bases could produce different expanded codes with different parameters and properties. It is still not yet known how to choose a basis to *a priori* maximize the minimum distance of the expanded code. One crucial property that might not be maintained by a basis expansion is orthogonality. To see this, let β be an arbitrary basis for $\mathbb{F}_{p^m}/\mathbb{F}_p$. If $\mathcal{C}_2 \subseteq \mathcal{C}_1$ over $\mathbb{F}_{p^m}$, then $\phi_\beta(\mathcal{C}_2) \subseteq \phi_\beta(\mathcal{C}_1)$ over $\mathbb{F}_p$ trivially, since if $x \in \mathcal{C}_2$ then $x \in \mathcal{C}_1$ and $\phi_\beta(x) \in \phi_\beta(\mathcal{C}_2)$ and $\phi_\beta(x) \in \phi_\beta(\mathcal{C}_1)$. It is well-known in classical coding theory, and can be verified by direct computation, that $(\phi_\beta(\mathcal{C}))^\perp = \phi_{\beta^\perp}(\mathcal{C}^\perp)$. Now suppose $\mathcal{C} \subseteq \mathcal{C}^\perp$. Then $\phi_\beta(\mathcal{C}) \subseteq \phi_\beta(\mathcal{C}^\perp)$ and $\phi_\beta(\mathcal{C})$ is self-orthogonal if and only if $\phi_\beta(\mathcal{C}^\perp) \subseteq (\phi_\beta(\mathcal{C}))^\perp = \phi_{\beta^\perp}(\mathcal{C}^\perp)$. It is sufficient for $\beta = \beta^\perp$ but not every field extension has a self-dual basis. Even if a self-dual basis for the extension exists, it is often difficult to find. The two most common bases are the polynomial bases of the form $\{1, \alpha, \ldots, \alpha^{m-1}\}$ and the normal bases of the form $\{\alpha, \alpha^p, \alpha^{p^2}, \ldots, \alpha^{p^{m-1}}\}$. If α is primitive, then the polynomial basis is called a primitive (polynomial) basis. Polynomial bases cannot be self-dual due to the presence of the 1. Normal bases are easy to find and it's trivial to check whether or not a given basis is self-dual.

Theorem 1.1.2 ([10])

The extension $\mathbb{F}_{q^m}$ has a self-dual basis over $\mathbb{F}_q$ if and only if either q is even or both q and m are odd.

Theorem 1.1.3 ([10])

The number, $sd(m,q)$, of distinct self-dual bases of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ is

$$sd(m,q) = \frac{c}{m!}\prod_{i=1}^{m-1}(q^i - a_i), \tag{1.4}$$

where

$$c = \begin{cases} 0 & \textit{if } q \textit{ is odd and } m \textit{ is even} \\ 1 & \textit{if } q \textit{ is even} \\ 2 & \textit{if } q \textit{ and } m \textit{ are odd} \end{cases}$$

and

$$a_i = \begin{cases} 0 & \textit{if } i \textit{ is odd} \\ 1 & \textit{if } i \textit{ is even.} \end{cases}$$

Theorem 1.1.4 ([10])

The extension $\mathbb{F}_{q^m}$ has a self-dual normal basis over $\mathbb{F}_q$ if and only if both q and m are odd or q is even and $4 \nmid m$.

It is also possible to enumerate the self-dual normal bases. Assume $\gcd(q,n) = 1$ and let $x^n - 1 = (x-1)\prod_{i=1}^{r} f_i(x)$ be the decomposition of $x^n - 1$ into r distinct irreducible factors $f_i(x)$ over $\mathbb{F}_q$. If $f_i(x)$ is an irreducible factor, then $f_i^*(x)$ is also an irreducible factor and the degree of $f_i(x)$ is necessarily even, $\deg f_i(x) = 2c_i$. Suppose t of these are self-reciprocal, $f_i(x) = f_i^*(x)$. Grouping the remaining factors into pairs $g_j(x) = f_j(x)f_j^*(x)$ of degree $2d_j$, we can rewrite the above factorization as $x^n - 1 = (x-1)\prod_{i=1}^{t} f_i(x)\prod_{j=t+1}^{u} g_j(x)$, where $u = (r+t)/2$.

Theorem 1.1.5 ([10])

The number, $sdn(m,p)$, of distinct self-dual normal bases of $\mathbb{F}_{p^m}$ over $\mathbb{F}_p$ is

$$sdn(m,p) = \begin{cases} \dfrac{2^a}{m} \displaystyle\prod_{i=1}^{t} (p^{c_i}+1) \prod_{j=t+1}^{u} (p^{d_j}-1) & \text{if } gcd(m,p) = 1 \\ \dfrac{1}{p} p^{(p-1)(s+b)/2} sdn(s,p) & \text{if } m = sp, \end{cases} \tag{1.5}$$

where

$$a = \begin{cases} 0 & \text{if } p \text{ is even and } 4 \nmid m \\ 1 & \text{if both } p \text{ and } m \text{ are odd,} \end{cases}$$

and

$$b = \begin{cases} 0 & \text{if both } p \text{ and } m \text{ are odd} \\ 1 & \text{if } p \text{ is even and } s \text{ is odd.} \end{cases}$$

In the cases where self-dual normal bases do not exist, self-dual bases will have to be found using another method. For the cases we are generally interested in, the remaining self-dual bases that are not normal are relatively easy to manually compute. This information is compiled in Table 1.1 for convenience. Recall that there are $(1/m!)\prod_{i=0}^{m-1}(q^m - q^i)$ (unordered) bases of $\mathbb{F}_{q^m}/\mathbb{F}_q$. Of these, almost all of these bases can be generated by scalar multiples and powers of a select few. For example, there are only 16 “distinct" bases for $\mathbb{F}_{2^4}/\mathbb{F}_2$ [11].

A code is cyclic if for every $(f_0, \ldots, f_{n-1}) \in \mathcal{C}$ the vector $(f_{n-1}, f_0, \ldots, f_{n-2})$ is also in C. Consider the polynomial $f = f_0 + f_1 x + \ldots + f_{n-1}x^{n-1}$. Multiplying by x and setting $x^n = 1$ gives $f_{n-1} + f_0 x + \ldots + f_{n-2}x^{n-1}$. Thus, elements of cyclic codes are naturally viewed as coefficient vectors of polynomials in $\mathbb{F}_p[x]/(x^n - 1)$. Cyclic codes are in bijection with ideals of this ring and hence with divisors of $x^n - 1$.

Let $x^n - 1 = g(x)h(x)$ for some $g(x), h(x) \in \mathbb{F}_p[x]$. Then $\mathcal{C} = (g(x))$ is a cyclic code with generator polynomial $g(x)$ viewed as an ideal of $\mathbb{F}_p[x]/(x^n - 1)$. Let $c(x) = a(x)g(x) \in \mathcal{C}$; then $c(x)h(x) = a(x)g(x)h(x) = a(x)(x^n - 1) \equiv 0$. In analogy with H,

Table 1.1: The number of bases of various types for each extension field. The normal bases* follow from direct computation, and the formulas do not apply to **.

Extension Field	# Bases	# Normal Bases*	# Self-Dual Bases	# Self-Dual Normal Bases
$\mathbb{F}_{2^2}/\mathbb{F}_2$	3	1	1	1
$\mathbb{F}_{2^3}/\mathbb{F}_2$	28	1	1	1
$\mathbb{F}_{2^4}/\mathbb{F}_2$	840	2	2	0
$\mathbb{F}_{2^5}/\mathbb{F}_2$	83328	3	6	1
$\mathbb{F}_{3^2}/\mathbb{F}_3$	24	2	0	0
$\mathbb{F}_{3^3}/\mathbb{F}_3$	1872	6	8	2
$\mathbb{F}_{4^2}/\mathbb{F}_4$	90	6	2	2**
$\mathbb{F}_{5^2}/\mathbb{F}_5$	240	8	0	0

$h(x)$ is called the parity check polynomial. The code $\mathcal{C}$ has parameters $[n, n-\deg g]$. Recall that the reciprocal (reverse) of a polynomial $f(x)$ of degree n is $f^r(x) = x^n f(x^{-1})$. An easy argument shows that $\mathcal{C}^\perp = (h^r(x)/h(0))$, where we have introduced a normalization factor to keep the generator polynomial monic. Note that if $h(x) \mid x^n - 1$, then $h^r(x)/h(0) \mid x^n - 1$, so the dual code of a cyclic code is cyclic. Let $\deg g = n - k$. Then the generator and parity check matrices for $\mathcal{C}$ are given by

$$G = \begin{pmatrix} g_0 & g_1 & \cdots & \cdots & \cdots & \cdots & g_{n-k} & & \\ & g_0 & g_1 & \cdots & \cdots & \cdots & \cdots & g_{n-k} & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & g_0 & g_1 & \cdots & \cdots & \cdots & \cdots & g_{n-k} \end{pmatrix}, \tag{1.6}$$

and

$$H = \begin{pmatrix} h_k & h_{k-1} & \cdots & \cdots & \cdots & \cdots & h_0 & & \\ & h_k & h_{k-1} & \cdots & \cdots & \cdots & \cdots & h_0 & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & h_k & h_{k-1} & \cdots & \cdots & \cdots & \cdots & h_0 \end{pmatrix}, \tag{1.7}$$

respectively.

The general idea to factoring $x^n - 1$ is always that over the splitting field, $x^n - 1 = \prod_{\alpha^n=1} (x-\alpha)$, where the product is taken over all n-th roots of unity, not necessarily primitive. While not over the splitting field, some of these terms need to be grouped together into irreducible factors, $x^n - 1 = \prod \min_\alpha(x)$, where $\min_\alpha(x)$ is the minimal polynomial for α over the appropriate base field. It follows from the binomial theorem that for $f(x) \in \mathbb{F}_p[x]$, $f(x^p) = f(x)^p$. Hence, for α a root of $f(x)$ in some extension field of $\mathbb{F}_p$, $f(\alpha^p) = f(\alpha)^p = 0$, implying $\alpha, \alpha^p, \alpha^{p^2}, \ldots$ are all roots of $f(x)$. This sequence stops when $\alpha^{p^r} = \alpha$ for some natural number r. Let E be the splitting field of $x^n - 1$ with $\gcd(n, p) = 1$, and let β be a primitive element of E. Then $\alpha = \beta^d$ is a primitive n-th root of unity with $d = (|E|-1)/n$. Then $\alpha^{p^r} = \alpha \rightarrow \beta^{dp^r-d} = 1$, or $dp^r \equiv d \mod (|E| - 1)$. Note that $|E| = p^{\mathrm{ord}_n(p)}$, where $\mathrm{ord}_n(p)$ is the smallest positive integer m such that $p^m \equiv 1 \mod n$.

We can collect this sequence of roots into the p-cyclotomic cosets modulo n (p-cosets),

$$C_s = \{s, sp, \ldots, sp^{r-1}\} \mod (n-1),$$

such that

$$\min_\alpha(x) = \min_{\beta^d}(x) = \prod_{j \in C_d} (x - \beta^j).$$

If the minimal polynomial contained any other roots, it would also need to contain all of its p powers and we could separate all of these new terms into a polynomial which divides $\min_\alpha(x)$, contradicting the irreducibility of the minimal polynomial. Hence, over $\mathbb{F}_p$, we have

$$x^n - 1 = \prod_{d|n} \min_{\alpha^d}(x). \tag{1.8}$$

The assumption $\gcd(n, p) = 1$ ensures there are no repeated roots in the factorization since $f(x^p) = f(x)^p$.

Let $\mathcal{C} = (g(x))$ be a cyclic code. Then $\mathcal{T} = \cup_s C_s$, where C_s are the p-cosets present in the construction of $g(x)$, is called the defining set of $\mathcal{C}$. As is clear from the definition, $\mathcal{T}$

completely determines $g(x)$ and vice versa, $g(x) = \prod_{j \in \mathcal{T}}(x - \alpha^j)$. The set of powers of α that are roots of $g(x)$ is called the variety (zero set) of $\mathcal{C}$, $\{\alpha^j \ : \ j \in \mathcal{T}\}$, and elements of the set are called zeros of the code. For two cyclic codes $\mathcal{C}_1$ and $\mathcal{C}_2$ with defining sets $\mathcal{T}_{\mathcal{C}_1}$ and $\mathcal{T}_{\mathcal{C}_2}$, respectfully, $\mathcal{C}_1 \subseteq \mathcal{C}_2$ if and only if $\mathcal{T}_{\mathcal{C}_2} \subseteq \mathcal{T}_{\mathcal{C}_1}$ ($g_2(x) \mid g_1(x)$).

The first of the three families of codes we will use in this work is a way of constructing a cyclic code with high minimum distance and high dimension by choosing $\mathcal{T}$ as small as possible that is a union of cyclotomic cosets with $\delta - 1$ consecutive elements. A BCH code $\mathcal{C}$ over $\mathbb{F}_p$ of length n and designed distance $2 \leq \delta < n$ is a cyclic code with defining set $\mathcal{T} = C_b \cup C_{b+1} \cup \ldots \cup C_{b+\delta-2}$ and zeros generated by a primitive element $\alpha \in \mathbb{F}_{p^m}$, where $m = \mathrm{ord}_n(p)$. A BCH code is called narrow-sense if $b = 1$ and primitive if $n = p^m - 1$. The number b is called the offset of the code. It is crucial to note that many sources define narrow-sense as $b = 0$. This definition uses the zero set $\{\alpha^{j+b}\}$ which is a shifted version of the definition above. While less common, the emphasis on the defining set over the zero set makes the choice $b = 1$ more natural for this work. The `Magma` and `Sagemath` coding theory libraries define narrow-sense as $b = 1$ and default to this parameter, although previous versions of the latter used $b = 0$.

The BCH bound says that if the defining set of a cyclic code contains a set of $\delta - 1$ consecutive integers (modulo n), then the minimum distance of the code is at least δ. BCH codes therefore have minimum distance at least δ by design and maximize dimension by containing no extra roots. The dual of a BCH is, in general, not a BCH code, as the remaining cyclotomic cosets giving $h(x)$ need no longer be consecutive; however, the dual of narrow-sense BCH codes are BCH codes.

The second family of codes we will utilize here are the cyclic RS codes, which are primitive BCH codes over $\mathbb{F}_{p^m}$ for an integer $m \geq 1$. In this case, $\mathbb{F}_{p^m}$ is the splitting field of $x^{p^m-1} - 1$ and each element α_i has minimal polynomial $x - \alpha_i$ with cyclotomic cosets of cardinality one. Hence, BCH codes are related to two fields while RS codes are only related to one. RS codes have the theoretically maximum possible distance with parameters

$[n, k, n-k+1]$. If $\mathcal{C}$ is an $[n, k, d]_{p^m}$ RS code, then $\mathcal{C}|_{\mathbb{F}_p}$ is the BCH code over $\mathbb{F}_p$ of length n and designed distance d. The proof of this follows immediately from the fact that the codewords of the BCH code are elements of $\mathbb{F}_p^n$ and the zero set of the RS is a subset of the zero set of the BCH code.

We will only be concerned with cyclic RS codes, but the more general, and original, definition of RS codes will lead us into the final family of codes we will use in this work. Let $\mathcal{P}_k(x)$ denote the set of polynomials of degree less than k in $\mathbb{F}_{p^m}[x]$. The RS code of length $n \leq p^m$ and dimension $k < n$ is given by

$$\mathrm{RS}_{p^m}(n, k) = \{(f(\alpha_1), \ldots, f(\alpha_n)) \mid f(x) \in \mathcal{P}_k(x)\},$$

where $\alpha_i \in \mathbb{F}_{p^m}$. The most common case $n = p^m$ is the extended code of the cyclic definition, but only the case $n = p^m - 1$ is, in general, cyclic.

Theorem 1.1.6

Let α be a primitive element of $\mathbb{F}_{p^m}$ and let $0 \leq k \leq n = p^m - 1$ be integers. Then

$$\mathrm{RS}_p(n, k) = \{(f(1), f(\alpha), f(\alpha^2), \ldots, f(\alpha^{n-1})) \mid f(x) \in \mathcal{P}_k(x)\}$$

is a cyclic code.

The proof is an application of the Chinese Remainder Theorem.

Unlike BCH codes which can have any length relatively prime with the characteristic of the field, RS codes over $\mathbb{F}_p$ have $n \leq p$ and therefore do not make good binary codes directly. Instead, one may construct a "binary RS code" using the expansion procedure for $\mathbb{F}_{2^m}/\mathbb{F}_2$ mentioned above.

Finally, we turn to the family of Reed-Muller (RM) codes. These are rather general mathematical objects and may be described and defined in a number of different ways. The generalized Reed-Muller codes are extensions of the original binary family to higher fields.

Let $(P_1, \ldots, P_{p^n})$ be points in the affine space $\mathbb{A}^m(\mathbb{F}_p)$. The Reed-Muller code over $\mathbb{F}_p$ is

$$\mathcal{RM}_p(r,m) = \{(f(P_1), \ldots, f(P_n)) \mid f \in \mathbb{F}_p[x_1, \ldots, x_m], \deg f \leq r\}, \tag{1.9}$$

where $0 \leq r \leq m(p-1)$ and $\deg f$ is the total degree of f (sum of the exponents). A missing subscript p is assumed to be 2. This definition of RM codes shows they are a possible generalization of RS codes to multivariate polynomials. An alternative definition using the $(u \mid u+v)$-construction is more useful for explicitly writing down a set of stabilizers,

$$\mathcal{RM}(r,m) = \{(u \mid u+v) \mid u \in \mathcal{RM}(r,m-1), v \in \mathcal{RM}(r-1,m-1)\}.$$

Let $G(r,m)$ be a generator matrix for $\mathcal{RM}(r,m)$ and set $G(0,m)$ to be the length-2^m all one's vector and $G(m,m)$ to be the $2^m \times 2^m$ identity matrix. Then the $(u \mid u+v)$ definition gives

$$G(r,m) = \begin{pmatrix} G(r,m-1) & G(r,m-1) \\ 0 & G(r-1,m-1) \end{pmatrix}. \tag{1.10}$$

Lastly, let $G_2 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. Then $\mathcal{RM}(r,m)$ is given by selecting all rows of weight at least 2^{m-r} from $G_2^{\otimes m}$ [12]. So binary RM codes have an underlying Kronecker product structure with a hand-selected weight distribution, potentially foreshadowing their use in QEC.

Reed-Muller codes have parameters $[2^m, \sum_{i=0}^{r} \binom{m}{r}, 2^{m-r}]$. Their minimal distance is generally below that of BCH and RS codes, but they are (classically) easier to encode and decode (with respect to hardware circuitry). The dual of an RM code is also an RM code, $\mathcal{RM}_p^{\perp}(r,m) = \mathcal{RM}_p(m(p-1)-r-1,m)$, and similar to BCH codes, $\mathcal{RM}_p(r_1,m) \subset \mathcal{RM}_p(r_2,m)$ for $r_1 \leq r_2$.

1.2 Quantum Mechanics For Mathematicians

Recall that a Banach space is a complete, normed vector space. A Banach space is called separable if it contains a countable dense subset. Let X and Y be normed vector spaces over the same ground field $\mathcal{K}$ and $T : X \to Y$ be a linear map. The norm of T is given by

$$\|T\| = \sup\{\|Tx\|_Y \, : \, x \in X, \|x\|_X \leq 1\}. \tag{1.11}$$

We say T is bounded if

$$\sup_{x \in X \setminus \{0\}} \frac{\|Tx\|_Y}{\|x\|_X} < \infty.$$

If T is bounded, then (1.11) is called the operator norm of T. If Y is a Banach space, the set of all bounded operators from X to Y, denoted by $\mathcal{B}(X, Y)$, is also a Banach space. If X is a Banach space, then $\mathcal{B}(X) = \mathcal{B}(X, X)$ is a unital Banach space (contains the identity) which is also an associative algebra over $\mathcal{K}$ with multiplication given by composition. This is called a Banach algebra. The subscripts X and Y on the norms are usually clear from context, and we drop them from here on out.

A Hilbert space $\mathcal{H}$ is a vector space equipped with an inner product $\langle \cdot, \cdot \rangle$ such that $\mathcal{H}$ is complete in the norm induced by the inner product. This inner product is a sequilinear form which is complex linear on the left in the math literature and on the right in the physics literature. Let $\mathcal{B}(\mathcal{H})$ denote the set of all bounded operators[2] on a complex separable Hilbert space $\mathcal{H}$. For any $A \in \mathcal{B}(\mathcal{H})$, the adjoint of A is the unique linear operator $A^* : \mathcal{H} \to \mathcal{H}$ such that $\langle \phi, A\psi \rangle = \langle A^*\phi, \psi \rangle$ for all $\phi, \psi \in \mathcal{H}$. An operator is called self-adjoint or Hermitian if $A = A^*$. Note that the adjoint is usually denoted by a †, pronounced “dagger", in the physics literature. Recall that A^* is bounded, $\|A^*\| = \|A\|$, and $*$ is an involution, i.e., $(A^*)^* = A$. With this extra structure, $\mathcal{B}(\mathcal{H})$ is called a $*$-algebra. A(n) (abstract) C^*-

[2]Many physical observables, such as position, momentum, and energy, are unbounded, but these can be reconstructed from bounded operators within the theory of rigged Hilbert spaces so it suffices to stick with $\mathcal{B}(\mathcal{H})$.

algebra[3] is a Banach algebra $\mathcal{A}$ for which $\|A^*A\| = \|A^*\|\|A\|$ for all $A \in \mathcal{A}$. Note that this definition makes no reference to Hilbert space. A (concrete) C^*-algebra is a subalgebra $C \subset \mathcal{B}(\mathcal{H})$ with the adjoint operator, which is closed in the operator norm topology. Every abstract C^*-algebra is isomorphic to a concrete C^*-algebra via the Gelfand-Naimark-Segal (GNS) construction, so we henceforth drop the qualifying label, and assume from here on out that all C^*-algebras are concrete. As a closed subspace of a Banach space, a C^*-algebra is also a Banach space. In particular, $\mathcal{B}(\mathcal{H})$ itself is a C^*-algebra. We will assume all C^*-algebras are unital, where I is the identity operator on $\mathcal{H}$.

If C is a C^*-algebra, then its topological dual C^* (linear functionals from C to $\mathbb{C}$) is a Banach space. A functional $\phi \in C^*$ is called positive if $\phi(A^*A) \geq 0$ for all $A \in C$. Such a map is called a state if ϕ is also normalized, i.e., $\phi(I) = 1$. The positivity and normalization conditions are required to guarantee the values $\phi(A^*A)$ may be interpreted as a probability. In this sense, a state associates an operator to its expectation value. This follows directly from the formula for ϕ given A in the GNS construction. There we find that either for a non-zero operator A there exists a state ϕ such that $\phi(A^*A) > 0$, or A is indistinguishable from 0.

Let $\mathcal{E}$ be the set of states on a C^*-algebra C. This is a convex subset of the set of linear forms on C. The extremal points of $\mathcal{E}$ are those states which cannot be written as a linear combination of two elements in $\mathcal{E}$. In the physics literature these are called pure states, with all other states being called density operators or mixed states. It is important to be able to distinguish between the two in this work, so we adopt the physics convention from here on out. The term “density" operator comes from its interpretation as a probability density in quantum statistical mechanics. The GNS construction says that pure states are projection operators.

Let A be a positive operator in $\mathcal{B}(\mathcal{H})$ and $\{e_i\}$ be an orthonormal basis of $\mathcal{H}$. Then the sum $\sum_i \langle e_i, Ae_i \rangle$ is invariant (possibly infinite) and is called the trace of A, $\mathrm{Tr}(A)$.

[3]The C in C^* stands for compact, not complex.

An element $A \in \mathcal{B}(\mathcal{H})$ is called trace class if $\mathrm{Tr}(A)$ is finite. A general $A \in \mathcal{B}(\mathcal{H})$ is called trace class if $\mathrm{Tr}(\sqrt{A^*A})$ is finite. If A is trace class, then A^* is also trace class and $\mathrm{Tr}(A) = \overline{\mathrm{Tr}(A^*)}$. If $A \in \mathcal{B}(\mathcal{H})$ is trace class, then for $B \in \mathcal{B}(\mathcal{H})$, AB and BA are also trace class with $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$. Gleason's Theorem establishes the existence and uniqueness of a trace-class operator A, given a state ϕ, such that $A \geq 0$, $\mathrm{Tr}A = 1$, and $\mathrm{Tr}(AB) = \phi(B)$ for every $B \in \mathcal{B}(\mathcal{H})$. This is important for establishing the probabilistic interpretation of states.

A von Neumann algebra is a C^*-algebra which is closed in the (strong) operator topology. This is not the only (or preferred) way to define von Neumann algebras, but it is the one of a handful of equivalent definitions that make the most sense in our context of Hilbert space. One may define abstract von Neumann algebras in a similar manner to abstract C^*-algebras above (and without reference to C^*-algebras), in which case von Neumann algebras are also known as W^*-algebras. We will not need this distinction here. The set $\mathcal{B}(\mathcal{H})$ is a von Neumann algebra. When $\dim \mathcal{H} = n < \infty$, there is no difference between von Neumann and C^*-algebras, and both are isomorphic to the $n \times n$ matrix algebra $\mathrm{Mat}(n, \mathcal{K})$ over the appropriate ground field $\mathcal{K}$. In particular, every operator is trace class when $n < \infty$.

The general spectral theorem of von Neumann provides a way to rigorously define a theory of probability from states in terms of measure theory[4]. In particular, an important consequence of Gleason's theorem is the Kochen-Specker theorem which says that there are no states which assign a probability of one to some operators and zero to the rest. (This is in direct contrast to the situation in classical mechanics.) For details on this process and how this "non-commutative" probability differs from that of classical probability theory, see e.g. [13] and [14].

With the mathematical foundations in place, quantum mechanics may then be charac-

[4]A result of Connes shows that von Neumann algebras correspond to non-commutative measure theory while C^*-algebras correspond to non-commutative topology.

terized by the Dirac-von Neumann axioms. Roughly, these say:[5]

- A quantum system is modeled by an infinite-dimensional, separable, complex Hilbert space $\mathcal{H}$. The Hilbert space of a composite quantum system is the tensor product of the Hilbert spaces of the component systems.
- The set of observables of a quantum system with the Hilbert space $\mathcal{H}$ consists of all self-adjoint operators on $\mathcal{H}$.
- The set of states of a quantum system with a Hilbert space $\mathcal{H}$ consists of all positive, normalized, trace-class operators.
- The result of a measurement of an observable $\mathcal{A}$ on a quantum system is the expectation value $\phi(A)$, where A is the operator representing $\mathcal{A}$.

The complicated nature of quantum mechanics (the uncertainty principle, Bell's inequalities, conditional probabilities, entanglement, etc) then follow directly from the structure of non-commutative von Neumann algebras. It is worth pointing out before we move on that classical mechanics may also be formalized in this manner but the resulting von Neumann algebra is abelian. The Gelfand-Naimark theorem says every abelian C^*-algebra is $*$-isometric to the algebra of continuous functions. As such, classical mechanics is missing the structure which makes quantum mechanics interesting.

The appearance of self-adjoint operators in the axioms above is not too surprising. The spectral theorem for self-adjoint operators guarantees a set of mutually orthogonal states and their eigenvalues, here corresponding to the possible outcomes of a measurement of the states, uniquely determines a self-adjoint operator. As written, the axioms suffer from some minor technicalities overlooked in the above discussion. Mathematically, the possible outcomes of measuring an observable $\mathcal{A}$ are given by the spectrum $\sigma(A)$ of the operator A

[5]We should be more careful here about the distinction between ensembles and single systems but a thorough discussion is complicated and beyond the intended scope of this work. The interested reader is referred to any of the many texts on the mathematical foundations of quantum mechanics.

representing $\mathcal{A}$. However, physically a measurement may obtain outcomes in the closure $\overline{\sigma(A)}$. Fortunately this doesn't cause any problems for the model as no measurement can distinguish between two arbitrarily close values. More importantly, not every self-adjoint operator on $\mathcal{H}$ represents a physically relevant (measurable) observable nor does every state in $\mathcal{H}$ correspond to a physically obtainable state.

The above material may be found in any reference on operator theory, C^*-algebras, von Neumann algebras, or the mathematics of (non-relativistic) quantum mechanics. A significant amount of work has gone into the relationship between these subjects and quantum theory. The operator approach, Tomita-Takesaki modular theory in particular, is especially important in quantum information theory, where it is used to establish bounds on measures of information. We do not touch on this here. We note for completeness that the above axioms do not hold globally due to the existence of superselection sectors inherent to von Neumann theory; however, they hold with respect to each sector.

A state ϕ of a quantum system with Hilbert space $\mathcal{H}$ is often denoted in the physics literature by $|\phi\rangle$ and a linear functional $\phi^* \in \mathcal{H}^*$ by $\langle\phi|$. For convenience, we will adhere to the physics convention that these are conjugate linear in $\langle\cdot|$ and linear in $|\cdot\rangle$ such that $A|\phi\rangle = |A\phi\rangle$ and $\langle\phi|A = \langle A^*\phi|$ for an appropriate operator $A \in \mathcal{B}(\mathcal{H})$. This notation dates back to Dirac and is called bra-ket (bracket) notation, $\langle\cdot|$ being a bra and $|\cdot\rangle$ a ket, which stems from the induced inner product $\langle\cdot|\,|\cdot\rangle \coloneqq \langle\cdot|\cdot\rangle = \langle\cdot,\cdot\rangle$. Expressions such as $|x\rangle\langle y| \in \mathcal{H} \otimes \mathcal{H}^*$ are extremely common in this field and should be interpreted as a linear operator on $\mathcal{H}$. In particular, let $\mathcal{H}$ be finite-dimensional and $|i\rangle$ denote an orthonormal basis for $i = 1, \ldots, \dim \mathcal{H}$. Then $\Pi_i = |i\rangle\langle i|$ is a projection operator onto the one-dimensional subspace spanned by $|i\rangle$. By the spectral theorem for self-adjoint operators, we may use this to express an operator $A \in \mathcal{B}(\mathcal{H})$ representing the observable $\mathcal{A}$ as

$$A = \sum_{i=1}^{\dim \mathcal{H}} \lambda_i \Pi_i = \sum_{i=1}^{\dim \mathcal{H}} \lambda_i |i\rangle\langle i|,$$

where $\lambda_i \in \mathbb{R}$ are eigenvalues of A. More generally, following the discussion above, a pure state is of the form $\rho = p_j|j\rangle\langle j|$ and a density operator ρ can be expressed as a sum of pure states,

$$\rho = \sum_{j=1}^{\dim \mathcal{H}} p_j |j\rangle\langle j|,$$

for $p_j \in \mathbb{C}$. The expectation value of A in the state ρ is then

$$\langle A \rangle_\rho \coloneqq \mathrm{Tr}(\rho A) = \sum_{j=1}^{\dim \mathcal{H}} p_j \langle A^* j, j\rangle = \sum_{j=1}^{\dim \mathcal{H}} p_j \langle j, Aj\rangle. \tag{1.12}$$

While this notation might seem cumbersome and unnecessary to mathematicians, it is unavoidable in this field and actually turns out to simplify many computations tremendously. The failure of this notation is in describing the inherent composability of the underlying physics [15].

Before moving on, it is appropriate to comment at this point on the last Dirac-von Neumann axiom above, concerning the measurement of an observable. At the time of measurement, the measurement device must be brought into contact with the system to be studied. Any rigorous definition of the measurement process must therefore include a complete description of the combined system of both the classical device and quantum system. This is far beyond the intended scope of this work and the reader is instead referred to [16] for more details. The general idea, however, will suffice for the proposes of this work and is as follows.

Definition 1.2.1

Let $\mathcal{H}$ be a finite-dimension Hilbert space representing the quantum system and Ω be an appropriate space representing the measurement device. Define $|i\rangle$ to be an orthonormal basis for $\mathcal{H}$ with $i = 1, \ldots, \dim \mathcal{H}$ with corresponding projectors $\Pi_i = |i\rangle\langle i|$. Then the

measurement of a density matrix

$$\rho = \sum_{i=1}^{\dim \mathcal{H}} p_i |i\rangle\langle i|$$

is an operator $\mathrm{MEAS}_\Omega : \mathcal{B}(\mathcal{H}) \to \mathcal{B}(\mathcal{H}) \times \Omega$ *which returns the classical result* i *with probability* $p_i = \mathrm{Tr}(\Pi_i \rho \Pi_i)$*, and*

$$\rho \mapsto \frac{\Pi_i \rho \Pi_i}{\mathrm{Tr}(\Pi_i \rho \Pi_i)}.$$

We see that after a measurement the resulting quantum state only has support on the subspace spanned by $|i\rangle$. In this way, the state is said to have "collapsed". This has dire consequences for any theory of QEC, as any information stored in the state ρ will be immediately destroyed by trying to read it. Of course, we can also project onto subspaces and not just a single state. In this case, the projector can be a sum, but we still need an orthonormal basis to distinguish between measurements.

The following result is also of great consequence to QEC and will force us to dramatically modify the previously described classical approach.

Theorem 1.2.2 (No-Cloning Theorem)

There is no unitary operator $U \in \mathcal{B}(\mathcal{H}) \otimes \mathcal{B}(\mathcal{H})$ *such that* $U|\phi\rangle \otimes |i\rangle \mapsto |\phi\rangle \otimes |\phi\rangle$ *for an arbitrary state* $|\phi\rangle$*.*

1.3 Quantum Error Correction

Let $q = p^m$ be a power of a prime. Normal (classical) computers store information in the bits $0, 1 \in \mathbb{F}_2$ (technically bytes, elements of $\mathbb{F}_8$). A bit can either be a 0 or a 1, not both. A quantum bit, called a qudit if $q > 2$ and a qubit if $q = 2$, is a normalized vector in $\mathbb{C}^q$. The normalization is necessary by the normalization of states from the previous section. By the axioms above, an n qudit system has Hilbert space $\mathbb{C}^{q^n} \cong \mathbb{C}^q \otimes \ldots \otimes \mathbb{C}^q$. The

corresponding von Neumann algebra is isomorphic to $\mathrm{Mat}(n, \mathbb{C})$. A global phase factor (scalar) times a state does not change the expectation values of a self-adjoint operator, so one can think of qudits as really living in projective Hilbert space.

Let $\{|i\rangle \mid i \in \mathbb{F}_q\}$ be an orthonormal basis for $\mathbb{C}^q$. Then a qudit may be a linear combination of basis vectors, $|\psi\rangle = \sum_i p_i |i\rangle$, which is often interpreted as saying a quantum bit may be simultaneously (in a superposition of) a 0 and a 1. Recall that if $\{a_i\}$ is a basis for A and $\{b_j\}$ is a basis for B, then $\{a_i \otimes b_j\}$ is a basis for $A \otimes B$. An n-qudit system may be written as $|\psi\rangle = \sum p_{a,\ldots,n} |a\rangle \otimes \ldots \otimes |n\rangle$, where $a, \ldots, n \in \mathbb{F}_q$. To keep the notation from being cumbersome, the tensor product symbol is omitted in the physics literature such that $|a \ldots n\rangle \coloneqq |a\rangle \otimes \ldots \otimes |n\rangle$. The state associated to a quantum system is $|\psi\rangle\langle\psi|$.

The initial formalism of QEC stuck to this Hilbert space and operator approach of the previous sections. Quantum codes are still often described by explicitly specifying vectors as sums of basis vectors and then discussing their orbit under the action of unitary operators. This led to the development of QEC fundamentals but left few good examples. The subsequent development of the stabilizer formalism by Gottesman [17] made constructing examples almost trivial. This switched the problem from one of analysis to one of groups. We will stick to the group perspective throughout this work, but we believe it is important to establish a rigorous connection between the two approaches. The stabilizer formalism is now dominant in QEC, and the operator formalism is studied in math and physics departments to establish fundamental limits on quantum information. This places limits on the subspaces that can be considered QECCs, their decoding procedures, the form certain operators can take, and the support these operators can have relative to the underlying subspace, which has been beneficial to the study of QEC. The operator formalism had a resurgence in the mid-2000's with the development of the so-called operator quantum error correction (OQEC), which studied the Hilbert space structure induced by the stabilizer formalism. This led to an extension of subspace-based codes to subsystem-based codes. We will refer to this in the last chapter and there is no way to rigorously understand how it works without

further extending our operator discussion above. We do this now before switching to the stabilizer formalism.

1.3.1 Operator Quantum Error Correction

Operations on quantum states may be viewed as linear maps between states. Of course, not just any linear map is going to be physically valid; it must have certain properties. Since states are positive, normalized, trace-class operators, a physically valid map must retain this notation of positivity and be trace preserving to conserve probability. A map $\Phi : X \to Y$ is positive for C^*-algebras X and Y if and only if $\Phi(A)$ is positive for all positive operators $A \in X$. Define $\Phi \otimes I_k : X \otimes I_k \to Y \otimes I_k$ in the natural way. Then Φ is called completely positive (CP) if and only if $\Phi \otimes I_k$ is positive for all positive integers k. It turns out that positive maps and CP maps have different properties [18], and quantum operations require the mathematically rich theory of completely-positive, trace-preserving (CPTP) maps. This is a natural requirement from a physical standpoint because one can always view a physical system as a subsystem of a larger physical system (such as the universe itself) and the physics of the subsystem should be invariant with respect to the way one chooses to look at it.

Quantum operations, called quantum channels in the quantum information literature in analogy to the classical channels of classical information theory, have a lot of nice properties, that make working with them feasible from the perspective of operator theory. The interested reader should consult [19] and the references there-in for an exhaustive but clear description. It suffices for the current discussion to note that CP maps may be completely characterized in four different but equivalent ways. Proving they are equivalent is easy but proving any of them characterize CP maps directly is more difficult. The two most useful descriptions in this work, and that are also provide an immediate interpretation with respect to the underlying physics, are the Choi decomposition and Stinespring's theorem [20], which are given in their absolute simplest forms below.

Theorem 1.3.1 (Stinespring's Dilation Theorem)

Let $\mathcal{A}$ be a unital C^*-algebra and $\mathcal{H}$ a separable, complex Hilbert space. For every completely positive map $\Phi : \mathcal{A} \to \mathcal{B}(\mathcal{H})$, there exists a Hilbert space K and a unital *-homomorphism $\pi : \mathcal{A} \to \mathcal{B}(\mathcal{H})$ such that $\Phi(a) = V^*\pi(a)V$, where $V : H \to K$ is a bounded operator.

Theorem 1.3.2 (Choi Decomposition)

Let $\mathcal{A}$ be a unital C^*-algebra and $\mathcal{H}$ a separable, complex Hilbert space. Every completely positive map $\Phi : \mathcal{A} \to \mathcal{B}(\mathcal{H})$ can be expressed in the form $\Phi(A) = \sum_i V_i A V_i^*$, where $\{V_i\}$ is a (possibly infinite) sequence of operators in $\mathcal{A}$.

This was first proved by Choi in the special case $\Phi : \mathbb{C}^{m\times m} \to \mathbb{C}^{n\times n}$ [21], but can be extended to a more general setting. This is often called the operator-sum representation, Choi-Kraus decomposition, or simply the Kraus decomposition, after the physicist Karl Kraus who popularized its use in quantum mechanics. In this context, the V_i are called Kraus operators.

The Stinespring representation is often more useful in a pure quantum information theory setting where one is more concerned with rigorous mathematical analysis. For example, while all valid quantum operations are unitary, a given quantum channel might not appear to act as a unitary operator on a quantum system. Stinespring says that one can always view the channel as a unitary operator acting on a larger system, often called the environment. Specifically, let ρ_Q be a density operator on a quantum system Q and let Φ be a channel acting on the appropriate Hilbert space of Q. Then, a Stinespring representation of Φ would be

$$\Phi(\rho_Q) = \mathrm{Tr}_E \left[V_{QE}^* \left(\rho_Q \otimes \sigma_E \right) V_{QE} \right]. \tag{1.13}$$

Here, we have introduced an auxiliary, non-physical/purely-mathematical system labeled by E (for environment) such that the action of the channel may be represented by a unitary operator V_{QE} acting on the combined system $\mathcal{H}_Q \otimes \mathcal{H}_E$. A density operator σ_E was intro-

duced out of necessity of the action of V_{QE} on E. It suffices to take this to be the "identity" on E, or the density operator $|0_E\rangle\langle 0_E|$ in a pure state of the ground state of the physical system, and as such is often left out of equations and implicitly assumed to be there. This is considered well-motivated as one can almost always isolate an experimental setup such that the system interacts with the environment in such a manner. Finally, as the action of Φ on Q is the only thing of interest, the system E is discarded by taking the partial trace over all of its degrees of freedom.

The partial trace in (1.13) is a bit inconvenient to deal with directly, so it is common to pass to the dual problem. The dual of a channel Φ, often denoted by $\widehat{\Phi}$ and called the complementary channel, is defined by the condition, referring back to equation (1.12), that $\mathrm{Tr}(\Phi(\rho)A) = \mathrm{Tr}\left(\rho\widehat{\Phi}(A)\right)$ for all appropriate ρ and A. This should be interpreted as switching between the Schrödinger and Heisenberg descriptions of quantum mechanics. The dual to (1.13) would be

$$\widehat{\Phi}(\rho_Q) = V_{QE}^* \left(\rho_Q \otimes \sigma_E\right) V_{QE}. \tag{1.14}$$

A physicist would recognize this as being of the form $\mathcal{E}(\cdot) = V^*(\cdot)V$. To see why this form is important, consider the action $U^*|\psi\rangle$ of a unitary operator U^* on a state $|\psi\rangle$. To represent this as a quantum channel, express this as a density operator $U^*|\psi\rangle\langle\psi|U$. Letting $\rho = |\psi\rangle\langle\psi|$ then gives a quantum channel $\mathcal{E}(\rho) = U^*\rho U$ whose action represents $U^*|\psi\rangle$ on the level of density operators instead of states. Note that it is common in the physics literature to switch the roles of the unitary and its adjoint in the above equations; the current notation has been chosen to match the form of Theorem 1.3.1.

The Choi decomposition 1.3.2 is often more useful in quantum error correction and in other situations where one has an explicit representation of the noise model for the system in terms of the matrices V_i, which allows for the calculation of useful quantities. Note that

in order to be trace preserving,

$$\mathrm{Tr}(\Phi(A)) = \mathrm{Tr}\left(\sum_i V_i A V_i^*\right) = \mathrm{Tr}\left(\sum_i V_i^* V_i A\right) = 1$$

must be true for all normalized, trace-class operators A. It follows that $\sum_i V_i^* V_i = I$. A map is unital if it preserves the identity, $\Phi(I) = I$. It follows that for unital maps $\sum_i V_i V_i^* = I$. Hence, a CP map is trace preserving if its dual map is unital. In general, quantum operations need not be unital maps. This distinction between unital and non-unital maps is what separates early work in the direction of OQEC [22, 23, 24, 25, 26, 27, 28] from the general theory in the references above.

Stripped down to its fundamentals, if $\mathcal{E}$ is a CPTP map representing an error occurring on the system, QEC is concerned with the existence of a CPTP map $\mathcal{R}$ such that $(\mathcal{R} \circ \mathcal{E})(\rho) = \rho$ for all states ρ. Such an $\mathcal{R}$, if it exists, is called the recovery operator for $\mathcal{E}$. The standard model of QEC is given by a triple $(\mathcal{R}, \mathcal{E}, C)$, where C is a subspace of the system's Hilbert space $\mathcal{H}$. The question then becomes given C and an $\mathcal{E}$ modeling the expected errors on the system, does a suitable $\mathcal{R}$ exist which "fixes" the errors introduced by $\mathcal{E}$? In such cases, C is said to be correctable for $\mathcal{E}$. Starting from a more physical perspective, one often only has $\mathcal{E}$ and the goal is to find an appropriate subspace C to encode the information such that $\mathcal{R}$ exists. One can write down conditions on when such an operator exists, the proof of which is constructive [29, 30].

An important, early result in the QEC literature is the related Knill-Laflamme theorem.

Theorem 1.3.3 (Knill-Laflamme [30])

Let $V \subset \mathcal{H}^{\otimes n}$ and $\mathcal{E}$ be a channel with a Kraus operators $\{E_a\}$. Then V can detect (correct) elements of $\mathcal{E}$ if and only if $\langle \Psi_i E_a, E_b \Psi_j \rangle = c_{ab}\delta_{ij}$, where $\{|\Psi_i\rangle\}$ is the basis of V and c_{ab} is non-negative, real.

If the matrix of coefficients (c_{ab}) does not have full rank, different errors from $\mathcal{E}$ have the same effect on the subspace and cannot be distinguished. If this occurs, V is called

degenerate (with respect to $\mathcal{E}$).

Let $\mathcal{B}(\mathcal{H})$ be finite, $\mathcal{E} : \mathcal{B}(\mathcal{H}) \to \mathcal{B}(\mathcal{H})$ be a given CPTP map with Kraus operators $\{E_a\}$, and $\mathcal{A}$ be the algebra generated by E_a and E_a^*. The following result then gives the unique decomposition of $\mathcal{A}$, called the interaction algebra, up to unitary equivalence of the form

$$\mathcal{A} \cong \bigoplus_i \left(\mathcal{M}_{m_i} \otimes I_{n_i}\right), \tag{1.15}$$

where I_{n_i} is the identity on $\mathbb{C}^{n_i}$ and accounts for the degeneracies of $\mathcal{M}_{m_i}$.

Theorem 1.3.4

Every finite-dimensional C^*-algebra $\mathcal{A}$ is $*$-isomorphic to the direct sum of full matrix algebras, $\mathcal{A} \cong \mathcal{M}_{m_1} \oplus \ldots \oplus \mathcal{M}_{m_k}$. In particular, every finite-dimensional C^*-algebra is unital.

The structure of (1.15) implies the commutant $\mathcal{A}' = \{\rho \in \mathcal{B}(\mathcal{H}) \mid E\rho = \rho E, \forall E \in \{E_a, E_a^*\}\}$ has the form

$$\mathcal{A}' \cong \bigoplus_i \left(I_{m_i} \otimes \mathcal{M}_{n_i}\right). \tag{1.16}$$

We are interested in the set of fixed points of $\mathcal{E}$, $\text{Fix}(\mathcal{E}) = \{\rho \in \mathcal{B}(\mathcal{H}) \mid \mathcal{E}(\rho) = \rho\}$, as these are those states immune to the errors modeled by $\mathcal{E}$. This equals is $\mathcal{A}'$ if and only if $\mathcal{E}$ is unital,

$$\mathcal{E}(\rho) = \sum_a E_a \rho E_a^* = \rho\left(\sum_a E_a E_a^*\right) = \rho.$$

When $\mathcal{E}$ is unital, this is called the theory of noiseless subsystems [26, 27, 28], and when $m_i = 1$ for all i this is called the theory of decoherence-free subspaces [22, 23, 24, 25], both of which predate the general theory of OQEC.

Equation (1.15) induces a decomposition of the Hilbert space in the form $\mathcal{H} = \bigoplus_i (\mathcal{H}_i^A \otimes \mathcal{H}_i^B)$, with $\dim \mathcal{H}_i^A = m_i$ and $\dim \mathcal{H}_i^B = n_i$. Without loss of generality, consider the decomposition $\mathcal{H} = (\mathcal{H}^A \otimes \mathcal{H}^B) \oplus K$, where K contains the rest of the terms in the direct

sum. Define the semigroup $\mathfrak{A}$ to be the set of states restricted to $\mathcal{H}^A \otimes \mathcal{H}^B$,

$$\mathfrak{A} = \{\rho \in \mathcal{B}(\mathcal{H}) \mid \rho = \rho^A \otimes \rho^B,\ \rho^A \in \mathcal{B}(\mathcal{H}^A),\ \rho^B \in \mathcal{B}(\mathcal{H}^B)\},$$

and let $\Pi_{\mathfrak{A}}$ be the projection of $\mathcal{H}$ onto $\mathcal{H}^A \otimes \mathcal{H}^B$. Then define the B-sector of $\mathfrak{A}$ to be correctable for $\mathcal{E}$ if there exists a recovery operator $\mathcal{R}$ such that

$$(\mathrm{Tr}_A \circ \Pi_{\mathfrak{A}} \circ \mathcal{R} \circ \mathcal{E})(\rho) = \mathrm{Tr}_A(\rho), \forall \rho \in \mathfrak{A},$$

where Tr_A is the partial trace of the subsystem A. Our model of error correction is thus now defined by the triple $(\mathcal{R}, \mathcal{E}, \mathfrak{A})$.

In essence, by tracing out the degrees of freedom in A, we only care about what happens to the information stored in B. In particular, we would like a physical system such that for all ρ^A and ρ^B there exists a τ^A such that $\mathcal{E}(\rho^A \otimes \rho^B) = \tau^A \otimes \rho^B$, i.e., $\mathcal{R} = I$. Thus storing information only in the B-sector achieves a form of passive error correction. With this in mind, $\mathcal{H}^A$ is said to be noisy and $\mathcal{H}^B$ noiseless. One can constructively write down conditions on when such a situation exists [31, 32, 33]. In particular, we need $\mathcal{A}' \neq \mathbb{C}I$, since the operators diagonal with respect to the chosen basis (with all of the $n_i = 1$) correspond to classical states.

1.3.2 The Stabilizer Formalism

We finally describe the stabilizer formalism of QEC. In order to build an error-correcting code, we first need to establish what the errors are. We will need a different model for a different choice of errors. The best set of errors was rigorously developed in the early QEC literature [34, 35, 36]; see also [37, 38, 39, 40].

Definition 1.3.5 ([37])

Let G be a group of order d^2 with identity element e. A nice error basis on $\mathbb{C}^d$ is a set $\mathcal{E} = \{\rho(g) \in \mathcal{U}(d) \mid g \in G\}$ of unitary matrices such that

(i) $\rho(e) = I$

(ii) $\mathrm{Tr}(\rho(g)) = n\delta_{g,e}\, \forall g \in G$

(iii) $\rho(g)\rho(h) = \omega(g,h)\rho(gh)\, \forall g, h \in G$,

where $\omega(g,h)$ *is a nonzero complex number depending on* $(g,h) \in G \times G$.

Conditions (i) and (iii) say ρ is a projective representation of G. Condition (ii) says that this (irreducible) representation is faithful and the matrices form an orthonormal basis with respect to the Hilbert-Schmidt inner product, $\langle A, B\rangle = \mathrm{Tr}(A^*B)/d$. Reference [37] classified all viable groups. It is known that nice error bases are also unitary bases. By the previous discussions, a physical action on a qudit is represented by a unitary matrix, $U|\psi\rangle$. Let $\{|i\rangle \mid i \in \mathbb{F}_q\}$ be an orthonormal basis for $\mathbb{C}^q$, and consider the two operators defined by

$$X(a)|i\rangle = |i + a\rangle \tag{1.17}$$

and

$$Z(b)|i\rangle = \omega^{\mathrm{Tr}(bi)}|i\rangle, \tag{1.18}$$

where ω is a primitive p-th root of unity in $\mathbb{C}$, Tr is the absolute trace for $\mathbb{F}_q/\mathbb{F}_p$, and the result of the trace is naturally embedded in $\mathbb{C}$. Over $\mathbb{F}_2$ we simplify the notation and find $X|0\rangle = |1\rangle$, $X|1\rangle = |0\rangle$, $Z|0\rangle = |0\rangle$, and $Z|1\rangle = -|1\rangle$. Hence X plays the role of the familiar bit-flip error from classical computing, but Z, called a phase-flip error, is completely new, e.g., $Z(|0\rangle \pm |1\rangle)/\sqrt{2}) = (|0\rangle \mp |1\rangle)/\sqrt{2}$. These are physically motivated from elementary quantum mechanics.

It is easy to see that (1.17) and (1.18) satisfy the commutation relations

$$[X(a), X(a')] = [Z(b), Z(b')] = 0, \tag{1.19}$$

$$X(a)Z(b) = \omega^{-\mathrm{Tr}(ba)}Z(b)X(a). \tag{1.20}$$

The second relation gives

$$X(a)Z(b)X(a')Z(b') = \omega^{\mathrm{Tr}(ba')}X(a+a')Z(b+b'), \tag{1.21}$$

$$(X(a)Z(b))^{\ell} = \omega^{\mathrm{Tr}\left(\frac{\ell(\ell-1)}{2}b\cdot a\right)}X(\ell a)Z(\ell b). \tag{1.22}$$

Since $X^p = Z^p = I$, the spectrum of the representations of both operators are all the p-th complex roots of unity. When $q = p$, operators with these conditions are called a Weyl pair.

Definition 1.3.6

A Weyl pair (A, B) in dimension d is a pair of $d \times d$ unitary matrices such that $A^d = B^d = 1$ and $AB - \eta BA = 0$, where $\eta^d = 1$.

We may extend the notation to $a, b \in \mathbb{F}_q^n$ by $X(a) = X(a_1) \otimes \ldots \otimes X(a_n)$ and similarly for $Z(b)$. This should be interpreted as $X(a_i)$ operating on the ith qudit. It is useful to recall that $(A \otimes B)(C \otimes D) = (AC \otimes BD)$. A straightforward proof shows that $\{X(a)Z(b) \mid a, b \in \mathbb{F}_q^n\}$ forms a nice error basis and a basis for the $q^n \times q^n$ unitary matrices,

$$U = \sum_{a \in \mathbb{F}_q^n} \sum_{b \in \mathbb{F}_q^n} \langle U, X(a)Z(b)\rangle X(a)Z(b). \tag{1.23}$$

If we can correct any errors of this form acting on our system, we can correct any linear combination of them, i.e., any unitary error.

We can promote this set to a group

$$\mathcal{P}_n = \{\eta^c X(a)Z(b) \mid a, b \in \mathbb{F}_q^n, c \in \mathbb{F}_p\}, \tag{1.24}$$

where η is a primitive p-th root of unity for p odd and a primitive $2p$-th root of unity for p even. This latter condition is necessary to fix the normalizer and essentially gives a double cover of the group. The inclusion of the scalars η^c are required from equation (1.21) for closure. This is often called the (generalized) Pauli group in the physics literature since in $\mathbb{F}_2$, X and Z acting on a single qubit are the standard σ_x, σ_z Pauli operators of elementary

quantum mechanics (generators of the Lie algebra $\mathfrak{su}(2)$). Mathematically, these satisfy the commutation relations of the Heisenberg group when $q = p$. The standard description of the Heisenberg group in math textbooks of matrices of the form

$$\begin{pmatrix} 1 & y & z \\ 0 & 1 & x \\ 0 & 0 & 1 \end{pmatrix}$$

are an irreducible representation of the group over $\mathbb{Z}_p$, whereas $\mathcal{P}_n$ is a faithful, irreducible, unitary representation. The name Heisenberg-Weyl (Weyl-Heisenberg) group, HW_n is also used in the physics literature due to the Weyl pair.

This is one of the main objects in this work, so we will discuss some of its properties. Repeatedly applying (1.20) gives

$$(\eta^c X(a)Z(b))(\eta^{c'} X(a')Z(b')) = \omega^{\mathrm{Tr}(b\cdot a' - b'\cdot a)}(\eta^{c'} X(a')Z(b'))(\eta^c X(a)Z(b)). \tag{1.25}$$

Hence the derived subgroup is $\mathcal{P}_n' = \{\omega^c I\} \cong \mathbb{Z}_p$ and the center is $\mathcal{Z}(\mathcal{P}_n)$ which is $\mathbb{Z}_p$ or $\mathbb{Z}_{2p}$ for $p \neq 2$ and $p = 2$, respectively. Equation (1.20) also gives

$$\begin{aligned} (X(a)Z(b))^T &= \omega^{-\mathrm{Tr}(b\cdot a)} X^T(a) Z^T(b), \\ (X(a)Z(b))^* &= \omega^{\mathrm{Tr}(b\cdot a)} X^*(a) Z^*(b) = \omega^{\mathrm{Tr}(b\cdot a)} X(-a)Z(-b). \end{aligned}$$

Consider the action on itself by conjugation, $\cdot : \mathcal{P}_n \times \mathcal{P}_n \to \mathcal{P}_n$

$$(g = \eta^{c'} X(a')Z(b'), \eta^c X(a)Z(b)) \mapsto g(\eta^c X(a)Z(b))g^{-1} = \omega^{\mathrm{Tr}(b\cdot a' - b'\cdot a)}(\eta^c X(a)Z(b)). \tag{1.26}$$

The stabilizer of the action are those elements which $\mathrm{Tr}(b \cdot a' - b' \cdot a) = 0$, which is also the centralizer by (1.25). If $a = b = 0$, then the conjugacy class is $\{\eta^c I\}$ which is of order p if $p \neq 2$ and $2p$ if $p = 2$. Otherwise, (1.26) is order p for $p \neq 2$. If $p = 2$, the ω term

has order p and the η term either has order p or $2p$, which splits the conjugacy class in two. The exponent of the group is therefore $\exp(\mathcal{P}_n) = p$ if $p \neq 2$ and $\exp(\mathcal{P}_n) = 2p$ if $p = 2$. Finally, we have that $|\mathcal{P}_n| = \exp(\mathcal{P}_n) q^{2n}$.

Let $\mathcal{S} \leq \mathcal{P}_n$ and define a quantum stabilizer code to be the joint $+1$-eigenspace of operators in $\mathcal{S}$,

$$\mathcal{Q} = \bigcap_{S \in \mathcal{S}} \left\{ |v\rangle \in \mathbb{C}^{q^n} \mid S|v\rangle = |v\rangle \right\}. \tag{1.27}$$

This is a QECC in the sense of the previous sections. Of course, not all subgroups $\mathcal{S} \leq \mathcal{P}_n$ are going to produce "good" quantum stabilizer codes. For starters, in order for $\mathcal{Q}$ to exist (not be the trivial subspace), $\mathcal{S}$ must be simultaneously diagonalizable. This forces $\mathcal{S}$ to be abelian. Furthermore, no two elements of $\mathcal{S}$ may be scalar multiples of each other. It also goes without saying that elements of $\mathcal{S}$ must have a $+1$-eigenspace. In $\mathbb{F}_2$, for example, $X^2 = Z^2 = I$ and so have eigenvalues ± 1 by Cayley-Hamilton. The operator $X(a)Z(a)$ has purely complex eigenvalues and therefore cannot be an element of $\mathcal{S}$ if $\mathcal{Q}$ is to be non-trivial, but $iX(a)Z(a)$ is permitted. These kinds of conditions are often included in the definition of the stabilizer group in the physics literature but is not strictly necessary from a mathematical point of view. If we assume $\mathcal{S}$ is abelian, we can generalize this further by taking an additive character of $\mathcal{S}$, $\chi : \mathcal{S} \to \mathbb{C}$ with $\chi(I) = 1$, and considering $S|v\rangle = \chi(S)|v\rangle$. Applying S, $\exp(\mathcal{P}_n)$ times we see that $\chi(S)$ must be a power of η. Not all choices of $\chi(S)$ give non-trivial $\mathcal{Q}$ for fixed $\mathcal{S}$ since a $\chi(S)$-eigenspace might not exist for a given operator, as before. We call $\mathcal{S}$ the stabilizer group of $\mathcal{Q}$ and elements of $\mathcal{S}$ stabilizers.

An orthogonal projector onto the codespace $\mathcal{Q}$ is

$$\Pi_{\mathcal{S}} = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} S. \tag{1.28}$$

Information is encoded in the system via $\mathrm{enc} : \mathbb{C}^{q^k} \to \mathbb{C}^{q^n} : |\psi\rangle \mapsto \Pi_{\mathcal{S}}|\psi\rangle$. Definition 1.2.1 says this can be physically constructed by measuring a generating set of operators of

$\mathcal{S}$. If the result of a measurement is not $+1$, a correction operation is applied to align the system into the $+1$-eigenspace of this operator.

Now consider the action of an arbitrary element $E \in \mathcal{P}_n$ on the encoded system, $E\Pi_{\mathcal{S}}|\psi\rangle$. It follows from (1.14) that errors (elements of $\mathcal{P}_n$) act on $\mathcal{S}$ by conjugation. Suppose an error E acts on the system and does not commute with at least one generator of $\mathcal{S}$. Then some term in $|\psi\rangle$ will be left with a phase factor (1.26) and $|\psi\rangle \notin \mathcal{Q}$. Measuring again will produce a $+1$ for every generator which commutes with E and some ω^c for any generator which does not commute with E. Fixing the ordering of the generators, the vector of measurement results is called the syndrome of E. Decoding in QECC is therefore determining a proper correction procedure to make the syndrome the zero vector. This is an implementation of the recovery operator of the previous section. Errors which commute with $\mathcal{S}$ are not detectable with this method. The set of detectable errors for $\mathcal{S}$ is therefore $\mathcal{S}\mathcal{Z}(\mathcal{P}_n) \cup (\mathcal{P}_n \backslash C_{\mathcal{P}_n}(\mathcal{S}))$, where $C_{\mathcal{P}_n}(\mathcal{S})$ is the centralizer of $\mathcal{S}$ in $\mathcal{P}_n$. Since $\mathcal{S}$ is abelian, $\mathcal{S} \leq C_{\mathcal{P}_n}(\mathcal{S})$. The actions of elements of $C_{\mathcal{P}_n}(\mathcal{S})\backslash\mathcal{S}$ represent non-detectable, non-trivial actions on the encoded information. One may reinterpret the Knill-Laflamme conditions in this setting but we will have no direct use for this here.

The Heisenberg group has long been studied with its connection to finite geometries. Consider the homomorphism $\Psi : \mathcal{P}_n \to \mathbb{F}_q^n \times \mathbb{F}_q^n : \eta^c X(a)Z(b) \mapsto (a, b)$ with kernel $\mathcal{Z}(\mathcal{P}_n)$. By (1.21) and (1.22), $\Psi(\mathcal{P}_n)$ is closed under addition and $\mathbb{F}_p$-multiplication. The trace-symplectic bilinear form $\mathrm{Tr}(b \cdot a' - b' \cdot a)$ induces a symplectic geometry on $\Psi(\mathcal{P}_n)$. The same equations show that $\Psi(\mathcal{S})$ is a subspace for $\mathcal{S} \leq \mathcal{P}_n$. Bases for this space are in one-to-one correspondence with generating sets for $\mathcal{S}$. Clearly $\Psi(C_{\mathcal{P}_n}(\mathcal{S})) \cong \Psi(\mathcal{S})^{\perp}$, where orthogonality is taken with respect to the symplectic form. As a finite-dimensional vector space, $\Psi(\mathcal{S})$ has a basis of rank $n - k$ for some $0 \leq k \leq n$ such that

$$q^{2n} = |\Psi(\mathcal{P}_n)| = |\Psi(\mathcal{S})||\Psi(\mathcal{S})^{\perp}| = p^{n-k}|\Psi(\mathcal{S})^{\perp}|.$$

The sum of all orthogonal projectors is the identity and $\Pi_{\mathcal{S}}$ fixes $n-k$ subspaces, so the size of the $\mathcal{Q}$ is $\mathrm{Tr}\Pi_{\mathcal{S}} = q^n/|\Psi(\mathcal{S})|$. In the special case $q = p$, $|\Psi(\mathcal{S})^{\perp}| = p^{n+k}$, $|\mathcal{Q}| = q^k$, and we say $\mathcal{S}$ encodes k qudits of information into n qudits. It is currently unknown how to think about the case $p^{n-k} \nmid q^{2n}$ as the number of encoded bits is non-integral. Working backwards to the group cardinalities, the size of the orbits of the action are $|\mathcal{P}_n|/|C_{\mathcal{P}_n}(\mathcal{S})| = |\mathcal{P}_n|/|\mathcal{Z}(\mathcal{P}_n)||\Psi(\mathcal{S})^{\perp}|$. When $q = p$, this is p^{n-k}. Since each space is size p^k, the action is transitive on the eigenspaces on the representations of $\mathcal{S}$.

From now on let $\mathcal{C} \coloneqq \Psi(\mathcal{S})$. This is called a stabilizer code[6] and is equivalent to a classical error-correcting code of length $2n$ over $\mathbb{F}_q$ whose rows are trace-symplectic orthogonal, i.e., $\mathcal{C}$ is an isotropic vector subspace of $\mathbb{F}_q^{2n}$. This is a bit inconvenient to work with since there are only n physical qudits and the operators acting on the i-th qudit are determined by the i-th and $(i+n)$-th elements of the vector. It is common in the physics literature to follow Ψ with a map to the quadratic extension such that $\mathcal{C}$ may be viewed as a subspace of $\mathbb{F}_{q^2}^n$. The length-$2n$ object is referred as the symplectic form(at)[7] of the length-n object. The quadratic form(at) is rarely used in the modern literature, perhaps due to lack of familiarity with algorithms over $\mathbb{F}_4$ as opposed to $\mathbb{F}_2$, but offers some theoretical benefits. For example, the trace-symplectic form can be replaced with familiar algebra over $\mathbb{F}_{q^2}$. Also, the weight of a length-$2n$ vector is the symplectic weight,

$$\mathrm{swt}((a,b)) = |\{i \mid (a_i, b_i) \neq (0,0), 1 \leq i \leq n\}|,$$

whereas the quadratic extension uses the standard Hamming weight. On the other hand, the dual space $\mathcal{C}^{\perp} \coloneqq \Psi(C_{\mathcal{P}_n}(\mathcal{S}))$ is easiest to compute by expressing a set of generators for $\mathcal{S}$ as a matrix $(A \mid B)$ and then taking the (right) kernel of $(B \mid -A)$ using elementary linear

[6]The literature sometimes calls both $\mathcal{Q}$ and $\mathcal{C}$ stabilizer codes but the terminology used here is historically correct and more accurate. In particular, $\mathcal{C}$ should not be thought of as a quantum object.

[7]The word form as in format should not to be confused with the precise mathematical definition of these terms.

algebra. The minus sign comes from the matrix representation of the symplectic form

$$\Omega = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}.$$

Note that unlike how the dual of a classical code is another classical code, the centralizer of $\mathcal{S}$ is necessarily nonabelian and therefore does not define its own a quantum stabilizer code $\mathcal{Q}$. This has profound consequences as much of the powerful theory of classical codes derives from the relationship between a code and its dual. It is common to treat $\mathcal{C}$ and $\mathcal{C}^{\perp}$ as fundamental and purely classical objects, enabling classical tools such as the MacWilliams identities, but care must be taken in their interpretation due to this.

Technically, as previously mentioned, stabilizer codes over $\mathbb{F}_q$ with $q \neq p$ are only closed under multiplication by $\mathbb{F}_p$ corresponding to (1.22) and so are only $\mathbb{F}_p$-additive modules. In classical coding theory these are called additive codes. One must be careful to avoid unwanted multiplications in algorithmic implementations of the stabilizer formalism over $\mathbb{F}_q$, for example, during Gaussian elimination. To distinguish the classical and quantum cases, QECCs are denoted by $((n, q^k, d))_q$ or $[[n, k, d]]_q$. The minimum distance of a QECC is defined to be the minimum weight of all non-detectable, non-trivial errors,

$$d = \min_{E \in \mathcal{C}_{\mathcal{P}_n}(\mathcal{S}) \setminus \mathcal{S}} \mathrm{wt}(E). \tag{1.29}$$

The previously described measurement process used to implement QEC is equivalent to mapping the error to a vector and a generating set for $\mathcal{S}$ to a parity-check matrix for $\mathcal{C}$, which in this context is referred to as the stabilizer matrix. The resulting measured syndrome is equivalent to the matrix-vector multiplication described in Section 1.1. (This is why we chose $\Psi(\mathcal{S})$ to have rank $n-k$ instead of k.) A parity-check description compared to the standard generator-matrix approach is necessary here because the only information about the system available to the experimenter are the measured syndromes. Combined

with its natural interpretation, the parity-check description is the reason (1.29) is defined with respect to its dual.

Suppose $\mathcal{S}$ is a stabilizer group with corresponding quantum stabilizer code $\mathcal{Q}$. The projector $\Pi_{\mathcal{S}}$ fixes $n-k$ degrees of freedom of the n-dimensional Hilbert space. The remaining k dimensions are free and may take any value without altering the fixed subspaces. This is where we are going to store our information; this is called the encoded or logical (sub)space. Since eigenspaces corresponding to distinct eigenvalues are orthogonal to each other, this induces the decomposition $\mathcal{H} = \mathcal{Q} \oplus \mathcal{Q}^{\perp}$. The stabilizer formalism does not specify the encoded subspace directly, it specifies the orthogonal subspace. A basis for the logical space induces another set of k elements of $\mathcal{P}_n$. These must commute with $\mathcal{S}$ or at least be an automorphism of $\mathcal{S}$ in order to not interfere with the fixed degrees of freedom, and they cannot be in $\mathcal{S}$ itself if they are to be non-trivial on this space. Stabilizer groups are abelian so these two conditions are equivalent, and elements act by conjugation so the subgroup is $N_{\mathcal{P}_n}(\mathcal{S})/\mathcal{S} \cong C_{\mathcal{P}_n}(\mathcal{S})/\mathcal{S}$. The physics literature is split on using the normalizer or the centralizer; both have valid interpretations depending on how one views the problem: Schrödinger versus Heisenberg.

If we store information into the vectors of $\mathcal{Q}$, we want to be able to manipulate the information. As before, operations are $q^k \times q^k$ matrices for which $\mathcal{P}_k$ is a basis. Choosing an orthonormal basis $\{|i\rangle_L \mid i \in \mathbb{F}_q\}^{\otimes k}$ for $\mathbb{C}^{q^k}$, elements $\eta^c X(a)Z(b) \in \mathcal{P}_k$ act as in (1.17), (1.18). Extending the previous notation to include the dimension, $\Psi_k(\mathcal{P}_k) \cong \mathbb{F}_q^{2k}$. Recall that any finite-dimensional symplectic vector space has a basis, called a symplectic basis, of $2k$ vectors $\{u_1, \ldots, u_k, v_1, \ldots, v_k\}$ satisfying

$$\langle u_i, u_j \rangle_s = \langle v_i, v_j \rangle_s = 0 \quad , \quad \langle u_i, v_j \rangle_s = \delta_{ij},$$

where the inner product is given by the symplectic form. This induces a generating set on

$\mathcal{P}_k$ following similar commutation relations, $X_i \coloneqq \Psi_k^{-1}(u_i)$ and $Z_i \coloneqq \Psi_k^{-1}(v_i)$,

$$[X_i, X_j] = [Z_i, Z_j] = 0 \quad , \quad [X_i, Z_j] = \eta^{-1}\delta_{ij}. \tag{1.30}$$

These are called logical operators (X-logicals, Z-logicals). Unfortunately, we don't have direct access to this space; the physical system is n qudits, so we must convert the logical operators into equivalent operators in $\mathcal{P}_n$. To do this we use the map $\gamma : \mathbb{F}_q^{2n} \to \mathbb{F}_q^{2k} : (a,b) \mapsto (a,b)\Psi_n(S)$ which sends elements to cosets of $\Psi_n(S)$. Then $\overline{X}_i \coloneqq \Psi_n^{-1}(\gamma^{-1}(u_i))$ and $\overline{Z}_i \coloneqq \Psi_n^{-1}(\gamma^{-1}(v_i))$ are equivalence class representatives of the operators we are looking for. There is no canonical choice of basis for $\mathbb{F}_q^{2k}$ and there are numerous choices one can make in arriving at a representative $\{\overline{X}_i\}$, $\{\overline{Z}_i\}$. Cleve and Gottesman provided an algorithm for deriving a set of representatives from the stabilizer matrix in symplectic form with the development of the stabilizer formalism [41], but this relies on putting the matrix into the "standard form"

$$\begin{array}{r} \\ r\{ \\ n-k-r\{ \end{array} \left(\begin{array}{ccc|ccc} \overbrace{I}^{r} & \overbrace{A_1}^{n-k-r} & \overbrace{A_2}^{k} & \overbrace{B}^{r} & \overbrace{C_1}^{n-k-r} & \overbrace{C_2}^{k} \\ 0 & 0 & 0 & D & I & E \end{array} \right),$$

where each letter is an appropriate block matrix resulting from a row-reduction procedure, which is less intuitive than the simple system of equations above. We will return to this in more detail in Chapter 4.

The stabilizers and logical operators are the two most important objects related to a QECC. The overline is often replaced with a subscript L, for logical, in the literature. It is unfortunate that the lines between the k-dimensional objects and the n-dimensional objects are blurred throughout the literature, and both are called logical operators. We will refer to $\{\overline{X}_i\}$, $\{\overline{Z}_i\}$ as physical logical operators. The relationships between the objects are shown in Figure 1.1, where ρ is the map from the Heisenberg group to its unitary representation.

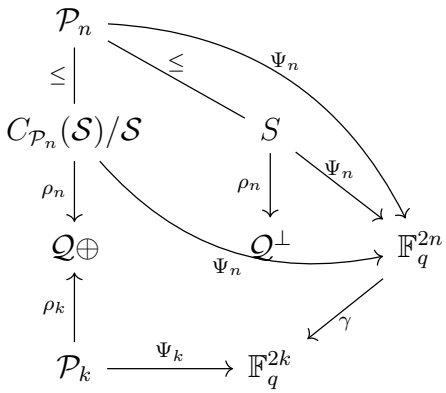


Figure 1.1: A summary of the relationships between the various fundamental mathematical objects in the stabilizer formalism.

1.3.3 CSS Codes

Given a classical error-correcting code which satisfies the properties required in the stabilizer formalism, one may interpret the vectors as elements of the stabilizer group and then construct a quantum stabilizer code. Unfortunately, the majority of classical codes are not applicable. Immediately following the birth of QEC, predating the stabilizer formalism, Calderbank and Shor [42] and Steane [43] (CSS) developed a way to convert some classical codes to stabilizer codes. The original construction takes the subspace approach but it is now common to view it with respect to the stabilizer formalism.

Theorem 1.3.7 (CSS Construction, Corollary 12.3 [4])

(i) Let $\mathcal{C}_1$ and $\mathcal{C}_2$ denote two classical linear codes with parameters $[n, k_1, d_1]_q$ and $[n, k_2, d_2]_q$, respectively, such that $\mathcal{C}_2 \subset \mathcal{C}_1$. Then there exists an $[[n, k_1 + k_2 - n, d]]_1$ stabilizer code, CSS$(\mathcal{C}_2, \mathcal{C}_1)$, with minimum distance $d = \min\{\mathrm{wt}(c) \mid c \in (\mathcal{C}_1 \backslash \mathcal{C}_2) \cup (\mathcal{C}_2^\perp \backslash \mathcal{C}_1^\perp)\}$.

(ii) If $\mathcal{C}$ is a classical linear $[n, k, d]_q$ code with $\mathcal{C}^\perp \subset \mathcal{C}$, then there exists an $[[n, 2k-n, \geq d]]_q$ stabilizer code, CSS$(\mathcal{C})$.

The proof of this takes vectors $(a, b) \in \mathcal{C}_2^\perp \times \mathcal{C}_1$ such that $\mathrm{Tr}(b \cdot a' - b' \cdot a) = \mathrm{Tr}(0 - 0) = 0$. Generators of the stabilizer group in this construction may be taken to have either $a = 0$ or

$b = 0$, and the stabilizer matrix may taken to be $H(\mathcal{C}_2^{\perp}) \oplus H(\mathcal{C}_1)$, where $H(\cdot)$ is the parity-check matrix of the corresponding code. Case (b) is a special case of (a) with stabilizer matrix $H(C^{\perp}) \oplus H(C^{\perp})$. More generally, codes where the elements of $\mathcal{S}$ can be put into the form $a = 0$ or $b = 0$ are called CSS codes.

While this might seem overly restrictive, the overwhelming majority of QECCs in the current literature are CSS. For starters, they are easier to construct and analyze. If the stabilizer matrix is of the form $A \oplus B$, the orthogonality condition for $q = p$ simplifies to $AB^T = 0$. Significant effort has been spent constructing binary matrices satisfying this condition which also have other desirable QEC properties such as rows with low Hamming weight and a high minimum distance, the former being experimentally preferable. Many of the currently favored codes come from taking these matrices to be boundary maps of a cellulation of a manifold or, more generally, adjacency matrices of pairs of graphs.

Since the X and Z components of CSS codes are independent classical linear codes, they have their own minimum distances, d_X, d_Z, with $d = \min\{d_X, d_Z\}$. This asymmetry can be useful when either bit flips or phase flips experimentally dominate, as is true in many state-of-the-art systems. If we are willing to ignore possible degeneracy and X-Z correlations, CSS codes can also use their corresponding classical decoders. Separating the X and Z problems are the most popular form of decoding.

The canonical example of a CSS code is the Steane code [43], which is constructed from the classical (binary) $[7, 4, 3]$ Hamming code,

$$H_1 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix},$$

and its dual

$$H_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix},$$

which is a $[7, 3, 3]$ code. This is therefore a $[[7, 4-3, 3]]$ quantum code. The canonical example of a quantum code that is not a CSS code is the $[[5, 1, 3]]$-code, which has stabilizer matrix

$$\begin{pmatrix} 1 & \alpha & \alpha & 1 & 0 \\ 0 & 1 & \alpha & \alpha & 1 \\ 1 & 0 & 1 & \alpha & \alpha \\ \alpha & 1 & 0 & 1 & \alpha \end{pmatrix}$$

over $\mathbb{F}_4$.

1.3.4 Quantum Topological Codes

The most thoroughly studied quantum codes is the class of topological stabilizer codes, developed immediately following the introduction of stabilizer codes [44, 45, 46]. The literature on this topic is vast, spreading in many disjoint directions. Here we briefly discuss the idea then present the codes we will utilize later. The general construction is as follows: Let M be a compact manifold of dimension n and let T be a cellulation of the manifold. Orientation has no physical meaning here, so define the k-chains of T, $C_k(T)$, over $\mathbb{Z}_2$ for $k \leq n$. We look for two subspaces of $C_k(T; \mathbb{Z}_2)$ to use the CSS construction. The natural choices are the boundaries $\mathcal{B}_p = \operatorname{im} \partial_{p+1}$ and the cycles $\mathcal{Z}_p = \ker \partial_p$. These subspaces are not orthogonal but since $\mathcal{B}_p \subseteq \mathcal{Z}_p$, $\mathcal{B}_p$ and $\mathcal{Z}_p^\perp$ are orthogonal and CSS$(\mathcal{B}_p, \mathcal{Z}_p)$ is a valid quantum code. In order to understand these codes, we thus need to understand the homology groups $H_p(M; \mathbb{Z}_2) = \mathcal{Z}_p/\mathcal{B}_p$ and $\mathcal{B}_p^\perp/\mathcal{Z}_p^\perp$. Of course, there's no particular reason we must use the CSS construction here but it is common and convenient.

Here we will only be concerned with tilings (tessellations) of compact surfaces. The dual spaces follow from a special case of Poincaré duality but they are simple enough to describe explicitly. The set B_1 is generated by the borders of faces. Let f denote a face enclosed with edges e_i and f^* be its corresponding dual vertex. Then the boundary $\partial f = e_1 + \ldots e_m$ is the sum of all edges with tail f^*, $\delta f^* = e_1^* + \ldots + e_m^*$ in the dual graph. Thus elements of $B_1^\perp$ are paths which pass through each vertex an even amount of times. These are closed loops so $B_1^\perp = Z_1^*$, hence $Z_1^\perp = B_1^*$, so $B_1^\perp / Z_1^\perp = Z_1^* / B_1^*$. Applying this to a uniform square lattice on a torus produces the $k = 2$ toric code. This is studied by mathematicians under the name quantum double models.

If qudits are placed on vertices and stabilizers represent faces, the number of encoded logical qudits is purely topological and determined solely by the genus (Euler's characteristic) of the manifold. This also follows immediately from computing $\dim H_p(M, \mathbb{Z}_2)$ using standard techniques of algebraic topology. All possible dimensions were immediately classified and discussed for various manifolds and cellulations (e.g. [46, 47, 48]). This idea has also been extended to 1-skeletons of general graphs with no underlying manifold, in particular, with tilings of hyperbolic space [49, 50, 51, 52]. Reference [53] connects this to the original C^*-algebra approach. Note that the problem of constructing good codes is now that of finding embeddings of graphs in such a way that both the embedded graph and its dual have a large minimum distances while minimizing the number of edges. Since it is difficult to physically implement a torus, many codes considered experimentally viable have introduced boundaries [45] either using a manifold with boundary from the start or puncturing the manifold to introduce holes [54] (equivalently removing faces from the cellulation), increasing the genus and hence the number of encoded qudits.

Let us introduce the specific examples of topological codes, the rotated surface and color codes, we will need in the following chapters. The term "surface code", or planar code, is often used in the literature as a blanket term to cover all quantum codes constructed by embedding a graph on the surface of a manifold. Here we reserve the term to

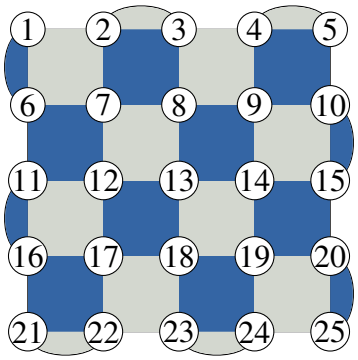
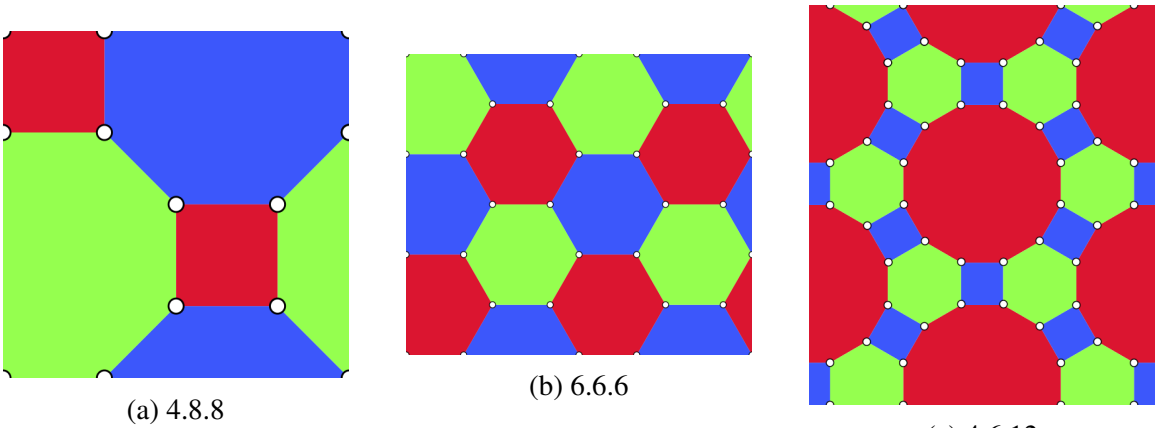


Figure 1.2: The $[[25, 1, 5]]$ rotated surface code used in this work: X stabilizers are given by grey (light) faces and Z stabilizers by blue (dark) faces.

explicitly refer to the so-called “rotated" surface codes of [55]. The most popular topological code is the previously mentioned toric code, which is related to the surface codes here by introducing boundaries and applying the medial-axis transform.

The particular configuration of (rotated) surface codes we are interested here is the so-called surface-17 family. Start with a $(2d-1) \times (2d-1)$ lattice with alternating faces colored in a standard checkerboard pattern, rotate by $\pi/4$, and remove the four corner vertices (e.g. Figure 1.2). Qubits (over $\mathbb{F}_2$) are assigned to each vertex. This is in contrast to the previously mentioned toric code which assigns a qubit to each edge. The vertices involved in each face give vectors interpreted as a or b in the stabilizer formalism, where one color is arbitrarily assigned to X and the other to Z. (For example, the vector with $a_1 = a_2 = a_6 = a_7 = 1$ and all the rest of the 25 elements zero corresponds to the first face in Figure 1.2.) Here, putting the qubits on the vertices of the lattice requires a boundary map $C_2 \to C_0$. The vertex/edge switch and checkerboard pattern stems from an attempt to view both the cellulation and its dual on the same graph.

The number of stabilizer generators, $n-k$, is the number of faces, where n is the number of vertices. This model therefore supports $k = 1$ logical qubit. Any path of vertices stretching from the left boundary to the right boundary overlaps each face an even number of times (commutes with all of the stabilizers) yet cannot be written as a product of faces, making it a (physical) logical operator. The logical operator of the other type stretches from



(a) 4.8.8 (b) 6.6.6 (c) 4.6.12

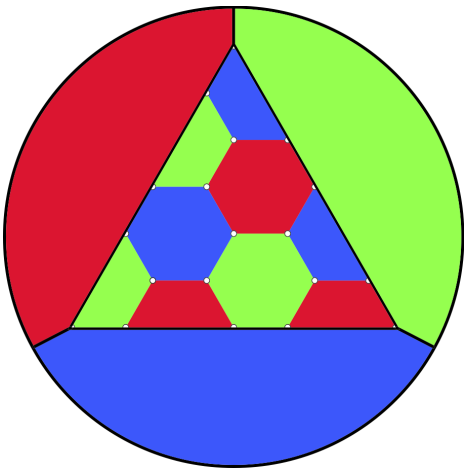
Figure 1.3: The 3-valent, 3-colorable tilings of the 2-sphere.

top to bottom. The minimum weight logical operator, and hence the minimum distance of the code, is d on a $d \times d$ lattice, resulting in a $[[d^2, 1, d]]$ QECC.

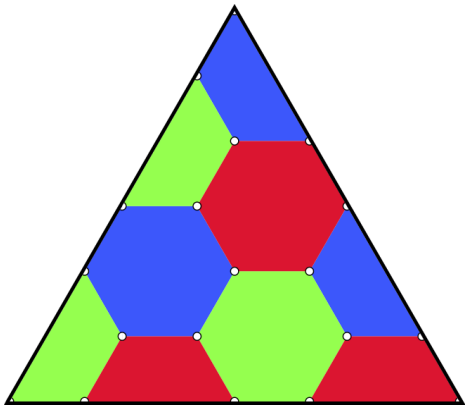
By color codes, we specifically mean the triangular color code families with $k = 1$. The original definition [47] allows for generalizations to other polynomials with $3m$ sides and $k = m$ logical qubits. To the author's knowledge, these have not been explored. Instead of a square lattice on the torus or a plane (4-valent, 2-colorable), the color codes are defined from 3-valent, 3-colorable tilings of the 2-sphere. Recall that in the standard vertex configuration notation for tilings, $a.b.c$ means every vertex touches an a-gon, a b-gon, and a c-gon. The relevant tilings are the 4.8.8 [56], 6.6.6 [57], and 4.6.12 [58] seen in Figure 1.3 and their corresponding color codes are distinguished by the same notation. The 4.6.12 color codes are not considered useful in the literature because a weight-12 stabilizer[8] is more difficult to handle experimentally. Taking the theorist's perspective, we ignore this and some codes in this work may have considerably larger weight stabilizers; yet, we will also ignore the 4.6.12 color code family for the rest of this work. See [59] for a discussion of its properties.

Fix a tiling and let $|V|$ be the number of vertices, $|E|$ the number of edges, and $|F|$ the faces; then $|V| = 2(|F| - 2)$ and $|E| = 3|V|/2$. Placing a qubit on each vertex and

[8]The weight of a stabilizer $\eta^c X(a)Z(b) \in \mathcal{S}$ is the Hamming weight of the a and b over the quadratic extensions or the symplectic weight over the prime subfield.



(a) The 6.6.6 tilling of the 2-sphere.



(b) The 6.6.6 $d = 5$ (triangular) color code created by puncturing (a).

Figure 1.4: The triangular color codes are constructed by puncturing a tiling of the 2-sphere.

associating both an X and a Z stabilizer with every face produces a $k = 0$ code. Puncturing the sphere by deleting one of the faces gives $k = 1$ (Figure 1.4). By a similar argument to the surface codes, logical operators are supported on any of the three edges of the triangle. The minimum distance of the code is hence the number of vertices along any edge, and the size of the tiling is classified by d (Figure 1.5).

The reader is referred to the various works of Kubica (e.g. [60, 61]) for a good presentation of this family, including its generalization to ℓ-spheres using higher-dimensional analogues of faces. Anderson used the medial transform to perform a thorough analysis of possible valencies and tiling properties required to support a topological stabilizer code and showed that if two stabilizers are going to be defined on a single face the lattice is necessarily 3-valent [51].

1.4 Quantum Circuit Model

At this point, we hope the reader has grasped the mathematical foundations of QEC. This level of understanding is more than enough to read the majority of the QEC literature with one glaring exception: much of QEC is typically described in pictures akin to circuit diagrams in electrical engineering. This is called the circuit model of quantum computation

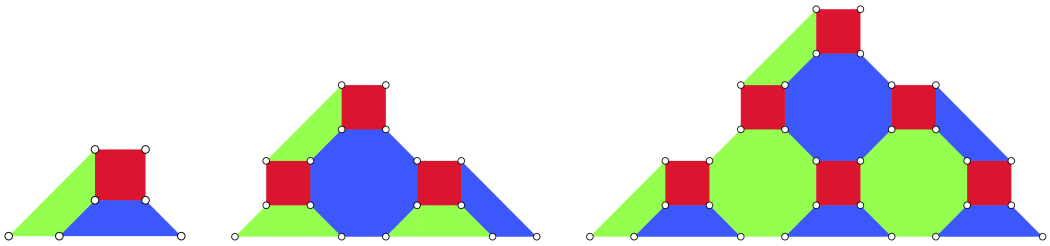


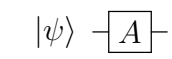
Figure 1.5: The 4.8.8 $d = 3, 5$, and 7 color codes. The tiling is often distorted so as to make smooth edges as in Figure 1.4b.

[62] and is invaluable to the field. Similar to the early day of classical computers, quantum computing must still be described at this elementary level. A large portion of the quantum computing literature is devoted to translating quantum routines to an optimized sequence of logic gates. Common techniques for this include machine learning, algebraic number theory (e.g. [63]), and category theory (e.g. [64]). Only in the past three to five years have the notions of what a high-level quantum programming language would look started to be developed. This is compounded by the fact that there is not a universal quantum assembly language; some physical systems support different native logic gates than others. The quantum circuit model is analogous to bytecode in a classical computer; all programs may be expressed in it and as long as an algorithm is compiled down to it the job of the theorist is done. (Machine code would be analogous to the actual (e.g.) laser pulses used to implement each gate.)

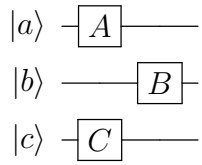
We will not have the opportunity to explicitly use the circuit model this in this work, but we will make several references to it throughout. In particular, the techniques presented in the next two chapters do not yet include all of the information required or provided by the circuit model, and it is an open problem to extend this work to something experimentalists can use in practice. This description will also demonstrate the main reason QEC is much more difficult than its classical counterpart.

For the moment, we will align with the literature and stick to $\mathbb{F}_2$. As before, a quantum

operation on n qubits is a unitary operator on $\mathbb{C}^{2^n}$. All unitary operations have inverses, so quantum circuits are reversible in the sense that given their output it is possible to uniquely reconstruct their input, which is not true of classical logical gates. A single qubit is described by a line (wire),



Time on this diagram is read left to right: a qubit is initially in an arbitrary linear combination of basis states $|\psi\rangle$ and then is acted on by a matrix A such that the output at the end of the wire is $A|\psi\rangle$. Multiple bits are stacked on top of each other and are read from top to bottom:



is equivalent to $(A \otimes B \otimes C)|abc\rangle = A|a\rangle \otimes B|b\rangle \otimes C|c\rangle$. Imaginary vertical slices in the diagram represent time steps. The diagram above takes two steps to implement as written but there's nothing stopping us from sliding B down the wire to implement it in one time step as indicated by the math. A vast literature exists on circuit optimization. If time was the only metric we cared about, we easily could solve this with some circuit identities and a directed graph.

The simplest quantum gates are those acting on a single qubit. In analogy to the discussion above, we have the so-called Pauli gates[9] $X := X(1)$, $Z := Z(1)$, and $Y = iXZ$, the Hadamard gate H, the phase gate P (often denoted S), and the T-gate or $\pi/8$-gate. Representations of these taken with respect to the natural ordering of the basis, $\{|0\rangle, |1\rangle\}$, are

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad , \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad , \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{1.31}$$

[9]The symbols X, Z, and Y will be used throughout this work from here on out without the explicit notational dependence on $\mathbb{F}_q$.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad , \quad P = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad , \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} = e^{i\pi/8} \begin{pmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{pmatrix}. \tag{1.32}$$

Defining $|+\rangle \coloneqq (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle \coloneqq (|0\rangle - |1\rangle)/\sqrt{2}$, we have

$$X|0\rangle = |1\rangle \quad , \quad X|1\rangle = |0\rangle \quad , \quad X|+\rangle = |+\rangle \quad , \quad X|-\rangle = -|-\rangle, \tag{1.33}$$

$$Z|0\rangle = |0\rangle \quad , \quad Z|1\rangle = -|1\rangle \quad , \quad Z|+\rangle = |-\rangle \quad , \quad Z|-\rangle = |+\rangle, \tag{1.34}$$

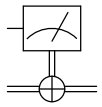
$$H|0\rangle = |+\rangle \quad , \quad H|1\rangle = |-\rangle \quad , \quad H|+\rangle = |0\rangle \quad , \quad H|-\rangle = |1\rangle. \tag{1.35}$$

It follows that the so-called computational basis $\{|0\rangle, |1\rangle\}$ are eigenvectors of Z with corresponding eigenvalues ± 1, respectively, and $\{|+\rangle, |-\rangle\}$ are eigenvectors of X with corresponding eigenvalues ± 1, respectively, and H allows one to switch between bases. The X gate is commonly referred to as the bit-flip operator as it is the quantum analog of the classical `NOT` gate. The Z gate is commonly referred to as the phase-flip operator. Note that $Z = P^2 = T^4$. The term Hadamard stems from a more general construction in pure math, while for $q > 2$ this matrix is referred to as the discrete Fourier transform.

The measurement process is represented by the meter symbol,

There is no wire extending to the right of a meter symbol due to the fact that the qubit has been destroyed by the measurement process. Technically, one must store the result of the measurement, most likely on a classical computer. This is represented by a meter symbol extending to a target symbol on a double wire representing the one bit of classical

information and the collapsed state (Definition 1.2.1).



Common two-qubit gates include the controlled-X gate, C_X, also called controlled-NOT or CNOT, the controlled-Z gate, C_Z, and, in general, any controlled 2×2 unitary gate U, C_U,

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad , \quad C_Z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad , \quad C_U = \begin{pmatrix} I_{2\times 2} & 0_{2\times 2} \\ 0_{2\times 2} & U \end{pmatrix}. \tag{1.36}$$

Another common two-qubit gate is the SWAP gate,

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

but one may easily show that this is simply a combination of three CNOT gates, so we will not utilize this here.

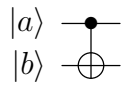
Following the operator-theoretic discussion of pure and mixed states, a two-qubit state is separable if it can be decomposed into a tensor product of two states. States which are not separable are mixed (with respect to individual qubits) and are said to be entangled. For example, $|00\rangle + |10\rangle = (|0\rangle + |1\rangle) \otimes |0\rangle$ is separable while $|00\rangle + |11\rangle$ is not. Now consider the action of a CNOT gate on a system whose first qubit is in state $|0\rangle + |1\rangle$ and

whose second qubit is in state $|0\rangle$:

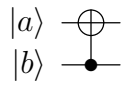
$$C_X((|0\rangle + |1\rangle) \otimes |0\rangle) = C_X(|00\rangle + |10\rangle) = |00\rangle + |11\rangle. \tag{1.37}$$

Hence, CNOT gates generate mixed states and may be used to create entanglement. This is an extremely important point as entanglement is a unique feature of quantum mechanics not found in classical mechanics, and is one of the main reasons for quantum supremacy over classical computing. When one sees a CNOT gate in a circuit diagram, it is useful to interpret it as creating a coupling between the two qubits.

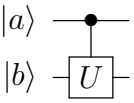
Symbolically, $C_X(|ab\rangle)$ is



The $\bullet$ is on the first qubit, called the control, and the $\oplus$ is on the second qubit, called the target. As before in equation (1.37), the target qubit will flip if and only if the control qubit is in state $|1\rangle$. Note that



represents $C_X(|ba\rangle)$. A general controlled gate is represented by



with $\bullet$ again representing the control qubit. Here, the operation U is applied to qubit $|b\rangle$ if and only if $|a\rangle$ is in the state $|1\rangle$.

There are common three- and higher multi-qubit gates, but we do not need them here for the following reason. In classical computation, the AND, OR, and NOT gates are universal in the sense that any boolean function can be expressed entirely in terms of only these three gates. On the quantum side, it is a bit trickier and there are a number of equivalent choices of sets of results [65], but we have the following result.

Theorem 1.4.1

The set of gates H, P, T, and CNOT *are universal.*

According to this, we can decompose any arbitrary n-qubit gate into a combination of H, P, $\pi/8$, and CNOT gates. Unfortunately, this process often requires an infinite sequence of gates. The Solovay-Kitaev algorithm provides a constructive way to approximate such a gate to arbitrary precision. Formally, (see [66] for a pedagogical review).

Definition 1.4.2

A set of unitary gates G is quantum computationally universal if for any n, any unitary operation $U \in SU(2^n)$ can be approximated to arbitrary accuracy ε in some norm $||\cdot||$ by a product of gates in G, i.e., $\forall \varepsilon > 0, \exists V = V_1 V_2 \dots V_{\eta(\varepsilon)}$ where each $V_i \in G$ such that $||V - U|| < \varepsilon$.

Theorem 1.4.3 (Solovay-Kitaev [66])

Let G be the group defined by a set of universal gates for $SU(n)$, and let a desired accuracy $\varepsilon > 0$ be given. Then there is a constant c such that for any $U \in SU(n)$ there exists a finite sequence of gates from G of length $\mathcal{O}(\log^c(1/\varepsilon))$ and such that the distance between U and S is less than ε.

It is interesting to note that the proof of this relies on cyclotomic polynomials over $\mathbb{Q}$. Work in this direction is still ongoing today in several directions. See [67] for a thorough and rigorous study of universality from a pure math perspective.

This formalization makes the significant assumption that the quantum system is closed, and henceforth ignores all interactions with the environment. For now, the model further assumes that every operation is completed without error. It is worthwhile and significant to mention at this point another significant theorem.

Theorem 1.4.4 (Gottesman-Knill [68])

Any quantum computer performing only: a) I, X, Y, Z, H, P, and CNOT*, b) measurements, and c) gates conditioned on classical bits, which may be the results of earlier*

measurements, can be perfectly simulated in polynomial time on a probabilistic classical computer.

It follows that we need more complicated operators than $\mathcal{P}_n$ for quantum computation, but since computation is performed on encoded information based on $\mathcal{P}_n$ we need these operations to be elements of $\mathcal{N}_{\mathcal{U}(n)}(\mathcal{P}_n)$, the normalizer of the Heisenberg group in the unitaries. This is called the Clifford group, Cliff_2, and in general we have the Clifford hierarchy.

Definition 1.4.5 (Clifford Hierarchy [69])

Let $\mathrm{Cliff}_1 = \mathcal{P}_n$. *Then the Clifford hierarchy is defined by*

$$\mathrm{Cliff}_k = \{U \mid U^* \mathrm{Cliff}_1 U \subset \mathrm{Cliff}_{k-1}, \forall P \in \mathcal{P}_n\}.$$

The sets $\mathrm{Cliff}_{k\geq 3}$ are not closed under multiplication and are therefore not groups, but the set of diagonal operators in each is a group. The structure of Cliff_k is unknown but there are several notable results in this direction, mainly for the diagonal operators [70, 71, 72, 73, 74, 75, 76]. The Clifford group is notoriously difficult to analyze and is usually only considered up to elements of $U(1)$ [77, 78, 79, 80]. Up to phases, the Clifford group (for qubits) has cardinality

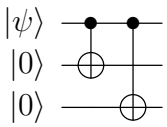
$$|\mathrm{Cliff}_2/U(1)| = 2^{n^2+2n} \prod_{i=1}^{n} (4^i - 1)$$

and generators H and P on any single qubit and `CNOT` between any pairs of qubits. It is known that any universal gate set must include at least one element outside the Clifford group; in the set above, $T \in \mathrm{Cliff}_3 \backslash \mathrm{Cliff}_2$.

1.4.1 The Shor Code

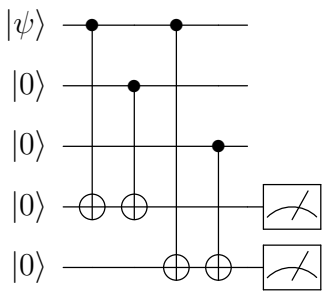
It is worthwhile to derive the first (binary) QECC, the Shor code, in the quantum circuit language as would be described in the physics literature. One of the simplest classical codes is called the repetition code. Suppose, for example, that $n = 3$, then the code is

defined by $0 \mapsto 000$ and $1 \mapsto 111$ such that $010100 \mapsto 000111000111000000$. To decode a codeword, we read it from left to right in batches of three bits at a time and decode each batch by majority vote. We see immediately that this can only correct one error since flipping any two bits will cause the majority vote to decode incorrectly. Reproducing this on the quantum side is a bit tricker. Suppose we wish to encode the unknown quantum state $|\psi\rangle$ using the repetition code. The No-Cloning theorem 1.2.2 prevents us from making a copy of $|\psi\rangle$, so we cannot create the state $|\psi\psi\psi\rangle$. Instead, expanding in the computational basis as $|\psi\rangle = a|0\rangle + b|1\rangle$, we can proceed as before with $|0\rangle \mapsto |000\rangle$ and $|1\rangle \mapsto |111\rangle$ such that $|\psi\rangle \mapsto a|000\rangle + b|111\rangle$. To build this state we must introduce two new physical qubits to the system and couple them to $|\psi\rangle$ using a `CNOT` gate with $|\psi\rangle$ as the control. Recalling from above that a `CNOT` gate flips the target if and only if the control is $|1\rangle$, the $a|0\rangle$ term and the new qubit $|0\rangle$ will produce $a|00\rangle$ and the $b|1\rangle$ term and the new qubit $|0\rangle$ will produce $b|11\rangle$. Repeating this with a second new qubit gives the desired result. This is called the three-qubit bit-flip code and is summarized by the circuit diagram below.



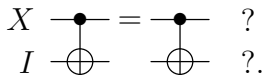
To decode the state $a|000\rangle + b|111\rangle$ we must be a little more clever than majority vote. Measuring one of the three qubits will collapse it to $|0\rangle$ with probability $|a|^2$ or $|1\rangle$ with probability $|b|^2$, thus destroying the superposition and the information stored in the state as a result. To get around this, we introduce more qubits into the system, couple them with the state which we wish to decode, measure (and hence destroy) these new qubits, and use the results to infer properties of the original state. As long as this coupling and corresponding measurement are done correctly, no information about the values of a and b is obtained and the original state remains intact. To separate the new qubits from the old, the qubits in the original state are called data qubits and the temporary qubits are called the ancilla, which is Latin for “an auxiliary or accessory". The decoding circuit for the three qubit bit-flip code

is given by the following circuit diagram, with the top three qubits being the data qubits from the encoding circuit above and the bottom two qubits the ancilla qubits.

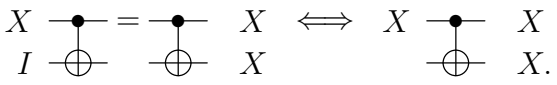


(1.38)

To see why this circuit works, we need to investigate the effect an X error has on a CNOT gate:



Writing this as an equation and moving the right CNOT gate to the left hand side gives $C_X^*(X \otimes I)C_X$, which can be directly multiplied to give

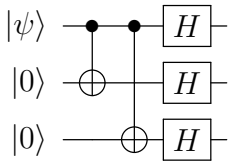


(1.39)

Suppose now that an X error occurs on exactly one qubit. Looking back at (1.38), if an X error occurs on the first qubit, $|\psi\rangle$, or the second qubit, it will propagate through the CNOTs and the first ancilla will flip from a $|0\rangle$ to a $|1\rangle$. Likewise, if an X error occurs on the first qubit or the third qubit, it will propagate through the CNOTs and the second ancilla will flip from a $|0\rangle$ to a $|1\rangle$. Recalling that $|0\rangle$ and $|1\rangle$ are eigenvectors of the Z operator with eigenvalues $+1$ and -1, respectively, measuring the first ancilla with respect to the Z basis tells us that if the outcome is $+1$ then the first ancilla was not flipped to a $|1\rangle$ and hence neither the first or second qubits suffered an X error. On the other hand, if the outcome is -1, the first ancilla was flipped implying an error happened to either the first or second qubits. Likewise, the outcome of the measurement on the second ancilla tells us if an error happened on the first or third qubits. If both measurement outcomes

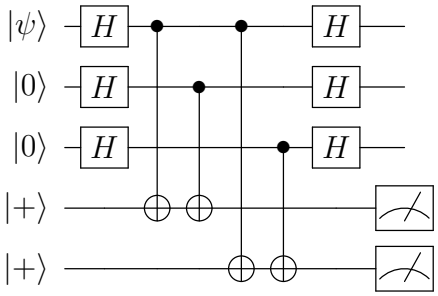
are $+1$, we can deduce that none of the qubits suffered an X error. If the first outcome is $+1$ and the second -1, we can deduce that the third qubit suffered the X error. If the first outcome is -1 and the second $+1$, we can deduce that the second qubit suffered the X error. Finally, if both measurement outcomes are -1, we can deduce that the first qubit suffered the X error. Once we know which qubit, if any, has an error, we correct the error by simply applying a X^* to the affected qubit. This ± 1 pattern of measurement outcomes is the syndrome defined in the sections above. Note that as with the classical repetition code, the three-qubit bit-flip code can only correct one error since, for example, if the first and second or first and third qubits both suffered an X error, the corresponding ancilla would be flipped twice ($|0\rangle \mapsto |1\rangle \mapsto |0\rangle$) and would not be detected by the measurement.

New to the quantum realm, we must also correct against Z errors (phase flips). But equation (1.34) says that a phase flip is a bit flip in the $|\pm\rangle$ basis. Therefore, we can mimic the bit-flip code via $a|0\rangle + b|1\rangle \mapsto a|{+}{+}{+}\rangle + b|{-}{-}{-}\rangle$ and then detecting errors by measuring in the X basis since $|\pm\rangle$ are eigenvectors of X with eigenvalues ± 1, respectively. The encoding circuit for the three-qubit phase-flip code is given by

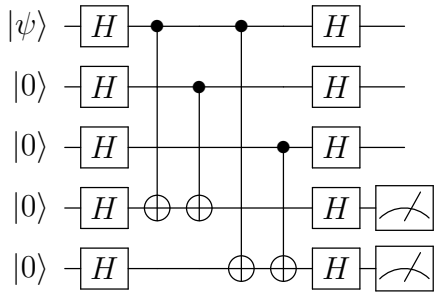


where we have used the fact that the Hadamard gate is a basis transformation. Likewise, the syndrome extraction circuit first changes basis, checks for bit flips, then changes back,

which follow from the previous equations and the fact that H is self-adjoint:[10]



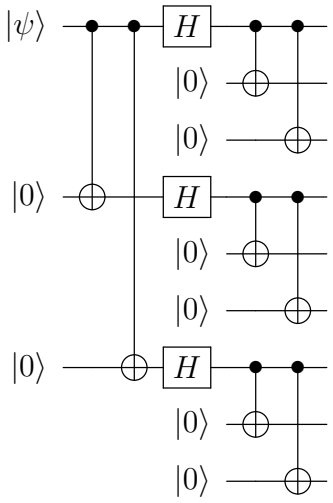
Note that these measurements are now in the X basis instead of the Z basis as before. It is often preferred to always measure in the Z basis due to experimental considerations, in which case our circuit becomes



The Shor code is a concatenation (combination) of the three-qubit bit-flip and phase-flip codes: first encode to protect against bit flips then encode the result to protect against

[10]Here we are implicitly assuming that the state has already been encoded into the system before running this circuit.

phase flips. The encoding circuit is simply:



(1.40)

This is a $[[9, 1, 3]]$ CSS code with X and Z stabilizer matrices

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

respectively, and (physical) logical operators $\overline{X}, \overline{Z}$ given by the all-ones vector, commonly denoted $\overline{X} = X^{\otimes 9}, \overline{Z} = Z^{\otimes 9}$ in the literature. Both of these can be derived by reasoning about the circuit diagram. For example, consider deriving the logical operators for the bit-flip code. Denote operations on the logical level ($\mathcal{P}_k$) with a subscript L for increased clarify. If we apply the X operator to the generic state $|\psi\rangle = a|0\rangle + b|1\rangle$ before encoding, we get $X|\psi\rangle = a|1\rangle + b|0\rangle$. We wish to repeat this logic such that apply X after encoding results in $X_L|\psi\rangle = a|111\rangle + b|000\rangle$, but since $|000\rangle = |0\rangle \otimes |0\rangle \otimes |0\rangle$ we get $X_L = X \otimes X \otimes X$, $|0\rangle_L = |000\rangle$, and $|1\rangle_L = |111\rangle$. As we see, the stabilizer formalism is more systematic and constructive; however, in the end, one does need to construct a circuit

diagram like (1.40) to actually build the system. It is interesting to note that the Shor code may be obtained as a cellulation of $\mathbb{R}P^2$ [46].

This simple example demonstrates that we need to relax our previous notation. It is common in the literature to see these stabilizers denoted as

$$\mathcal{S} = \langle XXXXXXIII, IIIXXXXXX,$$
$$ZZIIIIIII, IZZIIIIII, IIIZZIIII, IIIIZZIII, IIIIIIZZI, IIIIIIIZZ\rangle$$

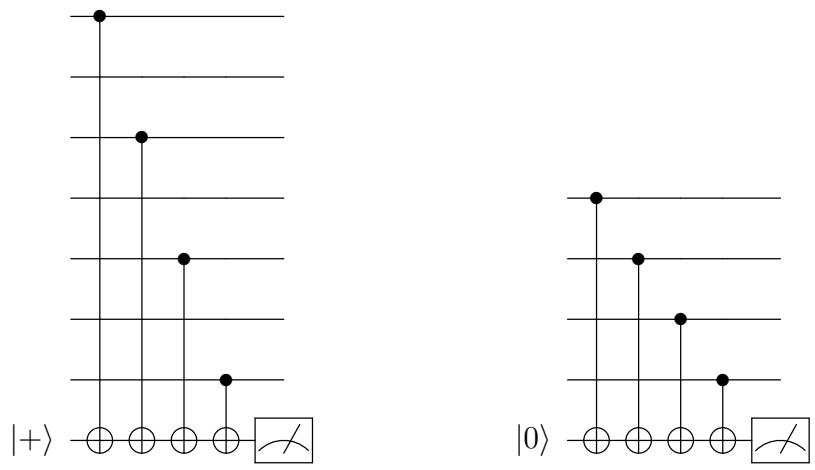
or $\mathcal{S} = \langle X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9, Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9\rangle$, the latter of which severely shortens expressions. As before, terms such as X_1Z_1 will be written Y_1. In general, qudit systems over $\mathbb{F}_q$ may utilize exponents such as X^aZ^b to denote $X(a)Z(b)$. This is the notation of choice in the physics literature and is immensely convenient, but then operations on these symbols must be explicitly defined to match the corresponding operations over $\mathbb{F}_{q^2}$, at which point one might as well simply just use $\mathbb{F}_{q^2}$. We will use the physics notation to reference stabilizers but the math notation when we wish to manipulate them, for clarity.

1.4.2 Fault-Tolerance And Transversality

While the concept of fault tolerance is easily understood and explained, even a semi-thorough discussion would take several pages and is beyond the intended scope of this work. At the same time, it is absolutely crucial to the theory of quantum computation and QEC. In brief, an error occurring during the application of a single-qudit gate $U|\psi\rangle$ only affects that qudit and may therefore be modeled by $EU|\psi\rangle$, where E is an appropriate error. However, an error occurring during a two- (or more) qudit gate can affect multiple qudits. Not only that, as we have seen with `CNOT` gates, a single-qudit error can multiply by propagating through a gate.

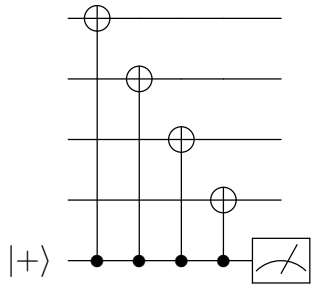
To emphasize this, let's consider the circuit below to measure the weight-four stabilizer

$Z_1Z_3Z_5Z_7$:

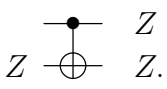


(1.41)

Since qubits 2, 4, and 6 are not involved in the circuit on the left, it suffices to ignore them as on the right. We will implicitly assume all diagrams are simplified in this manner going forward. There are no symbols $|\psi\rangle$ on the left of the wires denoting that we simply apply the gates to whatever configuration the qubits are currently in. The top qubits not being measured are the data and the measured qubit the ancilla. Note that the data is unaffected by this circuit but the ancilla is destroyed by the measurement. An X error on any of the data qubits will propagate down through the `CNOT` symbol and onto the ancilla (Equation (1.39)), flipping $|0\rangle$ to $|1\rangle$ (Equation (1.33)). The (Z) measurement will return a -1 eigenvalue (Equation (1.34)), altering us to an error. To measure $X_1X_3X_5X_7$ we use the circuit



and apply

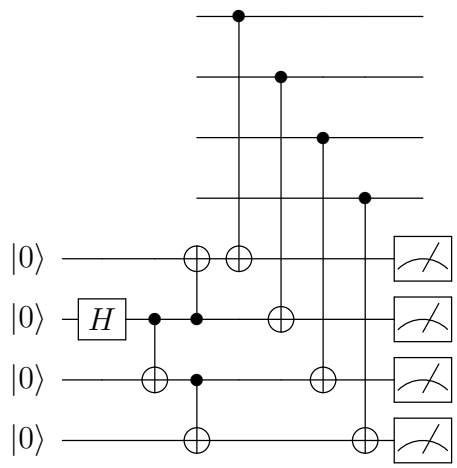


(1.42)

In this way we have “measured" an element of the form $\eta^c X(a)Z(b)$ and extracted the syndrome bit corresponding to the operator. For mixed cases with both a and b nonzero, a combination of CNOT gates of with targets and controls on the ancilla are used appropriately. Syndrome extraction in these cases become messier as errors propagating down from the data can now potentially propagate back up to other data qubits.

This model assumes everything works as intended. What can go wrong? Turns out, everything. First, if a Z error occurs on the ancilla of (1.41) it will propagate up onto all four, three, two, one, or no data qubits depending when it occurs (Equation (1.42)). If it occurs before the first CNOT, it will cause the weight-four error $Z_1Z_3Z_5Z_7$ on the data, but this is fine because this is a stabilizer. If it occurs after the last CNOT, $Z|0\rangle = |0\rangle$ so the error will be ignored by the measurement and no harm has been done. The problematic cases are when it occurs between the first and second, second and third, or third and fourth CNOTs. One possible fix is to switch to the circuit[11]

(1.43)

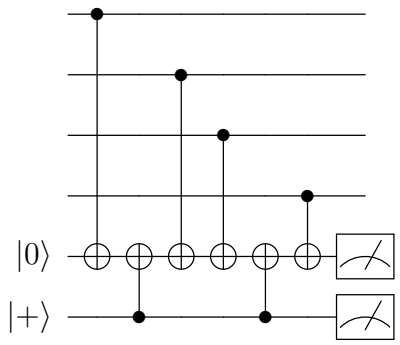


which handles the propagation at the expense of requiring extra resources. This is called Shor-style syndrome extraction. Such propagating errors from the ancilla are called hook errors, and the process of “controlling" errors in the implementation is loosely what is referred to as fault tolerance. Diagram (1.43) is called a fault-tolerant measurement gadget

[11] A practical circuit would also include a “verification" step, which we ignore here for simplicity of discussion.

for (1.41). See [81] for a general discussion of syndrome measurement techniques.

Chao and Reichardt proposed the following elegant modification to (1.41)



where the top measurement is in the Z basis and the bottom measurement is in the X basis [82, 83]. The second ancilla is called a flag and will measure -1 when a hook error occurs during the CNOTs. If an error is detected, the resulting combination of the two measurements is enough to determine which CNOT had the error and hence how to correct it. Of course this only works in rare circumstances, and it is an open problem to determine for which QECCs this technique is applicable. It is known that first order 15-qubit quantum Reed-Muller (QRM) codes (the Steane code), quantum cyclic codes [84], and the quantum Hamming codes [82] are able to use flag syndrome extraction. For some codes, permuting the order of the CNOTs in the circuit enables the use of flags. It is an open problem to determine a permutation, if one exists, for which this works. It is likely this is nondeterministic polynomial time (NP) in the same way finding permutations of linear codes to minimize the metrics of the next chapter is NP-Complete [85]. The optimal permutation for the QRM and Hamming codes are determined by mapping the columns of the stabilizer matrix to polynomials in $\mathbb{F}_2[x]$, mapping the polynomials to roots in an extension field, then considering geometric sums [82]. It is probable the decoder studied in this work can include the information provided by the flag, but other circuit-level information likely requires significant modification. We will return to this in Chapter 4.

If there are a lot of stabilizers, a single round of error correction will take a lot of time

and the laws of physics say errors may spontaneously occur even on idling qudits. Sometimes it is possible to measure stabilizers in parallel. There is no general theory for this. If the stabilizers are high-weight, which is unfortunately often considered to be greater than weight four, the propagation patterns from the CNOTs become difficult to reason about and the resources required to implement the second measurement circuit style become prohibitive as the more qubits are required for syndrome extraction the less are available for the code itself. Of course errors may occur during the syndrome extraction circuit which cancel previous errors, resulting in an incorrect error correction. Or they may occur undetected until the next round of measurements, which implies we must do this process continuously, ignoring the fact that it was the syndrome extraction process which caused the error in the first place.

Thinking back to the mathematical definition of measurement, Definition 1.2.1, the result of the measurement is actually probabilistic. There is a chance that given a combination of experimental error and the laws of physics, performing, for example, a Z measurement on $|0\rangle$ will incorrectly return the eigenvalue -1. To handle this, every stabilizer of a $[[n, k, d]]_q$ code is often measured $\mathcal{O}(d)$ times and the majority vote of the measurement outcomes is taken to be true result. But this gives more, for example, CNOTs, which provide more opportunities for everything to go wrong.

The dominating source of circuit errors in current state-of-the-art systems is the failure to correctly experimentally implement a gate. Gate errors are problematic because they can affect every qubit which it interacts with. Suppose one supplements the Clifford group with a four-qubit gate in Cliff_3 to make a universal gate set. Anytime this gate fails, a single failure can cause four errors on the data qubits. This is not “fault-tolerant". It would be nice if the four-qubit gate can be decomposed into the tensor product of single qubit gates since then a gate error would only affect a single data qubit. Gates of this form are an example of transversal gates and are automatically fault-tolerant.

Definition 1.4.6 (Loose Version)

Consider information encoded in an error-correcting code which is able to correct $t = \lfloor (d-1) \rfloor / 2$ *errors. Then a circuit is fault tolerant if whenever the number of input errors plus the number of circuit errors is no more than* t*, the output of the circuit has no more than* t *errors.*

The idea behind fault-tolerance is that the left-over errors can be corrected by the next round of error correction as long as there are not too many of them.

As previously mentioned, gates acting on encoded data must be elements of the normalizer of the stabilizer group so it does not destroy the encoding. A code is said to support a gate if it commutes with the code projector. Unfortunately, the Eastin-Knill theorem [86] says that no QECC can support a transversal, universal gate set. A significant amount of effort has gone into studying transversal gates. The general idea to QECC design is usually to get as many transversal gates as possible, typically all but one, and then determine a procedure to implement the missing gate(s). There are many possible QECCs with great properties but the supported gate set is either unknown or considered poor. This is another reason QEC is more difficult than classical coding theory.

A few general statements can be made, for example, [17, 87]

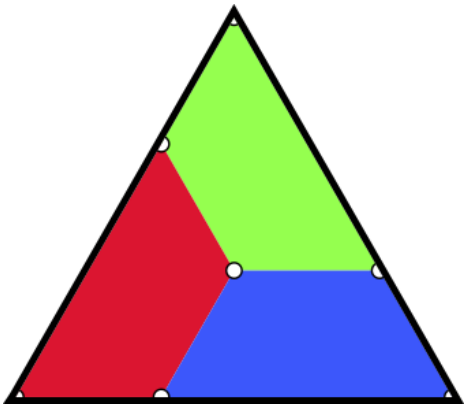
(i) all CSS codes support a transversal `CNOT`

(ii) all CSS($\mathcal{C}$) codes support a transversal H

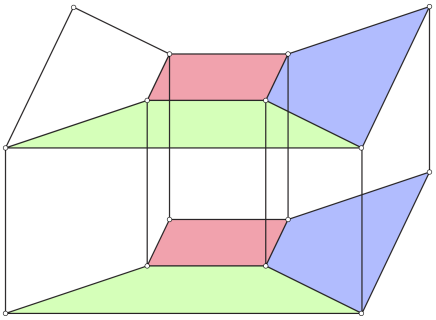
(iii) all CSS($\mathcal{C}$) codes whose parity-check matrix has row weights $0 \pmod 4$ support a transversal P.

Further conditions are known for gates we have not discussed in this work. Both necessary and sufficient conditions for CSS codes supporting a T gate were finally settled in [88, 89]. In particular, [88] showed that CSS codes may even be optimal for obtaining transversal T gates compared to general non-degenerate stabilizer codes $X(a)Z(b)$.

There are a couple of ways to get around Eastin-Knill. One way is to use multiple QECCs with complementary transversal gates and then passing the encoded information



(a) The Steane code as a $d = 3$ color code.



(b) The QRM code contains two copies of the Steane code plus an extra qubit.

Figure 1.6: Common codes viewed as color codes.

between them as needed during the algorithm. This is complicated and must be done with care so as not to expose the data to errors in the process of switching. The canonical example of this is the Steane code with transversal gates H, P, and CNOT and the $[[15, 1, 3]]$ QRM code with transversal T [90]. The Steane code is actually equivalent to the 4.8.8 and 6.6.6 $d = 3$ color codes (Figure 1.6a), and the QRM code can be seen as two copies of the Steane code plus an extra qubit (Figure 1.6b). Data is encoded into the Steane code and then the QRM code is constructed and deconstructed whenever a T gate appears in the quantum circuit. This is a subsystem code of Equation (1.15) where the Hilbert space of the QRM code is decomposed into the tensor product of the Steane code with the rest of the space. Errors may accumulate in the unused part of the Hilbert space while in the Steane code, and the conversion process projects the remaining degrees of freedom onto the $+1$-eigenspace of the rest of the stabilizers to form the QRM. This is called gauge fixing with the unused degrees of freedom being referred to as gauges. Bombín used the gauge fixing idea to stack color codes to make a 3D color code supporting transversal T [91, 92, 93, 94]. Similar ideas appear in [95, 96, 97]. We will return to this example in Chapter 4.

Another way to circumvent Eastin-Knill is to have a part of a the computer mass producing eigenspaces of the missing gates which are then “injected" or teleported into the running circuit as necessary. Current proposals to do this can be orders of magnitude more

costly than if the gate was transversal, and this is expected to dominate the resources of near-term devices pushing many useful algorithms out until quantum computers with millions of quantum bits are developed. This is one of the biggest challenges in QEC, but we will not go into it further. Select references which may be useful to the interested reader are [98, 99, 100, 101, 102, 103, 104, 105]

The general procedure to implement a quantum algorithm on a future quantum computer is roughly given by the following steps:

(i) pick a universal gate set and QECC

(ii) decompose the algorithm into an optimized circuit using elements of (i)

(iii) replace the logical operations with physical implementations in the encoded space

(iv) replace circuit elements with appropriate fault tolerant versions.

At the time of writing, (iv) is also the least understood. There is no currently known unifying principal for fault tolerance, and it is unclear how to rigorously discuss this mathematically. There is a common definition of fault tolerance, but [106] shows that it is sufficient but not necessary. It was conjectured in [107] that "fault-tolerant logical gates can always be expressed as arising from monodromies of an appropriate fibre bundle with a flat projective connection". Beyond logical arguments for a handful of codes and a number of works on general ideas, little is known about this topic. Landahl et al. discuss the assumptions that all fault-tolerant protocols appear to make [108].

1.5 Computer Simulations Of Codes

To "simulate an error-correcting code" we mean to determine how well the code protects the information it stores against errors. This of course depends on exactly what the errors are and how they attack the system. For classical codes these are generally bit flips applied independently and identically distributed (i.i.d.) to each bit or to a collection of bits with

local support (burst errors). For quantum codes, there are many options and an explicit error channel must be specified. This also highly depends on the chosen decoder. A decoder which maps all inputs to the zero vector (or identity element) does not do as good a job of protecting the information as does one which actively attempts to compute the most likely input.

Irrespective of all these elements, the idea of a simulation is simple: compute the probability that the decoder will "fail" for a given error model,

$$\Pr\left(\text{failure}\right) = \sum \Pr\left(\text{failure} \mid \text{input}\right)\Pr\left(\text{input}\right). \tag{1.44}$$

The decoders we will discuss here are deterministic so the conditional probability is either 0 or 1. This often cannot be computed exactly in practice as the number of inputs is either $|\mathbb{F}_q^n|$ for classical and $|\mathbb{F}_{q^2}^n|$ for quantum. In these cases estimates of (1.44) are preformed via Monte Carlo simulations.

To the author's knowledge, there is no explicit discussion on choosing the proper the sample size for estimating (1.44) in the QEC literature. It is possible this is widely understood, but many papers appear to dramatically over or under sample, the former being a waste of resources. Applying the central limit theorem, as long as the sample size, N is "large enough", randomly sampled estimates of (1.44) produce a normal distribution. Standard arguments about Z-scores give

$$N > \left(\frac{\sigma \Phi^{-1}\left(\frac{1+\alpha}{2}\right)}{\epsilon} \right)^2, \tag{1.45}$$

where $\Phi^{-1}(\cdot)$ is the quantile function (the inverse of the cumulative distribution function), α is the confidence level, and ϵ is the accuracy tolerance. For example, if we want to estimate (1.44) with 97% confidence that the true value is within 10^{-4} of the sampled value, $\alpha = 0.97, \epsilon = 10^{-4}$. The standard deviation of the sampling, σ, is unknown and must be estimated by preforming a small number of test simulations. Note that for small codes

($n \sim \mathcal{O}(1)$) and high tolerance, N is often larger than the population size. The sample size is directly correlated to the probability of error p through σ and should be computed independently for each data point.

Consider the qubit depolarizing noise channel[12]

$$\mathcal{E}(\rho) = (1-p)I\rho I^* + \frac{p}{3}X\rho X^* + \frac{p}{3}Y\rho Y^* + \frac{p}{3}Z\rho Z^*. \tag{1.46}$$

Here, a qubit either has an (i.i.d.) error with probability p or it doesn't with probability $(1-p)$; hence errors form a binomial distribution. When an error occurs, elements of $\mathcal{P}_1$ are randomly sampled with a uniform distribution. Collecting the errors on every qubit (a Bernoulli process) forms an element of $\mathcal{P}_n$. Note that this is different than uniformly sampling elements of $\mathcal{P}_n$. The probability of the input (1.44) is therefore $\Pr(v) = p^{\mathrm{wt}(v)}(1-p)^{n-\mathrm{wt}(v)}$. The expected value, here the weight of the error, is np. For fixed p, as n increases, d must therefore also increase to keep (1.44) low. The binomial distribution however, is skewed right (positive skew) for $p < 0.5$ so even if the expected value of errors is equal to the number of correctable errors, the majority of errors are going to have weight less than this. A more precise statement can be made using the cumulative distribution function.

This is good for physical implementations of the code but can be bad for sampling as in many cases of interest the majority of samples contain no errors. The decoder should detect the no-error case and return the input resulting in a successful decode. The artificial number of successes can underestimate (1.44). To handle this we may either sample until a certain number of errors are detected, which may take a long time and could skew the data, or switch from direct (forward) sampling to importance sampling techniques. To see what

[12]We include the adjoints here to make the connection back to the operator formalism from which this derives but drop them from here on out as these operators are all self-adjoint.

this means in this context, consider an n, k, $d = 2t + 1$ code. Then

$$\Pr\left(\text{failure}\right) = \sum_{i=0}^{n} c_i \binom{n}{i} p^t (1-p)^{n-t},$$

where c_i is the percentage of errors of weight i that are correctable by the decoder. Note that $c_i = 1$ for $0 \leq i \leq t$ such that

$$\Pr\left(\text{failure}\right) \leq \sum_{i=t+1}^{n} \binom{n}{i} p^t (1-p)^{n-t} = 1 - \sum_{i=0}^{t} \binom{n}{i} p^t (1-p)^{n-t}.$$

If the goal is to estimate each of these terms to a tolerance of ϵ', then we may precompute the values

$$A_i = \binom{n}{i} p^t (1-p)^{n-t}$$

and sample the weight-i vectors to estimate c_i for all i such that $A_i \leq \epsilon'$. In general, importance sampling samples from a new, biased distribution then recombines the results using a weighting function in a manner consistent with the original problem. The second part of this statement requires proof which is missing in the literature but we will skip this here. This is more efficient for low p but as p increases the number of i to include also increases. At some p, the number of i is too large and direct sampling is more efficient. A common metric used to determine the transition point is the speed-up factor σ_D^2/σ_I^2, where the σ's are the standard deviations from direct and importance sampling, respectively. References [106, 109] briefly discuss importance sampling in the context of this work.

What it means for decoding to fail is different for classical and quantum codes. Consider first the classical case. In this scenario, a message at the source is said to be transmitted over a channel to a receiver. The channel is noisy so the message at the source and receiver may be different. If so, a decoder tries to recover the message at the source given the message at the receiver and some knowledge about the channel. Let $X \in \mathcal{C}$ be a random variable over the inputs and $Y \in \mathbb{F}_q^n$ be a random variable over the outputs. The

message at the source, x, may be thought of as chosen at random with probability $\Pr_X(x)$. The channel is determined by its transition probability $\Pr_{Y|X}(y \mid x)$ and is assumed to act i.i.d.. on each bit. By a basic result in coding theory, elements of $\mathbb{F}_q$ are evenly distributed amongst elements of $\mathcal{C}$. Given a message y at the receiver, the decoder implements the map $y \mapsto \hat{x}(y) \in \mathcal{C}$. This is incorrect with probability $1 - \Pr_{X|Y}(\hat{x}(Y) \mid y)$, in which case decoding is declared to have failed.

To minimize the risk of failure, choose $\hat{x}(Y)$ to maximize $\Pr_{X|Y}(\hat{x}(Y) \mid y)$,

$$\begin{aligned}\hat{x}^{MAP}(y) &= \operatorname*{argmax}_{x \in \mathcal{C}} \Pr_{X|Y}(x \mid y) \\ &= \operatorname*{argmax}_{x \in \mathcal{C}} \Pr_{Y|X}(y \mid x) \frac{\Pr_X(x)}{\Pr_Y(y)} \\ &= \operatorname*{argmax}_{x \in \mathcal{C}} \Pr_{Y|X}(y \mid x) \Pr_X(x).\end{aligned}$$

This is called maximum *a posteriori* (MAP) decoding. If all codewords are equally likely, $\Pr_X(x)$ is uniform and

$$\hat{x}^{MAP}(y) = \operatorname*{argmax}_{x \in \mathcal{C}} \Pr_{Y|X}(y \mid x) = \hat{x}^{ML}(y). \tag{1.47}$$

This is called maximum likelihood (ML) decoding. Ties in all schemes are broken arbitrarily. As written, these equations consider elements of the code as a whole, but we may repeat the same arguments to instead maximize the probability at each vector index. This is called bit-wise decoding and the above approach block-wise decoding.

Classically, we have $y \in \mathbb{F}_q^n$ and we want $x \in \mathbb{F}_q^k$, which we recover by identifying the error $e \in \mathbb{F}_q^n$. In QEC, the state of the quantum system encoded in some quantum stabilizer code $\mathcal{Q}$ is described by some density matrix ρ. An error $E \in \mathcal{P}_n$ occurs and $\rho \mapsto \mathcal{E}(\rho) = E\rho E^*$. We have only the syndrome $s \in \mathbb{F}_q^{n-k}$ and we wish to recover ρ, which we will do by identifying E. Since we physically cannot talk of ρ directly, we can think of ρ as $I\rho I$ and consider the map $I \mapsto E$. We also don't have access to the output state

$E\rho E^*$, so the best we can do is use the measured syndrome s. As before, if $\hat{E}(s) \in \mathcal{P}_n$ is the output of the decoder given the syndrome s, we want to maximize $\Pr(\hat{E}(s) \mid s)$, where we have dropped the notational dependence on the random variables to simplify upcoming expressions.

Underlying the classical case is that if $y = x + e_1 = x + e_2$ then $e_1 = e_2$ for some errors $e \in \mathbb{F}_q^n$ introduced by the channel. This does not hold on the quantum side, further complicating the decoding process. To see this note that $\mathcal{S} \leq C_{\mathcal{P}_n}(\mathcal{S}) \leq \mathcal{P}_n$, so we can write any element $E \in \mathcal{P}_n$ in the form $E = TLS$, where $T \in \mathcal{P}_n \backslash C_{\mathcal{P}_n}(\mathcal{S})$, $L \in C_{\mathcal{P}_n}(\mathcal{S}) \backslash \mathcal{S}$, and $S \in \mathcal{S}$. Physically, T's are called pure errors, L's are logical operators, and S's are stabilizers.

It is easy to find a T given a syndrome s. In the physics literature this is often described by promoting each stabilizer to a stabilizer-destabilizer pair by assigning an element which commutes with all generators except for the given stabilizer. Pure errors T are then given by linear combinations of destabilizers corresponding to nonzero syndrome bits. We will ignore the storage of and all discussion of destabilizers in this work and stick to the purely mathematical approach of simply multiplying the syndrome by an appropriate pseudoinverse. The traditional Moore-Penrose formula is invalid for many modules in this work since for self-orthogonal codes $G^T H = H^T H = 0$. Instead, denote the stabilizer matrix by H, append an appropriate identity, $(I \mid H^T)$, and row reduce H^T to the form $(I\, 0)^T$. Write the result as $MH^T = (I\, 0)^T$. The rows of M corresponding to the identity on the right give the appropriate pseudoinverse. Denote the pure error corresponding to s by T_s.

It remains to find L and S given s, or equivalently, given T_s. Multiplication by elements of $\mathcal{S}$ have no effect on the system or the syndrome since $\mathcal{S}$ is closed under this operation. Hence, a pure error T_s has the same effect and syndrome as $T_s S$ for any $S \in \mathcal{S}$. Thus, unlike in the classical case, multiple distinct errors can produce the same output. This is called degeneracy and QECCs with this property are called degenerate. Unlike the classical case, what is going on at each qudit is both inaccessible and irrelevant as long as the system

is returned to the joint $+1$-eigenspace; bit-wise decoding does not exist in QEC. Hence, it does not matter which element of the coset $T_s\mathcal{S}$ is chosen for the correction step as all will return the system back to the eigenspace. The element L we do very much want to know, however, as it has a detrimental effect on the integrity of the stored data. The problem is therefore

$$\hat{E}^{\mathrm{ML}}(s) = \underset{E\in\mathcal{P}_n}{\operatorname{argmax}} \Pr\left(L, S \mid T_s\right) = \underset{E\in\mathcal{P}_n}{\operatorname{argmax}} \frac{\Pr\ (L, S, T_s)}{\Pr\left(T_s\right)} = \underset{E\in\mathcal{P}_n}{\operatorname{argmax}} \Pr\ (L, S, T_s)\,, \tag{1.48}$$

where the last equality holds because $\Pr\left(T_s\right)$ is just a positive constant. It follows that we can sample the joint probability distribution instead of the conditional probability distribution. This is suboptimal because it searches for the most likely error instead of the most likely equivalence class of errors defined by the sum of the probabilities of each element in the coset. A decoder taking the degeneracy of errors into account would therefore solve

$$\hat{E}^{\mathrm{DML}}(s) = \underset{L\in C_{\mathcal{P}_n}(\mathcal{S})\setminus\mathcal{S}}{\operatorname{argmax}} \Pr\left(L \mid T_s\right) = \underset{L\in C_{\mathcal{P}_n}(\mathcal{S})\setminus\mathcal{S}}{\operatorname{argmax}} \sum_{S\in\mathcal{S}} \Pr\left(L, S \mid T_s\right). \tag{1.49}$$

Note that for the depolarizing channel (1.46),

$$\begin{aligned}
\hat{E}^{\mathrm{ML}}(s) &= \underset{E\in\mathcal{P}_n}{\operatorname{argmax}} \Pr\ (L, S, T_s) \\
&= \underset{E\in\mathcal{P}_n}{\operatorname{argmax}} p^{\mathrm{wt}(T_sLS)}(1-p)^{n-\mathrm{wt}(T_sLS)} \\
&= \underset{E\in\mathcal{P}_n}{\operatorname{argmin}} \mathrm{wt}(T_sLS) = \hat{E}^{\mathrm{wt}}(s).
\end{aligned} \tag{1.50}$$

Decoders solving Equation (1.50) are called minimum-weight decoders and are a popular choice of decoder for QECCs. Confusingly, decoders solving Equation (1.48) are also sometimes called minimum weight by extension. Decoders solving Equation (1.49) are sometimes called ML (e.g. [110, 111]), despite the fact that Equation (1.48) matches the classical ML problem. Here, we align with decades of coding theory literature and refer

to Equation (1.48) as ML decoding, Equation (1.49) as degenerate decoding, and Equation (1.50) as minimum-weight decoding. Maximum-likelihood degenerate decoding (MLDD) or maximum-likelihood logical decoding (MLLD) would also be good descriptions for (1.49).

The large majority of current decoders for QECCs attempt to solve (1.48) and not (1.49). Ties are usually broken arbitrarily. The difficulty in switching from (1.48) to (1.49) is that the latter not only has to solve the former but also organize the results into cosets. It often happens that $\hat{E}^{\mathrm{ML}}(s)$ is not an element of the most likely equivalence class of errors and $\hat{E}^{\mathrm{ML}}(s) \neq \hat{E}^{\mathrm{DML}}(s)$. Classical and non-degenerate decoding is NP-Hard [112, 113, 114, 115] and degenerate decoding is $\#P$-complete (up to Turing reduction) [116]. See [116] for a discussion on the impact of degenerate decoding.

Since T_s is easy to find, any good quantum decoder should at least return the system back to the correct eigenspace. Success and failure is thus defined by whether or not a non-trivial logical operator occurred on the system. Consider the Steane code in Figure 1.6a and number the qubits 1, 2, 3 along the bottom, 4 on the left, 5 in the center, 6 on the right, and 7 on top. Label the red face (stabilizer) by f_1, blue by f_2, and green by f_3. Order the syndrome bits by (f_1, f_2, f_3). This is a distance three code and so can correct $t = (3-1)/2 = 1$ errors. If an error happens on qubit 1, only the red face is affected and the syndrome is $(-1, 1, 1)$. The same goes for the other two corners by symmetry. If qubit 2, 4, or 6 has an error, the resulting syndrome pattern flips two of the three faces and the error is uniquely identifiable. If the middle qubit 5 has an error, the syndrome is $(-1, -1, -1)$, which also uniquely identifies the error. In this way, syndrome decoding corrects all single-qubit errors. Now consider the two-qubit error on 1 and 4. Two errors on the red face leave the overall parity of at $+1$, the blue face is not affected, and the green face has a single error: $(1, 1, -1)$. This appears like a single error on qubit 7 and correcting this way leading to a string of errors on 1, 4, and 7. This is a logical operator for this code and was not made by the error channel but by the decoder itself. This is the danger of smaller

QECCs.

We can check for logical errors during simulations by noting that by the $\{\overline{X}_i, \overline{Z}_i\}$ commutation relation (1.30) any $E = T_s L S$ with nontrivial L must not commute with at least one canonical logical operator generator. This is a bit artificial since we don't have access to the current state of the system experimentally, and since errors can occur during syndrome extraction and propagate and measurement errors can occur we must assume that the final round of syndrome measurements is perfect.

Computer simulations of QECCs randomly sample errors and determine the probability of failure for a given error model over a range of error probabilities. This produces a curve on the graph of "logical error rate" versus probability of error. The goal is to determine the maximum possible probability at which using the QECC protects the information better than if it was unencoded. This value is called the threshold and is historically defined by a recursive construction called concatenation. Assume for the moment that $k \mid n$ and call the unencoded data level $L = 0$. First encode k bits of information into n physical bits - call this $L = 1$. Then repeat the process by encoding the n bits again to give $L = 2$. The threshold is the (unstable) fixed point of the logical error rates of this map as $L \to \infty$. For families of codes which naturally have a constructive method for increasing the distance, such as the rotated surface and color codes, a threshold can be defined as the probability p_{th} at which for $p < p_{th}$ the logical error rates go to zero as $d \to \infty$. Many QECCs do not have thresholds since this point may not exist. Analytical proofs of the threshold, called the threshold theorem, have been completed for a handful of codes and error models and establish the theoretical justification that QEC can work as intended [117, 118, 119, 120, 121].

It is difficult, if not impossible to exactly compute the threshold. Even computer simulations can only practically handle small L. Trends in the data might also be deceptive. For example, the codes of the next chapter are simulated for $d < 21$ and there is strong evidence that the data is converging. However, [122] points out that the true threshold of a certain

decoder for a surface code only truly appears in their simulations for $d > 21$. Computing the threshold is often relaxed to computing a pseudothreshold which is defined at some finite L. A pseudothreshold is the probability where one code outperforms another for all probabilities of smaller magnitude. This is can be pictured as the intersection between the logical error rate curves of the two codes, if it exists. Arguments may be made about the threshold by studying the limit of the pseudothresholds of between increasing levels of L or d. This is the approach taken in the following chapters.

Note that the threshold is not a topological invariant. Different cellulations of the same manifold can produce different thresholds. The rotated surface and color codes considered here are special cases in this regard as the X and Z stabilizers have a symmetry. The cellulation and its dual cellulation are generically different, providing an X-Z asymmetry in error correction. In this case we can define an X minimum distance and a Z minimum distance $[[n, k, d_X/d_z]]$ such that $d = \min\{d_X, d_Z\}$. This is useful in physical systems where, for example, Z errors are significantly more common than X errors. This is of course only applicable for CSS codes. Fujii and Tokunaga demonstrated the difference between X and Z thresholds for the independent X and Z channels (below), respectively, on the kagome, hexagonal, and tri-hexa lattices on the surface of a torus [123].

As described, the threshold is a measure of how well a system can store information. This may be extended to measure the threshold of a specific circuit such as a fault tolerant gadget. It may be that the circuit is so complex that the probability of error required to run it successfully is orders of magnitude lower than that for pure error correction. We will not consider this further in this work. Regardless of what is being measured, the threshold is important because it constrains the physical error limits which must be experimentally realized but also, in combination with fault tolerance, the resource overhead required to achieve it (e.g. the level of L).

We establish the foundations of a new decoder to solve (1.48) in Chapter 2 and then show how to extend the idea to solve (1.49) in Chapter 3. To properly understand its role

in the field, we make numerous references to current state-of-the-art decoders in the literature, specifically those based on minimum-weight perfect matching (MWPM). There are three types of simulations: code-capacity (memory model), phenomenological, and circuit-level. In code-capacity errors occur on the data qubits and all circuit elements are assumed to work perfectly. Hence, a specific circuit is not needed since it will always return the syndrome given by simple matrix-vector multiplication. In the phenomenological model, errors occur on data qubits but measurement errors are also taken into account in which a measurement returns the correct eigenvalue with probability $1 - p_m$ and a different eigenvalue with probability p_m. This takes Definition 1.2.1 (measurement) into account. Finally, a circuit-level simulation assumes data errors, idling errors, gate errors, and measurement errors.

Circuit-level simulations are the ideal but take considerably more resources and require explicit (hopefully fault-tolerant) circuits. Any decoder designed for a code-capacity simulation can be extended to a phenomenological model by repeating every syndrome measurement a certain number of times, majority voting on each bit, then treating the result as in the code-capacity case. This often leads to a poor result and a bit of thinking can design a better system. It is unclear how to extend many decoders to circuit-level simulations. It's clear that the results of each level of simulation is an upper bound on the next level of detail, but the difference between code-capacity and circuit-level results can be several orders of magnitude. The new decoding technique described in this work is currently restricted to code-capacity. Phenomenological models could be done as above, but we see no reason to do this at this moment before the technique is fully extended to circuit-level; a strong limitation towards practical implementation. Let us briefly review the MWPM decoder for code-capacity simulations; we will compare our technique against it in the next chapter.

Consider the rotated surface code of Figure 1.2. MWPM assumes an i.i.d., independent

X-Z error model,

$$\mathcal{E}_X(\rho) = (1-p_X)I\rho I + p_X X\rho X \tag{1.51}$$

$$\mathcal{E}_Z(\rho) = (1-p_Z)I\rho I + p_Z Z\rho Z \tag{1.52}$$

such that

$$\Pr(0) = (1-p_X)(1-p_Z) \quad , \quad \Pr(1) = p_X(1-p_Z),$$
$$\Pr(\alpha) = (1-p_X)p_Z \quad , \quad \Pr(\alpha^2) = p_X p_Z$$

for each qubit and decodes each channel separately. This error model and decoding scheme is common for CSS codes. Without loss of generality, consider just the X stabilizers (grey faces of Figure 1.2) which detect Z errors from (1.52). If an error happens on qubit 7, for example, the stabilizers $X_1X_2X_6X_7$ and $X_7X_8X_{12}X_{13}$ will return -1, uniquely identifying the location of the error. If an error $Z_7Z_8Z_9$ happens, stabilizers $X_1X_2X_6X_7$ and $X_9X_{10}X_{14}X_{15}$ are triggered. The other two stabilizers along this path have two errors each keeping the overall parities at $+1$. Also note that errors $Z_7Z_8Z_9Z_{14}$ and $Z_7Z_{12}Z_{13}Z_{14}$ give equivalent syndromes. Thus, for any continuous "string" of errors we see that we can only ever detect the endpoints. Given a collection of endpoints, the goal is therefore to determine which endpoints belong to the same error string, i.e., the equivalence class of the pure error in the first homology group.

Define a new graph by placing a vertex on each face with measurement outcome -1. The vertices are connected in a complete graph with weights given by the Manhattan metric in the original lattice. Matching endpoints is now a minimum-weight perfect matching on this graph [110]. The minimum-weight condition ensures this returns $\hat{E}^{\text{wt}}(s)$. We refer the interested reader to standard graph theory texts for details on bipartite matching and (e.g.) [124, 122, 125] for details on its application to QEC. The original algorithm by Edmonds [126, 127] has runtime $\mathcal{O}(n^4)$ but modern implementations have reduced to this almost

$\mathcal{O}(n^2)$ [128]. Further simplifications for quantum have reduced this to an impressive $\mathcal{O}(1)$ parallel runtime [122, 125]. (For example, the binomial distribution says that high-weight error strings are less likely than low-weight strings implying that it probably suffices in practice to not build a complete graph but instead only connect nearby vertices.)

MWPM can be modified to handle the full depolarizing channel (1.46) [129].[13] For phenomenological models, the syndromes are measured multiple times, producing a matching graph for each. Each round of measurements is placed at a discrete time $t = 1, 2, \ldots$ and edges are now added between time slices. This should be viewed as a foliation of the surface. Full circuit-level simulations may also be done by modifying the weights on the edges of the graph to relate to probabilities of errors due to things like hook errors [130]. It turns out that the surface codes have a high threshold and are therefore good for storing quantum information. But, as seen in Figure 1.2, stabilizers either have Hamming weight two or four, prohibiting a transversal implementation of the full Clifford group. The color code family has a lower threshold but better gate set, making it more useful for quantum computation.

An interesting thing about color codes is the entire concept of color. In particular, we did not need to invoke them at all to define the code. Consider Figures 1.4b, 1.5, and 1.6a. Each boundary of the triangle is missing a single color due to the puncture. For example, in Figure 1.5 the left is missing blue faces, the bottom red faces, and the right green. Each boundary is assigned a color equal to its missing color. When decoding, the stabilizers (faces) are measured and the corresponding eigenvalues recorded. To model this, take the dual graph where each face becomes a vertex of the same color and assign the eigenvalue to the vertex. In the literature, an error on a red face can only be matched with other red errors or a red boundary, and similarly for the other colors. The general idea is to make subgraphs (of the dual graph) containing combinations of two colors, ignoring

[13] One may also attempt to incorporate correlations by simply applying Bayes' theorem. After determining the X errors, the Z channel probabilities should be updated as $\Pr(I \mid I) = \Pr(I)/(\Pr(I)+\Pr(Z))$, $\Pr(I \mid X) = \Pr(X)/(\Pr(X)+\Pr(I))$, $\Pr(Z \mid I) = \Pr(Z)/(\Pr(I)+\Pr(Z))$, $\Pr(Z \mid X) = \Pr(Y)/(\Pr(X)+\Pr(Y))$. Similar expressions can be derived for qudits.

the third color. Popular color code decoders then project these 2-colorable subgraphs onto separate copies of the surface code, decode using MWPM, and then attempt to combine and interpret the results back on the original color code lattice [131, 132, 133, 134, 135, 136]. As clever as these ideas are, there are problems with this method. In short, the two code families are fundamentally different and too much information is lost in the back-and-forth mappings. Kubica and Delfosse used homological insights to improve this but problems still remain [61, 137, 138]. It is the (unpopular) opinion of this author that the main problem with these approaches is the artificial reliance on colors. Furthermore, the boundary maps $C_2 \rightarrow C_0$ instead of $C_2 \rightarrow C_1$ make forcing homological results into geometrical arguments awkward. Trying to mimic MWPM natively on the color codes leads to a hypergraph matching problem for which there is no known efficient algorithm. We tackle this in Chapter 2 where we develop a decoder which handles the syndrome information purely combinatorially without any reference to dual graphs, homology, or colors. This is the first native decoder for the color codes.

We will not deal with this here but note for completeness that the error models previously discussed are based on the idea that $\mathcal{P}_n$ is a basis for the space. This is not always accurate due to physical considerations concerning the nature of the environment in Equation (1.13). Some error models, such as the pulse-area model of coherent errors, cannot be written in this manner and simulations studying them must keep track of all the information in $\mathbb{C}^{q^n}$ and can only be done for small q, n. Current proofs of the threshold theorem do not take such errors into account. Numerical evidence suggests that including things like coherent errors can drastically lower (pseudo)thresholds (e.g. [139]). Other popular error models include the amplitude damping channel and the so-called Bloch sphere polarization given by Kraus operators

$$E_0 = |0\rangle\langle 0| + \sqrt{1-\gamma}|1\rangle\langle 1| \qquad , \qquad E_1 = \sqrt{\gamma}|0\rangle\langle 1|$$

and

$$E_0 = \sqrt{1-p_\phi}I \qquad , \qquad E_1 = \sqrt{p_\phi}[\cos(\phi)X + \sin(\phi)Y],$$

respectively.

1.6 Extension To Arbitrary Rings

The theory of stabilizer codes can be generalized to other rings. To see this, notice that the finite field only serves as a label for the orthogonal basis for $\mathbb{C}^q$ and can therefore be replaced with $\{|r\rangle \mid r \in R\}$ for some appropriate ring R. The operators become $X(a)|r\rangle = |r+a\rangle$ and $Z(b)|r\rangle = \chi(br)|r\rangle$ for $a, b \in R$ and χ an irreducible character of $(R, +)$. The errors are $\mathcal{P}_n = \{\chi(c)X(a)Z(b) \mid a, b \in R^n, c \in R\}$. Two elements $X(a)Z(b)$ and $X(a')Z(b')$ commute if and only if $\chi(b \cdot a' - b' \cdot a) = 1$. The quantities $\mathcal{S} \leq \mathcal{P}_n$ and $\mathcal{Q}$ are defined as before and can be studied via the map to R^{2n}. There exists a unique $\psi(r)$ such that $\chi(r) = e^{2i\pi\psi(r)}$, producing the bilinear form $\langle (a,b),(a',b')\rangle_\chi = \psi(b \cdot a' - b' \cdot a)$. Orthogonality is now given by left and right annihilators

$$\mathcal{S}^\perp = \{r \in R^{2n} \mid \langle S, r\rangle_\chi = 0 \, \forall S \in \mathcal{S}\}$$
$$^\perp\mathcal{S} = \{r \in R^{2n} \mid \langle r, S\rangle_\chi = 0, \forall S \in \mathcal{S}\}.$$

One may show that R must be a Frobenius ring. Examples of Frobenius rings are finite fields, integer rings modulo a number, and Galois rings. The emphasis in the current literature has been on finite chain rings. Let R be a finite chain ring with Jacobson radical $J(R) = \mathrm{rad}(R)$. Then the residue field $\mathbb{F} = R/J(R)$ allows us to reconstruct the previous theory. It is yet unclear if there is any advantage of using the heavy machinery of commutative algebra to obtain codes with the same parameters as ones over finite fields. It is also an experimental problem to determine how to implement the new ring and its operations. Although we do not do it here, the proceeding chapters may also be cast into this framework, providing the first decoder for these codes. Rains presents extensions to algebraic

number theory in [140].

CHAPTER 2

TRELLIS DECODING STABILIZER CODES

This chapter reproduces work from “Trellis Decoding For Qudit Stabilizer Codes And Its Application To Qubit Topological Codes" by E. Sabo, A. B. Aloshious, and K. R. Brown [141]. Throughout this chapter we restrict to the prime subfield $q = p$ and all stabilizer phases will be set to $\eta^c = 1$. Elements of $\mathbb{F}_{p^2}^n$ will be denoted by length-n vectors in the $\{1, \alpha\}$ basis such that $X(a)Z(b)$ is written $a + b\alpha$. Occasionally, $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ will be denoted in physics notation by the elements $\{I, X, Z, Y\}$, respectively.

2.1 Introduction

We ended the last chapter with a brief discussion of simulations of quantum codes and mentioned a few decoding schemes. If one reports a new QECC or family of QECCs, it is expected to also describe how to decode it. There are presently few generic decoders applicable to random QECCs. Most decoders developed so far are specifically tailored to a given code family based on intuitive, visual, or physical arguments. Many decoders and decoding schemes in the literature are unique to the simulation they are presented with while others, especially those for topological codes, have enjoyed widespread use, study, and success.

For topological codes, such as rotated surface and color codes, the decoding problem is typically reduced to MWPM on independent X and Z channels. This is applicable to topological codes whenever the syndrome and errors have a string-like pattern. The union find decoder [142, 143, 144, 145] has been successfully applied to surface codes and homological product codes, but a color code or more general stabilizer code implementation has still yet to be developed. Other popular decoders for topological codes include those based on cellular automata [146, 147, 148], integer programing [108], and renormalization [149,

150, 151]. Topological codes are often simulated on infinite lattices or finite lattices with periodic boundary conditions, and decoders sometimes require structural features such as locality and translational invariance.

Of particular interest are decoders which apply to multiple families of codes with little to no modification up to input data. Belief propagation [152, 153], tensor network [111, 154, 155, 156, 157, 158], and machine learning [159, 160, 161, 162, 163, 164, 160] based decoders generally fall into this category. Belief propagation is useful when codes satisfy specific sparsity properties but is inherently more difficult for quantum than classical codes. Recent work has improved this by introducing a common classical post processing step to prevent the decoder from getting stuck in loops [165, 166]. Tensor network decoders are theoretically exact maximum likelihood decoders but remain practically limited by computation with finite resources. As with much of machine learning, these decoders trade good performance with a thorough understanding of its decisions and the theoretical guarantees that come with it.

One benefit to general decoding techniques is that they remove things such as geometric or topological constraints that complicate algorithms. There are a few general decoding techniques for classical codes, but they typically involve bit-wise comparisons between the received vector and the suspected input vector, which is just not applicable for QEC where the only available information is the syndrome. Ollivier and Tillich pointed out how to port one of the most successful of these classical algorithms to QEC in a short four page 2006 paper [167]. This is applicable to any stabilizer code but only a $[[5, 1, 1]]$ code was explored. As written, the decoder was not practical and could not be used for large-scale simulations, and as such no numerical results were included. It was unclear if the proposed decoder was useful, and it was only revived again in [168, 169] for quantum convolutional codes, which are a different concept in coding theory than the linear codes explored in this work. The decoding algorithm relies on the stabilizer matrix being in a certain canonical form and [170] presented an algorithm for this over $\mathbb{F}_p^{2n}$.

The proposed decoder works by building a highly compact graphical representation, called a trellis, of the algebraic structure of the image of $\mathcal{S}^{\perp} \coloneqq C_{\mathcal{P}_n}(\mathcal{S})$ in $\mathbb{F}_{p^2}^n$. The trellis contains all valid combinations of logical operators and stabilizer generators that will return the system to its code space. Decoding proceeds by using dynamical programming to globally search for the most likely path in the trellis corresponding to the measured syndrome. This is performed efficiently in exactly n major steps for an $[[n, k, d]]$ stabilizer code, although the amount of work required in each step varies with respect to a predictable, code-dependent formula and can be significant depending on the amount of available resources. Many fundamental questions and theoretical properties remained unanswered following [167].

In this work, we make a simple observation which prevents the repeated processing of potentially massive amounts of data (all of the elements of $\mathcal{S}^{\perp}$), thus making trellis decoding viable. However, this is still unpractical as enumerating the elements of $\mathcal{S}^{\perp}$ may not be possible for many interesting codes. Here we expand on the previous literature by fully developing the theoretical foundations of the decoder. The outcome is a way to extract all the information needed to construct the trellis solely from the generators of $\mathcal{S}$ and $\mathcal{S}^{\perp}$. This object is independent of error model and may be computed once and saved for future simulations.

Trellis decoding is well-studied in classical coding theory. The question is what changes in the switch from classical to quantum? We explore this in this chapter. There are two main difference right from the start between the classical and quantum cases. First, many of the best results in classical trellis theory rely on the interplay between the code and its dual, a relationship which does not exist in QEC. Such results therefore require new proofs, if they hold at all. Second, classical codes are vector spaces over $\mathbb{F}_p$ whereas quantum codes are only additive $\mathbb{F}_p$-modules over $\mathbb{F}_{p^2}$. Suppose $\{1, \alpha\}$ is a basis for $\mathbb{F}_{p^2}/\mathbb{F}_p$. Algorithms and formulas require updates because 1 cannot remove an α in, for example, Gaussian elimination without multiplication. Third, the previous chapter showed that the classical

and quantum decoding problems are fundamentally different, and classical trellis theory is a bit-wise decoder. We tackle all of these problems here, often recovering the known classical result with “quantum corrections". We point out changes and provide references to the original classical results throughout. We also provide missing proofs for claims previously made and widely cited in the classical literature which were otherwise based on observation. This work can also be seen as an extension of the classical theory to codes over $\mathbb{F}_p \times \mathbb{F}_p$.

We begin by summarizing the main ingredients and fundamental results of our work in Section 2.2. A rigorous theory is developed in Section 2.3. We attempt to mimic the classical coding theory literature as closely as possible to make this work available to the widest audience. It is perhaps remarkable that many of our results have classical analogues despite the differences between classical and quantum codes forcing alternative proof techniques. We view this as a strength of the overall theory of trellis decoding.

In Section 2.3.3 we adopt a classical, quantitative metric for the difficulty of decoding based on the structure of the trellis to stabilizer codes. To the author’s knowledge, this is the first such metric for QECCs. This allows us to show that the color codes are fundamentally more difficult to decode than the rotated surface codes, and the 4.8.8 color code is more difficult to decode than the 6.6.6 color code, facts which were only intuitively understood from past simulations. We also make quantitative arguments as to exactly how much easier it is to decode a CSS code using independent X and Z decoders versus a single decoder for the full code. Although not unique to trellis decoding, we show that large gains in terms of storage and time complexity are made when splitting the trellis into X and Z parts.

Numerical results for code-capacity simulations are presented in Section 2.5. First, we demonstrate the power of the theory by decoding four high-density codes from the QEC Best Codes Table at [171]. As we are unaware of any other decoding algorithm for these codes, it is difficult to quantify the performance of the trellis with this data. We therefore, second, proceed to apply our work to the rotated surface and color code families for

which there already exist numerous and highly optimized decoders. Notably, the color codes (with boundary) are natively decoded without reference to other codes or notions of color, homology, matching, boundaries, lifting, projections, restrictions, charges, excitations, strings, and/or mappings. Trellis decoding for stabilizer codes over the X, Z, or depolarizing channels is optimal minimum-weight and numerical results for Z-only noise should therefore match that of MWPM for the surface codes and exceed the current best thresholds in the literature for the color codes, although we only simulate codes up to distances 17-21. Finally, we simulate a 2-stage, suboptimal decoder for the level-2 concatenated Steane code and compare it to a concatenated minimum weight decoder.

The main goal and overall contribution of this work is in making trellis decoding practical. To date, it is the most general and widely applicable decoding technique in QEC.

2.2 The Syndrome Trellis

The classical coding theory literature contains several definitions for the trellis of a linear block code. Some of them, such as the the Bahl-Cocke-Jelinek-Raviv (BCJR) [172], the Wolf [173], and the Forney-Muder [12, 174] trellises, are now known to be isomorphic. With the benefit of this hindsight, Ollivier and Tillich demonstrated how to port what is often known as the syndrome trellis to the quantum setting [167]. They referred to their construction as the Wolf trellis; however, the discussion here more closely follows that of BCJR. As such, we simply use “syndrome trellis" in this work, although, since unlike classical coding theory, there is only one definition of a trellis for quantum error correction, we may just as well shorten this to simply “trellis". One may attempt to port the other classical trellises to the quantum case but the definition used in [167] and this work is known to be minimal in a rigorous sense that will be defined in Section 2.3, rendering alternate definitions undesirable. The interested reader is referred to the tutorial piece [175] for a general discussion of classical trellises.

Recall that a directed edge, e, in a graph goes from source, $s(e)$, to terminus, $t(e)$.

Definition 2.2.1 ((Quantum) Syndrome Trellis)

A trellis for an $[[n, k, d]]_p$ stabilizer code with stabilizer group $\mathcal{S}$ is a directed multigraph with vertex set, V, and edge set, E, such that

(i) there are $n+1$ disjoint sets of vertices V_i with $V = V_0 \sqcup \ldots \sqcup V_n$ and $|V_0| = |V_n| = 1$;

(ii) there are n disjoint sets, E_i, of directed edges from V_{i-1} to V_i with $E = E_1 \sqcup \ldots \sqcup E_n$;

(iii) each vertex $v \in V_i$ has a unique label given by an $(n-k)$-tuple of syndromes, although the same label may be present in multiple V_i;

(iv) each edge $e \in E_i$ is a unique triple of the form $(s(e), P, t(e))$, where $P \in \mathcal{P}_1$ is a label;

(v) each edge is assigned a weight, $\mathrm{wt}(e) \in \mathbb{R}^- \cup \{-\infty\}$.

Following the classical literature, vertices are referred to as states, the V_i as state spaces, and V_i is said to be at depth i. The edge sets E_i are referred to as the ith section and the edges as branches. The condition that edge labels must be unique between a fixed pair of source and terminus vertices means our trellises are proper. Trellises which do not satisfy this condition are called improper and are not considered in this work. It is perhaps more convenient to define the trellis without weights (v), but we stick with decades of precedent in including them here.

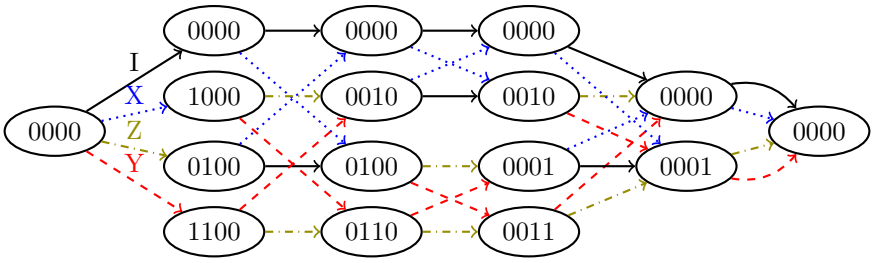
To construct the syndrome trellis, choose a set of stabilizer generators $\{S_1, \ldots, S_{n-k}\}$ for the stabilizer group $\mathcal{S}$, and fix this set throughout. For $P \in \mathcal{P}_n$, let a subscript P_i denote the ith element of this product.

Definition 2.2.2

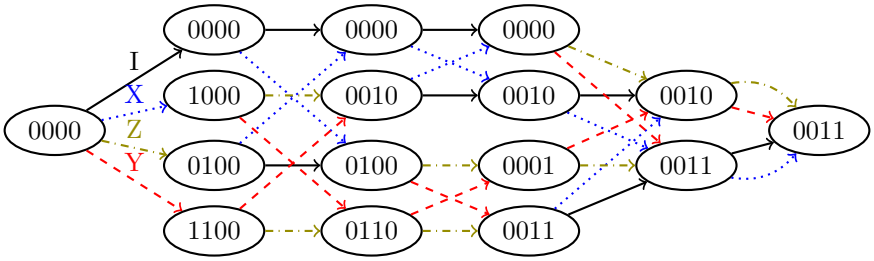
Let $\mathcal{J} \subset \{0, \ldots, n\}$ be an index set. Then define $\pi_{\mathcal{J}} : \mathcal{P}_n \to \mathcal{P}_n$ to be the map which projects all elements at indices $\mathcal{J}^c$ to the identity, where $\pi_0(P)$ is the all identity.

Let $\sigma_i : \mathcal{P}_n \to \mathbb{F}_p^{n-k}$ be the map from an element to its syndrome,

$$\sigma_i(P) = (\langle S_1, \pi_{\{0,\ldots,i\}}(P)\rangle, \ldots, \langle S_{n-k}, \pi_{\{0,\ldots,i\}}(P)\rangle), \tag{2.1}$$



(a)



(b)

Figure 2.1: (a) The trellis for the $[[5,1,1]]$ code with stabilizers $\{[\alpha,1,0,0,0], [1,\alpha,1,0,0], [0,1,\alpha,1,0], [0,0,1,\alpha,1]\}$ demonstrated in [167]. Measuring the syndrome $s = (0,0,1,1)$, we determine $P_s = [0,0,0,\alpha,\alpha]$. Adding $\pi_0(P_s) = \pi_1(P_s) = \pi_2(P_s) = \pi_3(P_s) = (0,0,0,0)$, $\pi_4(P_s) = (0,0,1,0)$, and $\pi_5(P_s) = (0,0,1,1)$ to the appropriate V_i produces (b), the trellis found in Figure 2 of [167]. The presence of the parallel edges in E_5 demonstrate that this code cannot tell the difference between 0/1 and α/1 + α on qubit five.

where $\langle \cdot, \cdot \rangle$ is an appropriate inner product after mapping group elements to a numerical form, and let $P_s \in \mathcal{P}_n$ with syndrome s with respect to the chosen generators, $s = \sigma_n(P_s)$. Compute $\mathcal{S}^\perp$ and then $P_s\mathcal{S}^\perp$. Since everything in $\mathcal{S}^\perp$ has zero syndrome, everything in $P_s\mathcal{S}^\perp$ has syndrome s.

The vertices in each set are labeled by the values $\sigma_i(P)$ for all $P \in P_s\mathcal{S}^\perp$,

$$V_i = \{\sigma_i(P) \,:\, 1 \leq j \leq n-k, P \in P_s\mathcal{S}^\perp\}. \tag{2.2}$$

A directed edge is created from the vertex $\sigma_{i-1}(P) \in V_{i-1}$ to the vertex $\sigma_i(P) \in V_i$ and labeled with the ith component of P,

$$E_i = \{(\sigma_{i-1}(P), P_i, \sigma_i(P)) \,:\, P \in P_s\mathcal{S}^\perp\}. \tag{2.3}$$

The weight of an edge with label P_i is defined to be the log-likelihood $-\log \Pr(P_i)$, where this probability comes from the assumed error channel. Duplicate edges, which have the same source, label, and terminus, are not allowed, although they will appear often during this method of construction. Parallel edges, which have the same source and terminus but different labels, are allowed and imply the existence of a weight one error for the code. These will generally not appear, but we will consider them in this work for completeness. The example trellis of Figure 2.1 has parallel edges in E_5.

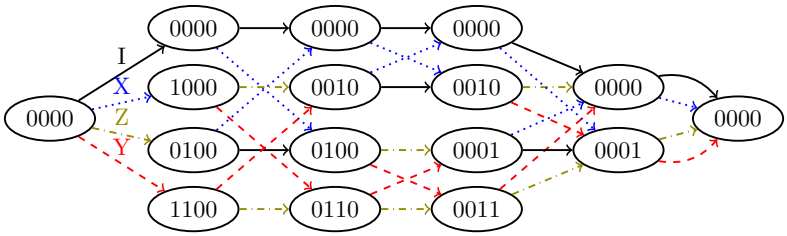
This represents the construction process of [167] and [170]. Assuming $\mathcal{S}^\perp$ is able to be computed and stored, the trellis as written so far is dependent on P_s and must be recomputed for every measured syndrome, which is computationally expensive for a practical implementation for many codes. This can be avoided by noting that since for $P \in P_s\mathcal{S}^\perp$, $\sigma_i(P) = \sigma_i(P_s) + \sigma_i(P')$ for $P' \in \mathcal{S}^\perp$, the trellis with respect to syndrome s is simply a shift of the trellis with respect to the zero syndrome. Since $\sigma_i(P_s)$ is a single value, the set $\{\sigma_i(P_s) + \sigma_i(P')\}$ is unique when $\{\sigma_i(P')\}$ is unique so $|V_i|$ remains invariant. Likewise, the map $(\sigma_{i-1}(P'), P'_i, \sigma_i(P')) \mapsto (\sigma_{i-1}(P_s) + \sigma_{i-1}(P'), P_{s\,i}P'_i, \sigma_i(P_s) + \sigma_i(P'))$ is an isomorphism permuting edge labels via the action of $P_{s\,i}$. It follows that one may precompute the trellis for the zero syndrome then update each V_i with the syndrome of $\pi_i(P_s)$, updating edges accordingly. See Figure 2.1 for an example. We may thus trade the affine space $P_s\mathcal{S}^\perp$ with the vector space $\mathcal{S}^\perp$. This is convenient as many mathematical objects are not well-defined over affine spaces and working with the associated vector space allows for easier proofs of properties that more closely mimic their classical counterparts.

We take the trellis for $\mathcal{S}^\perp$ (with respect to the zero syndrome) to be the fundamental object in this work. The general decoding scheme proceeds as follows: 1) construct the trellis, 2) measure a syndrome and shift the base trellis with respect to it, 3) decode the shifted trellis, 4) repeat steps 2) and 3). We avoid assigning edge weights to the trellis until step 2) so the trellis resulting from step 1) is independent of any measured syndromes or error model. Hence step 1) is an “offline" procedure while 2) and 3) are “online". Trellises

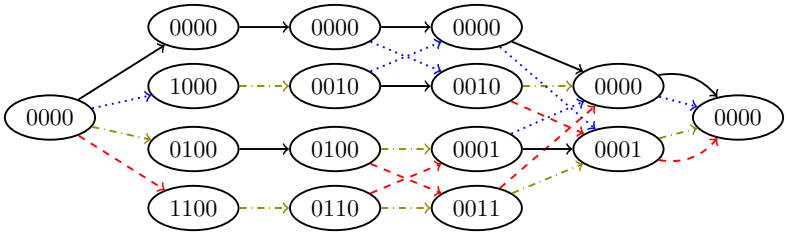
for larger distance codes may therefore be computed once and saved for future use. For the codes considered in this work, the efficiency of the construction algorithm made this unnecessary for most distances. Since $|V| \leq |E|$, shifting the trellis is $\Theta(|E|)$ in time but the procedure is easily parallelized. As we will see later, decoding is also $\Theta(|E|)$ and may also be parallelized.

With the edge weights prescribed above, the desired correction is given by the maximum-likely (or minimum-weight) path from V_0 to V_n. We refer to this in the following as the "optimal path". The Viterbi algorithm is an example of forward dynamical programming and is the most common trellis-based decoder [176]. The idea is as follows. Choose an arbitrary vertex $v \in V_i$ for some $i \neq 0$ or n and suppose that the optimal path travels through v. Then the optimal path can be split into two parts: the optimal path going from V_0 to v and the optimal path going from v to V_n. Compute the optimal path from V_0 to v and repeat for all v. Once the optimal paths are computed for every vertex in V_{i-1}, the optimal path for a vertex in V_i is chosen by finding the minimum value of the sums of the incoming edge weights plus the weight of the optimal path for each edge's source vertex in V_{i-1}. The weight at V_0 is arbitrary but is convenient to initialize to zero. Having completed this for all vertices, the Pauli correction may be read off the trellis by taking the edge labels for the optimal path connecting V_0 to V_n. Figure 2.2 demonstrates the Viterbi algorithm on Figure 2.1; an example implementation is also provided in Algorithm 1. The algorithm is often described in a manner in which vertices with no outgoing edges are removed from the system. This creates nicer diagrams but actual deletion steps can be algorithmically expensive and should be ignored as in Figure 2.1. Also note that the partial syndromes, or vertex labels, make no appearance in the algorithm and may be safely removed after the construction phase, also ignoring vertices in the shifting phase. The partial syndromes of the distance 15 rotated surface code are 224 bits long, for example, and not storing them, even in a more efficient fashion, leads to large savings in storage.

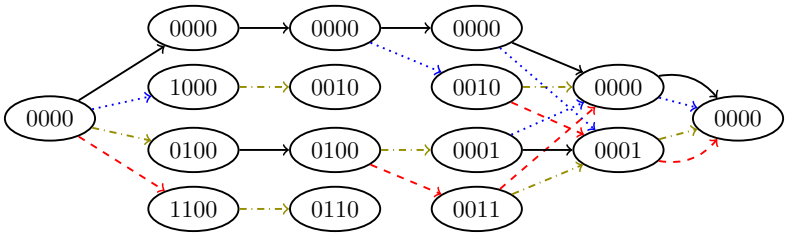
The Viterbi algorithm is sometimes confused with other standard minimum-path graph



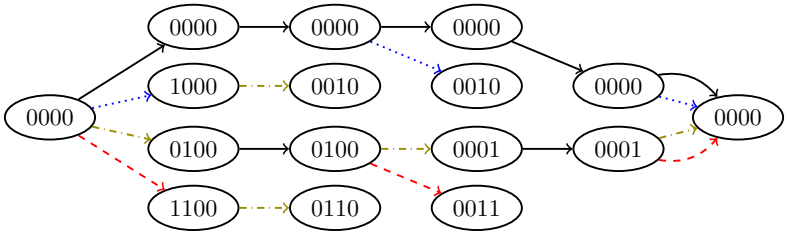
(a) Section 1



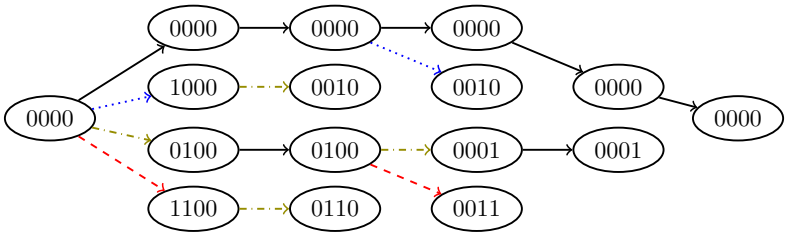
(b) Section 2



(c) Section 3



(d) Section 4



(e) Section 5

Figure 2.2: The Viterbi algorithm applied to the example trellis of Figure 2.1 for the error model $\Pr(I) = 0$, $\Pr(X) = \Pr(Z) = 1$, and $\Pr(Y) = 2$. Ties were broken in a manner to keep the result looking clean. The path from V_0 to V_n provides the final correction, in this case $IIIII$ - which is true since there was no error.

Algorithm 1: Viterbi

Input: A vertex set, V, along with an edge set, E, corresponding to a valid trellis.
Output: Pauli string

Struct *Vertex* **contains**
    int $prev$ ▷ Final step acts as a linked list
    float $value$
    Edge $edge$ ▷ Initialized to $null$
end
Struct *Edge* **contains**
    Vertex $source$
    float $weight$
    char $label$
end
$V_0 \ni \overline{0}.value \leftarrow 0$
for $i \leftarrow 1$ ***to*** n **do**
    foreach $v \in V_i$ **do**
        $v.value \leftarrow \min_{\substack{e \in E_i \\ t(e)=v}} \{e.source.value + e.weight\}$ ▷ For e which achieves minimum
        $v.prev \leftarrow e.source$
        $v.edge \leftarrow e$
    end
end
$v \leftarrow \overline{0} \in V_n$ ▷ Trace optimal path backward for correction
$correction \leftarrow$ "" ▷ Empty string
for $i \leftarrow n$ ***to*** 2 ***by*** -1 **do**
    $correction \leftarrow v.edge.label + correction$ ▷ Concatenate to left of string
    $v \leftarrow v.prev$
end
return *correction*

algorithms but it is distinct and more efficient given the strict edge and vertex dependencies required in the definition of a trellis. Underlying the success of this algorithm is the implicit assumption that the trellis behaves as a Markov chain with each V_i only depending on the result at V_{i-1}. In particular, at each V_i, all the information regarding the correction of all previous qudits has already been processed.

The behavior of the decoder depends strongly on the assumed error model. If all Pauli

operators are equally likely, as in the standard depolarizing noise model, the Viterbi algorithm acts as a minimum Hamming-weight decoder and may run into a significant number of ties which may be broken arbitrarily. For a biased noise model, the decoder is able to differentiate between, say, X and Z, and is able to make a more informed choice.

2.3 Properties

Throughout this section we denote the vertex with zero syndrome by $\overline{0}$. Dimension will always refer to the number of generators of an object, and $|\cdot|$ will be reserved for cardinality. An element of $\mathcal{P}_n$ will be denoted by P, whereas p will denote the characteristic of the underlying field. We first show several properties of the syndrome (base) trellis then we turn to trellis-oriented form before finally discussing the Viterbi algorithm. Many, but not all, of the results here have classical analogies, and we attempt to provide original citations to the best of our knowledge for those which are not typically discussed in textbooks. Classically, however, codes are vector spaces whereas here we consider stabilizer codes as groups. This changes the techniques, but we try to maintain the same overall direction of the proofs if possible. We consider it a strength of the theory that a single framework can handle both classical and quantum trellises. The closest classical work to this is that of Forney and Trott on convolutional group codes [177].

2.3.1 Syndrome Trellis

The first result was understood and implicitly used in [167] and [170] but was never explicitly stated. It is perhaps obvious, but we include a proof here for completeness.

Proposition 2.3.1

There is a one-to-one correspondence between elements in $\mathcal{S}^{\perp}$ and length-n paths in the trellis.

Proof. It is clear that by construction every element in $\mathcal{S}^{\perp}$ corresponds to a length-n path in the trellis. It remains to show that every length-n path in the trellis corresponds to an

element in $\mathcal{S}^\perp$ and that all such paths are unique. Let $(\overline{0}, P_1, \sigma_1(P_1 I \dots I)) \in E_1$ be an edge with label P_1, then pick an arbitrary edge $(\sigma_1(P_1 I \dots I), P_2, \sigma_2(P_1 P_2 I \dots I)) \in E_2$. Continuing this process, we end with an edge $(\sigma_{n-1}(P_1 \dots P_{n-1} I), P_n, \sigma_n(P_1 \dots P_n)) \in E_n$. This last terminus is the zero syndrome by construction, hence, concatenating the edge labels in the path, the element $P_1 P_2 \dots P_n$ has zero syndrome and is therefore an element of $\mathcal{S}^\perp$ by definition. Since each vertex can only have a unique outgoing edge with a given label, this string uniquely identifies a path in the trellis and no other path can have the same label. □

Corollary 2.3.2

Let $v_i \in V_i$ and $v_{i+1} \in V_{i+1}$ be arbitrary. If there is an element $P_i \in \mathcal{P}_1$, which takes v_i to v_{i+1}, then $(v_i, P_i, v_{i+1}) \in E_i$.

Proof. Let $v_i \in V_i$, $v_{i+1} \in V_{i+1}$, and suppose concatenating a Pauli element P_i to a path with terminus v_i changes the syndrome of v_i to the syndrome of v_{i+1}. Let $\overline{V_0 v_i}$ and $\overline{v_{i+1} V_n}$ be a path from V_0 to v_i and v_{i+1} to V_n, respectively. Then the element represented by the path $\overline{V_0 v_i} P_i \overline{v_{i+1} V_n}$ has a length-n and zero syndrome, and is hence an element of $\mathcal{S}^\perp$. The result then follows from the previous proposition. □

In the previous section we attempted to stick to the notation of the preceding literature, following [167]. At this point, we make some adjustments which we find necessary to simplify the proofs and discussion. The main reason for this is that S and $\mathcal{S}^\perp$ are groups whereas V_i and E_i are vector spaces. Mapping elements of $\mathcal{P}_n$ to vectors or the vector spaces to groups provides for a more coherent argument. Choosing groups, we henceforth introduce the following notation. We begin with a rather general definition for completeness; however, our goal and use case throughout this paper is the rather natural splitting of the qudit indices into a “past" and “future", which is defined later in Definition 2.3.4.

Definition 2.3.3

- *Let $\mathcal{J} \subset \{0, \dots, n\}$ be an index set and define $\mathcal{A}_\mathcal{J}$ to be the set of elements in a*

subgroup $\mathcal{A} \leq \mathcal{P}_n$ whose Pauli operators are equal to the identity on the complement of $\mathcal{J}$, $\mathcal{J}^c \coloneqq \{0, \ldots, n\} \setminus \mathcal{J}$.

- *Denote by $\mathcal{A}_{|_\mathcal{J}}$ the set of elements in $\mathcal{P}_1^{|\mathcal{J}|}$ constructed from elements in $\mathcal{A}$ whose elements at indices $\mathcal{J}^c$ have been deleted. Alternatively, $\mathcal{A}_{|_\mathcal{J}}$ is the image of $\mathcal{A}$ whose elements at indices $\mathcal{J}^c$ have been set to the identity.*

The alternative definition of $\mathcal{A}_{|_\mathcal{J}}$ is useful to keep all elements the original length, whereas in the first definition the strings are shortened to length $|\mathcal{J}|$. Both definitions are conceptually equivalent.

Remark: If $\mathcal{C}$ is a classical code, then $\mathcal{C}_\mathcal{J}$ and $\mathcal{C}_{|_\mathcal{J}}$ are the shortened and punctured codes of $\mathcal{C}$ with respect to $\mathcal{J}$, respectively. The concepts of shortening and puncturing are a bit more complicated for quantum codes [140], so we refrain from using this terminology here.

Example 1. Let $\mathcal{J} = \{0, \ldots, i\}$. Then $\mathcal{A}_\mathcal{J}$ is the set of elements which naturally have identity elements in indices $\{i+1, \ldots, n\}$ and $\mathcal{A}_{|_\mathcal{J}} = \operatorname{im} \pi_\mathcal{J}(\mathcal{A})$ is the set of elements whose elements in indices $\{i+1, \ldots, n\}$ have been projected to the identity regardless of their initial value.

Critical to our proofs is the fact that since $\pi_\mathcal{J}(PP') = \pi_\mathcal{J}(P)\pi_\mathcal{J}(P')$ for $P, P' \in \mathcal{P}_n$, $\pi_\mathcal{J}$ is a group homomorphism. Then $\ker \pi_\mathcal{J}(\mathcal{A}) = \mathcal{A}_{|_{\mathcal{J}^c}}$ and $\ker \pi_{\mathcal{J}^c}(\mathcal{A}) = \mathcal{A}_{|_\mathcal{J}}$, and $\mathcal{A}_{|_{\mathcal{J}^c}} \trianglelefteq \mathcal{A}$ and $\mathcal{A}_{|_\mathcal{J}} \trianglelefteq \mathcal{A}$ as all kernels are normal. The product group $\mathcal{A}_{|_\mathcal{J}}\mathcal{A}_{|_{\mathcal{J}^c}}$ is also normal, and as $\mathcal{J} \cap \mathcal{J}^c = \varnothing$, $\mathcal{A}_{|_\mathcal{J}}\mathcal{A}_{|_{\mathcal{J}^c}} = \mathcal{A}_{|_\mathcal{J}} \times \mathcal{A}_{|_{\mathcal{J}^c}} \trianglelefteq \mathcal{A}$.

In keeping with the standard trellis literature, we introduce the following further terminology.

Definition 2.3.4

Fix an index $i \in \{0, \ldots, n\}$. Then the past and future, with respect to i, are defined to be the index sets $\mathfrak{p}_i \coloneqq \{0, \ldots, i\}$ and $\mathfrak{f}_i \coloneqq \{i+1, \ldots, n\}$, respectively.

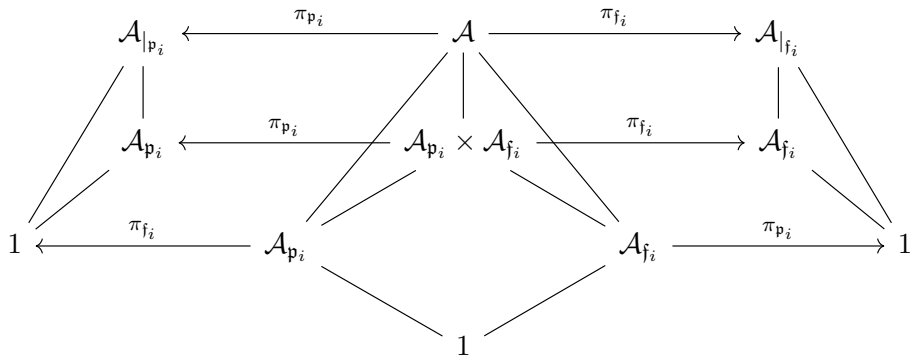


Figure 2.3: A summary of the relationships between the groups and the maps (Figure 5 of [177]). A line without an arrow between two groups means the group in the lower level is a subgroup of the upper level.

In particular, we will utilize the sets $\mathcal{S}_{\mathfrak{p}_i}$, $\mathcal{S}_{\mathfrak{f}_i}$, $S_{|\mathfrak{p}_i}$, $S_{|\mathfrak{f}_i}$, and likewise for $\mathcal{S}^\perp$. See Figure 2.3 for a summary of the relationships between the groups. Note that the literature varies on which set includes the index i. In this work we index the qudits on $\{1, \ldots, n\}$ but it is necessary to include zero in order to project to the identity.

The previously mentioned subgroups (Figure 2.3) depend only on the set $\mathcal{A}$ and not the choice of generators, although $\mathcal{S}$ and $\mathcal{S}^\perp$ may have different subgroups at the same index. Lagrange's theorem restricts their cardinality to powers of p. (In fact, this was a motivation for promoting V and E to groups instead of dealing with less-constrained vector subspace dimensions.) Initially, $\mathcal{A}_{\mathfrak{f}_0} = \mathcal{A}$ and monotonically decreases in size as i goes to n, $\mathcal{A}_{\mathfrak{f}_n} = \{I^{\otimes n}\}$. Likewise, $\mathcal{A}_{\mathfrak{p}_0} = \{I^{\otimes n}\}$ and monotonically increases in size as i goes to n, $\mathcal{A}_{\mathfrak{p}_n} = \mathcal{A}$.

For a fixed i, sort the elements of $\mathcal{S}^\perp$ into sets $\mathcal{S}^\perp_{\mathfrak{p}_i}$, $\mathcal{S}^\perp_{\mathfrak{f}_i}$, and $\mathcal{S}^\perp_{\mathfrak{a}_i}$, where the "active" set, denoted by $\mathfrak{a}$, contains the remaining elements neither wholly in the past nor future. Include the identity in $\mathcal{S}^\perp_{\mathfrak{a}_i}$ to give these trivial intersection. An element of $\mathcal{S}^\perp$ may hence be decomposed in the form $P_\mathfrak{p} P_\mathfrak{a} P_\mathfrak{f}$, where $P_\mathfrak{p} \in \mathcal{S}^\perp_{\mathfrak{p}_i}$, $P_\mathfrak{a} \in \mathcal{S}^\perp_{\mathfrak{a}_i}$, and $P_\mathfrak{f} \in \mathcal{S}^\perp_{\mathfrak{f}_i}$. An element $P_{|\mathfrak{p}_i} \in \mathcal{S}^\perp_{|\mathfrak{p}_i}$ is of the form $P_{|\mathfrak{p}_i} = (P_{\mathfrak{p}_i})_{|\mathfrak{p}_i} (P_{\mathfrak{a}_i})_{|\mathfrak{p}_i}$, and similarly for the future. Thus $\mathcal{S}^\perp_{|\mathfrak{p}_i} = \text{span}\left\{\mathcal{S}^\perp_{\mathfrak{p}_i}, \left(\mathcal{S}^\perp_{\mathfrak{a}_i}\right)_{|\mathfrak{p}_i}\right\}$ and $\mathcal{S}^\perp_{|\mathfrak{f}_i} = \text{span}\left\{\mathcal{S}^\perp_{\mathfrak{f}_i}, \left(\mathcal{S}^\perp_{\mathfrak{a}_i}\right)_{|\mathfrak{f}_i}\right\}$. Fix a $P \in \mathcal{S}^\perp_{\mathfrak{a}_i}$ and let v be the vertex generated by P in V_i. Then every length-i path from V_0 to v is generated by the coset

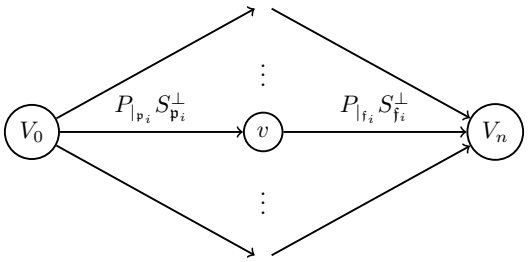


Figure 2.4: Grouping all paths from V_0 to v into a single line, the trellis may be seen as the depicted set of cosets. As is clear in the diagram, each vertex has a unique past and future.

$P_{|\mathfrak{p}_i}\mathcal{S}^{\perp}_{\mathfrak{p}_i}$. Likewise, every path from v to V_n is generated by an element of the coset $P_{|\mathfrak{f}_i}\mathcal{S}^{\perp}_{\mathfrak{f}_i}$. Putting these together, every path from V_0 to V_n may be viewed as a coset $P_{|\mathfrak{p}_i}\mathcal{S}^{\perp}_{\mathfrak{p}_i}P_{|\mathfrak{f}_i}\mathcal{S}^{\perp}_{\mathfrak{f}_i}$ with respect to v. See Figure 2.4 for a visual summary.

As mentioned in the previous section, there are historically numerous definitions of trellises in the classical literature. It was therefore important to determine whether some are "better" than others, leading to the concept of a "minimal" trellis (defined below). We will see below that if another trellis would be defined for stabilizer codes, the trellis of this work is minimal. The following proposition takes us in this direction and holds for any definition of a trellis for a stabilizer code. To connect with the literature, we define the dimension of a group to be the minimum number of generators in any generating set. This terminology is induced by considering a basis under the map $\mathcal{P}_n \to \mathbb{F}_{p^2}^n$.

Proposition 2.3.5 (Lemma 1 [167])

Given a stabilizer code $\mathcal{S}$, any trellis for $\mathcal{S}$ must satisfy

$$|V_i| \geq p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_i} - \dim \mathcal{S}^{\perp}_{\mathfrak{f}_i}},$$
$$|E_i| \geq p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}} - \dim \mathcal{S}^{\perp}_{\mathfrak{f}_i}}.$$

Proof. Every path from V_0 to V_n must go through some vertex $v \in V_i$. The set of all elements in $\mathcal{S}^{\perp}$ which map to the vertex v under σ_i is a subset of the cosets described in Figure 2.4 and hence has cardinality bounded above by $|\mathcal{S}^{\perp}_{\mathfrak{p}_i}||\mathcal{S}^{\perp}_{\mathfrak{f}_i}|$. The trellis can be written

as a union of paths passing through v over all $v \in V_i$, so the number of such cosets is $|V_i|$. Using Proposition 2.3.1, we then have

$$|\mathcal{S}^{\perp}| \leq |V_i||\mathcal{S}^{\perp}_{\mathfrak{p}_i}||\mathcal{S}^{\perp}_{\mathfrak{f}_i}|.$$

Likewise, every path in the trellis contains one edge in section E_i. Since edges are of the form $(s(e), P, t(e))$, a similar argument gives

$$|\mathcal{S}^{\perp}| \leq |\mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}}||E_i||\mathcal{S}^{\perp}_{\mathfrak{f}_i}|.$$

□

Definition 2.3.6

A trellis that meets the lower bounds of Proposition 2.3.5 is minimal.

Remark: Non-minimal trellises will not be discussed in this work. As a trivial example, one could define a trellis where each element of $\mathcal{S}^{\perp}$ consists of its own path from V_0 to V_n without intersecting any other path. The number of vertices at each depth would then be $|\mathcal{S}^{\perp}|$. We refer the reader to [175] for further (classical) examples. The concept of a minimal trellis comes from [174].

While the edges of the trellis encode $\mathcal{S}^{\perp}$, the vertices do not take the logical operators into account and are determined by $\mathcal{S}$, an important point in sharp contrast with the classical case whose vertices and edges are both constructed relative to the same object: the code.

For the vertices, fix i, let P be an arbitrary element of $\mathcal{S}^{\perp}$, and let S_j be a fixed generator of $\mathcal{S}$. If $S_j \in \mathcal{S}_{\mathfrak{p}_i}$ then $\langle S_j, \pi_{\{0,\ldots,i\}}(P)\rangle = 0$ since S_j is the identity at indices $\{i+1, \ldots, n\}$ and these will commute with any elements of P at these positions. Thus, the syndrome of P with respect to the generator S_j has already been completely determined (and is zero

since $P \in \mathcal{S}^{\perp}$). Likewise, if $S_j \in \mathcal{S}_{\mathfrak{f}_i}$ then $\langle S_j, \pi_{\{0,\ldots,i\}}(P)\rangle = 0$ since S_j is the identity at indices $\{1,\ldots,i\}$, which, again, commute with $\pi_{\{0,\ldots,i\}}(P)$. Only generators $S_j \in \mathcal{S}_{\mathfrak{a}_i}$ can have nonzero syndromes.

Theorem 2.3.7 (Quantum Space Theorem(s))

The syndrome trellis of Section 2.2 is minimal, i.e.,

(i) (Quantum State Space Theorem, Lemma 2 [167])

$$|V_i| = p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} - \dim \mathcal{S}_{\mathfrak{f}_i}^{\perp}} \tag{2.4}$$

(ii) (Quantum Branch Space Theorem)

$$|E_i| = p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp} - \dim \mathcal{S}_{\mathfrak{f}_i}^{\perp}} \tag{2.5}$$

Proof. Let i be fixed.

(i) The vertices are the image of $\mathcal{S}^{\perp}$ under σ_i. By the above comments, the past and the future have zero syndrome; therefore, $V_i = \sigma_i(\mathcal{S}^{\perp}) \cong \mathcal{S}^{\perp}/\mathcal{S}_{\mathfrak{p}_i}^{\perp} \times \mathcal{S}_{\mathfrak{f}_i}^{\perp}$ by the first isomorphism theorem for groups.

(ii) The kernel of the map from $\mathcal{S}^{\perp}$ to E_i consists of the elements which map to $(\overline{0}, I, \overline{0})$, i.e.,

$$\begin{aligned}
&\ker \sigma_{i-1}(\mathcal{S}^{\perp}) \cap \{P \in \mathcal{S}^{\perp} \mid P_i = I\} \cap \ker \sigma_i(\mathcal{S}^{\perp}) \\
&= \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp} \times \mathcal{S}_{\mathfrak{f}_{i-1}}^{\perp} \cap \{P \in \mathcal{S}^{\perp} \mid P_i = I\} \cap \mathcal{S}_{\mathfrak{p}_i}^{\perp} \times \mathcal{S}_{\mathfrak{f}_i}^{\perp} \\
&= \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp} \times \mathcal{S}_{\mathfrak{f}_i}^{\perp}.
\end{aligned}$$

□

Remark: The sequences $\{|V_i|\}_{i=0}^{n}$ and $\{|E_i|\}_{i=1}^{n}$ are often referred to as the state space and branch space complexity profiles, respectively. The quantity $\max_i \dim V_i$ is the state

complexity, $\max_i \dim E_i$ the branch complexity, and $|E|$ the edge complexity. Computing the branch complexity for a given family of codes is a major theme of the classical trellis literature. We will not utilize these in this work.

Lemma 2.3.8

The set of edges in E_i with terminus vertex $\overline{0}$ is a subgroup of E_i.

Proof. The proof of Theorem 2.3.7 shows that E_i is a group. Clearly the identity edge exists and closure follows since if $(s_1, P_1, \overline{0}), (s_2, P_2, \overline{0}) \in E_i$, then $(s_1, P_1, \overline{0})(s_2, P_2, \overline{0}) = (s_1 + s_2, P_1P_2, \overline{0}) \in E_i$. It remains to show that E_0 contains inverses. Let $(a, b, \overline{0}) \in E_0$ and (c, d, e) be its inverse in E_i. The group operation on E_i forces $e = \overline{0}$. □

Corollary 2.3.9

For $1 \leq i \leq n$, every vertex $v \in V_i$ has incoming degree

$$\deg_{in}(v) = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_i} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}}}, \tag{2.6}$$

and for $0 \leq i \leq n-1$, every vertex $v \in V_i$ has outgoing degree

$$\deg_{out}(v) = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{f}_i} - \dim \mathcal{S}^{\perp}_{\mathfrak{f}_{i+1}}}. \tag{2.7}$$

Proof. Let $v \in V_i$ and denote by $E_{\text{in}}(v)$ the set of edges with terminus v, $E_{\text{in}}(v) = \{e \in E_i \mid t(e) = v\}$. By definition, $\deg_{\text{in}}(v) = |E_{\text{in}}(v)|$. By Lemma 2.3.8, $E_{\text{in}}(v)$ is a coset of $E_{\text{in}}(\overline{0})$ in E_i. In particular, $|E_{\text{in}}(v)| = |E_{\text{in}}(0)|$. Since there are $|E_i|$ total edges equally divided among $|V_i|$ vertices, $\deg_{\text{in}}(v) = |E_i|/|V_i|$. The result follows from Theorem 2.3.7. An identical proof gives the second equality with $\deg_{\text{out}}(v) = |E_{i+1}|/|V_i|$. □

Lemma 2.3.10

Let $\mathcal{A} = S$ or $\mathcal{S}^{\perp}$. Then $0 \leq \dim \mathcal{A}_{\mathfrak{f}_i} - \dim \mathcal{A}_{\mathfrak{f}_{i+1}} \leq 2$ for $0 \leq i \leq n-1$ and $0 \leq \dim \mathcal{A}_{\mathfrak{p}_i} - \dim \mathcal{A}_{\mathfrak{p}_{i-1}} \leq 2$ for $1 \leq i \leq n$.

Proof. We show the result for the first inequality and the remaining proof is identical. An easy counting argument shows that the number of I's in the ith column of $\mathcal{A}_{\mathfrak{f}_i}$ are either $|\mathcal{A}_{\mathfrak{f}_i}|$, $|\mathcal{A}_{\mathfrak{f}_i}|/p$, or $|\mathcal{A}_{\mathfrak{f}_i}|/p^2$. (If there are not all I's, then pair any element with its inverse to create an element with an I. Likewise, pair any element without an I with an element with an I. Do the same for X and Z combinations.) These are the elements which are also in $\mathcal{A}_{\mathfrak{f}_{i+1}}$, so $|\mathcal{A}_{\mathfrak{f}_{i+1}}| \geq |\mathcal{A}_{\mathfrak{f}_i}|/p^2$. The dimension change is therefore no more than two. □

Corollary 2.3.11

For $0 \leq i \leq n-1$, $\deg_{out}(v) \in \{1, p, p^2\}$, *and for* $1 \leq i \leq n$, $\deg_{in}(v) \in \{1, p, p^2\}$. *Equivalently, for* $1 \leq i \leq n$, $\dim V_i \leq \dim E_i \leq \dim V_i + 2$.

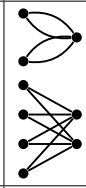
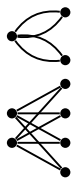
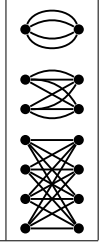
Table 2.1 records this result graphically for the special case $p = 2$. It is well-known in classical trellis theory that the dimension change in Lemma 2.3.10 is bounded by one [178]. This is due to classical codes only having one symbol alphabet instead of the two, X, Z, which allows one to completely row reduce to a single pivot per column. Any configuration in Table 2.1 with a two is therefore unique to the quantum setting. The next theorem shows that these are the only possible configurations. No corresponding proof exists in the classical literature, but the proof here also holds in the classical case with minor modifications.

Theorem 2.3.12

Consider an arbitrary edge $e = (s(e), P_i, t(e)) \in E_i$ *and define* $\mathcal{I}_i = \{v \in V_i \mid \exists e' \in E_i, t(e') = t(e)\}$ *and* $\mathcal{I}_{i+1} = \{v \in V_{i+1} \mid \exists e' \in E_i, s(e') = s(e)\}$. *Then the vertices of* $\mathcal{I}_i$ *and* $\mathcal{I}_{i+1}$ *form a completely-connected bipartite graph and no other elements of* $V_i \backslash \mathcal{I}_i$ *or* $V_{i+1} \backslash \mathcal{I}_{i+1}$ *are connected to the vertices in* $\mathcal{I}_{i+1}$ *and* $\mathcal{I}_i$, *respectively. If there exists a parallel edge in* E_i *then all edges in* E_i *are parallel with the same number of edges in parallel.*

Proof. From Corollary 2.3.2 and Lemma 2.3.8, it suffices to show this for the edge $(\overline{0}, 0, \overline{0}) \in E_i$, as all edge configurations are a shift of this one. Choose an arbitrary $v_i \in \mathcal{I}_i$ and $v_{i+1} \in \mathcal{I}_{i+1}$. By definition there exists Pauli labels P_s and P_t such that $(v_i, P_s, \overline{0})$, $(\overline{0}, P_t, v_{i+1}) \in$

Table 2.1: The various possible edge configurations for $p = 2$. Since $\Delta\dim \mathcal{S}^{\perp}_{\mathfrak{p}_i}$ and $\Delta\dim \mathcal{S}^{\perp}_{\mathfrak{f}_i}$ are bounded by one in classical theory, any configurations with a two is unique to the quantum setting.

$\Delta \dim S^{\perp}_{\mathfrak{p}_i}$	$\Delta \dim S^{\perp}_{\mathfrak{f}_i}$	E_i	$\Delta \dim S^{\perp}_{\mathfrak{p}_i}$	$\Delta \dim S^{\perp}_{\mathfrak{f}_i}$	E_i	$\Delta \dim S^{\perp}_{\mathfrak{p}_i}$	$\Delta \dim S^{\perp}_{\mathfrak{f}_i}$	E_i
0	0		1	0		2	0	
0	1		1	1		2	1	
0	2		1	2		2	2	

E_i. Hence, $(v_i, P_sP_t, v_{i+1}) \in E_i$. This works for any pair of vertices in $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$, proving the first statement. Now suppose, without loss of generality, there exists a $v'_{i+1} \in V_{i+1}\backslash\mathcal{I}_{i+1}$ connected to a $v_i \in \mathcal{I}_i$ via (v_i, P', v'_{i+1}) but is not part of the bipartite graph. As a subgroup, the inverse syndrome to v_i, v_i^{-1}, exists with some edge label P_i. Then $(v_i^{-1}, P_i, \overline{0})(v_i, P', v'_{i+1}) = (\overline{0}, P_iP', v'_{i+1}) \in E_i$, a contradiction to the fact that $v'_{i+1} \notin \mathcal{I}_{i+1}$.

To prove the last statement, pick a source/terminus pair $v_i \in V_i$ and $v_{i+1} \in V_{i+1}$ with parallel edges uniquely labeled by $\{P_1, \ldots, P_k\}$. We will show that every other edge must also have k parallel edges; hence it suffices, without loss of generality, to assume that both syndromes are $\overline{0}$. Now choose any other edge $(v'_s, P', v'_t) \in E_i$ with v'_s and v'_t not both $\overline{0}$. By Lemma 2.3.8, $(\overline{0}, P_1^{-1}, \overline{0})$ exists and $(v'_s, P', v'_t)(\overline{0}, P_1^{-1}, \overline{0})(\overline{0}, P_j, \overline{0}) = (v'_s, P'P_1^{-1}P_j, v'_t)$ for some $1 < j \leq k$. Since $P'P_1^{-1}P_{j_1} = P'P_1^{-1}P_{j_2}$ implies $P_{j_1} = P_{j_2}$, a contradiction, there are $k-1$ additional parallel edges between v'_s and v'_t of this form. Since the same idea works when P_1^{-1} is replaced with any other of the k labels, the number of parallel edges from v'_s to v'_t is greater than or equal to k. Running the argument backwards shows that for each edge between v'_s to v'_t there is a corresponding edge from $\overline{0}$ to $\overline{0}$. Since the number of

these is k, we have that the number of parallel edges from v'_s to v'_t is exactly k. □

Corollary 2.3.13

The number of disjoint edge configurations in E_i is $p^{\dim \mathcal{S}^\perp - \dim \mathcal{S}^\perp_{\mathfrak{p}_{i+1}} - \dim \mathcal{S}^\perp_{\mathfrak{f}_i}}$.

This is simply $|V_{i+1}|/\deg_{\text{out}}(v)$ for $v \in V_i$.

At first glance, Equation (2.4) should be viewed with skepticism. It is common for $\dim \mathcal{S}_{\mathfrak{p}_i} = \dim \mathcal{S}_{\mathfrak{f}_i} = 0$ (see Example 4) and if the same holds true for $\mathcal{S}^\perp$, then $|V_i| = p^{n+k}$ which is notably greater than the number of total possible syndromes, p^{n-k}. Applying the classical formula for V_i or using the logic above, one would expect $\dim V_i = \dim \mathcal{S} - \dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{f}_i}$, and in fact this also works. The inclusion of the logicals therefore always forces $\dim \mathcal{S}^\perp_{\mathfrak{p}_i} + \dim \mathcal{S}^\perp_{\mathfrak{f}_i} \geq 2k$. The various formulas throughout this subsection may therefore be written with this alternative view of $\dim V_i$, but this leads to a lack of cancellation of terms and an unpleasant factor of $2k$ floating around whose lack of obvious effect on the results requires justification.

Corollary 2.3.14

For $0 \leq i \leq n$, $\dim \mathcal{S}^\perp_{\mathfrak{p}_i} + \dim \mathcal{S}^\perp_{\mathfrak{f}_i} \geq 2k$.

Proof. Starting with $\dim \mathcal{S}^\perp_{\mathfrak{f}_0} = n+k$ and repeatedly applying Lemma 2.3.10 gives $\dim \mathcal{S}^\perp_{\mathfrak{f}_i} \geq n + k - 2i$ and, likewise, $\dim \mathcal{S}^\perp_{\mathfrak{p}_i} \geq -n + k + 2i$. Combining these two equations gives the desired result. □

The same proof applied to $\mathcal{S}$ instead of $\mathcal{S}^\perp$ shows $\dim \mathcal{S}_{\mathfrak{p}_i} + \dim \mathcal{S}_{\mathfrak{f}_i} \geq 0$.

The next result says that for a fixed qudit ordering the choice of stabilizer or logical generators is irrelevant.

Lemma 2.3.15

Any two minimal trellises for the same stabilizer code are isomorphic.

Proof. Since $\dim \mathcal{S}^\perp_{\mathfrak{p}}$ and $\dim \mathcal{S}^\perp_{\mathfrak{f}}$ are invariant, we know from Corollary 2.3.9 that any trellises satisfying Theorem 2.3.7 have the same vertex degrees and are hence isomorphic.

It remains to specify the mapping. Let there be two minimal trellises for the code with sets V, E and V', E', respectively. Fix a $v \in V_i$ and consider its past coset (see Figure 2.4). Pick an element of $\mathcal{S}^{\perp}$ in this coset and determine its terminus $v' \in V_i'$. The map $f : V \to V'$, $f(v) = v'$ is clearly an isomorphism. It immediately follows that the edge isomorphism is given by $(s(e), P_i, t(e)) \mapsto (f(s(e)), P_i, f(t(e)))$. □

2.3.2 Trellis-Oriented Form

The term “trellis-oriented" was introduced in passing in an appendix by Forney [12] and wasn't thoroughly defined until seven years later by Kschischang and Sorokine [179]. Here we adapt the latter approach to the quantum setting, as did [167], but we will stick closer to the classical notation than that of [167]. The term “minimal-span" is preferred by some authors in the literature.

Let $P \in \mathcal{P}_n$ be arbitrary with components P_i at index i.

Definition 2.3.16

(i) The left index of P, $L(P)$, is the smallest index such that $P_i \neq I$ and the right index of P, $R(P)$, is the largest index such that $P_i \neq I$.

(ii) The span of P is the index set $\{L(P), L(P)+1, \ldots, R(P)\}$ and the span length of P is the cardinality of the span of P, $R(P) - L(P) + 1$. The span of the identity string is defined to be $\{\,\}$ and the corresponding span length to be 0.

(iii) Say P is active at depth i if $i-1$ and i are in the span of P, i.e., $L(P) \leq i-1$ and $R(P) \geq i$.

Definition 2.3.17 (Left-Right Property)

A set of elements of $\mathcal{P}_n$ is said to have the left-right property if no two elements with the same left or right index have more than one $X(a)$ or $Z(b)$ at these index locations.

The next definition is intended to be applied to a set of generators and not the entirety of its span.

Definition 2.3.18 (Trellis-Oriented Form)

A set of $\mathcal{P}_n$ is said to be in trellis-oriented form (TOF) if the sum of the span lengths of the elements is as small as possible.

Proposition 2.3.19

A set of elements $\mathcal{P}_n$ is in TOF if and only if it has the left-right property.

Proof. Suppose a set of elements $\mathcal{P}_n$ is in TOF but does not have the left-right property. Then there exists two elements of the set, $\tilde{P}$ and $\tilde{P}'$ such that $L(\tilde{P}) = L(\tilde{P}')$ or $R(\tilde{P}) = R(\tilde{P}')$. Without loss of generality, assume $L(\tilde{P}) = L(\tilde{P}')$ and $R(\tilde{P}) > R(\tilde{P}')$. One can always replace any $X(a)$ or $Z(b)$ with $X(1)$ and $Z(1)$ by repeatedly applying it to itself since p is prime and any power generates the cyclic group $\langle p \rangle$. Assume this is done for $\tilde{P}$ and replace the X or Z in $\tilde{P}'$ by $X(p-1)$ or $Z(p-1)$, $\tilde{P} \mapsto P$ and $\tilde{P}' \mapsto P'$. Then PP' has lower span length than $\tilde{P}$, a contradiction to the fact that the set is in TOF.

Now suppose the set has the left-right property but is not in TOF. The only way to reduce the span length of the set is to increase a left index or decrease a right index. But this is impossible since there is only a single X or Z at these indices by the left-right property. Hence the set already has the lowest total span length. □

Example 2. The generators for the rotated surface and color codes naturally have the left-right property. The canonical form of the stabilizer generators of the $[[5, 1, 3]]$ code as cyclic shifts of $[1, \alpha, \alpha, 1, 0]$ may be put into TOF as follows:

$$\begin{array}{l} S_1 = [1, \alpha, \alpha, 1, 0] \\ S_2 = [0, 1, \alpha, \alpha, 1] \\ S_3 = [1, 0, 1, \alpha, \alpha] \\ S_4 = [\alpha, 1, 0, 1, \alpha] \end{array} \quad \begin{array}{c} \\ \\ \mapsto \\ \\ \end{array} \quad \begin{array}{ll} S_1 & = [1, \alpha, \alpha, 1, 0] \\ S_4 & = [\alpha, 1, 0, 1, \alpha] \\ S_2 & = [0, 1, \alpha, \alpha, 1] \\ S_1S_3 & = [0, \alpha, \alpha^2, \alpha^2, \alpha] \end{array} \quad \begin{array}{c} \\ \\ \mapsto \\ \\ \end{array} \quad \begin{array}{ll} S_1 & = [1, \alpha, \alpha, 1, 0] \\ S_1S_3S_4 & = [\alpha, \alpha^2, \alpha^2, \alpha, 0] \\ S_2 & = [0, 1, \alpha, \alpha, 1] \\ S_1S_3 & = [0, \alpha, \alpha^2, \alpha^2, \alpha] \end{array}.$$

The left and right indices for each row are, in order, $\{[1, 4], [1, 4], [2, 5], [2, 5]\}$, where we have used the notation $[L(P), R(P)]$. It is customary, and sometimes included in the definition, to rearrange the strings by increasing left index when viewing them in a matrix-like

format, similar to a row-reduced matrix, but this is not strictly necessary.

The procedure used in the previous example is quite simple: after reducing the left side from top to bottom, the right side is reduced from bottom to top and the upper generator is always replaced instead of the lower. This prevents the combination of the generators from changing the left indices. The key to automating this is considering elements as length-n vectors over $\mathbb{F}_{p^2}$ and using appropriate operations in this field. This should be compared with the complex algorithm given in [170] for obtaining the TOF using the symplectic representation over $\mathbb{F}_p$.

The quantum TOF is more complicated than in classical theory. To gain some intuition for this consider a set of elements of $\mathcal{P}_n$ in TOF. A quantum index, either left or right, is of the form $X(a)Z(b)$, where $a, b \in \mathbb{F}_p$. Given two indices $X(a)Z(b)$ and $X(a')Z(b')$ with all integers nonzero, one can only eliminate the other to produce an identity if and only if they are scalar multiples of each other in $\mathbb{F}_p$, i.e., $(a', b') \in \langle (a, b) \rangle \in \mathbb{Z}_p \times \mathbb{Z}_p$. In the more general case, by repeated application of the string to itself, a and b can individually be made to take any value in $\mathbb{F}_p$. In particular, $-a' \in \langle a \rangle = Z_p$ since p is prime, and likewise for b. So, one of a' or b' may always be eliminated. An index $X(a)$ cannot eliminate an index $Z(b)$ and vice versa. To summarize, the possible elements at left or right index locations for qubit codes are $\{\{1, 0\}, \{\alpha, 0\}, \{\alpha^2, 0\}, \{1, \alpha\}, \{\alpha^2, 1\}, \{\alpha^2, \alpha\}\}$. This is in stark contrast to the classical case where a matrix can always be put into reduced-row echelon form.

Let P and P' be two elements of $\mathcal{P}_n$ with the left-right property and span $[L, R]$, then PP' also has span $[L, R]$. This is often referred to as the predictable span property. If P and P' have span $[L, R_1]$ and $[L, R_2]$, respectively, with $R_1 < R_2$, then PP' has span $[L, R_2]$, and likewise for $[L_1, R]$ and $[L_2, R]$. These two cases are not possible classically when there is only one copy of $\mathbb{F}_p$, necessitating different proof strategies.

Example 3. The first two stabilizers for the TOF of the following $[[8, 2, 3]]$ code [171]

have the same left index but different right indices:

$$\begin{array}{l}
[\alpha^2, \alpha, \alpha, \ \alpha, \ \ 1, \ \ 0, \ \ 0, \ \ 0] \\
[\alpha, \ \ 0, \ 0, \ \ 1, \ \ 0, \ \ \alpha^2, 0, \ \ 0] \\
[0, \ \ 1, \ \alpha, \ \alpha, \ \alpha^2, \ 1, \ \ \alpha^2, \ 0] \\
[0, \ \ \alpha, \alpha^2, 0, \ \ \alpha^2, \alpha^2, 1, \ \ 0] \\
[0, \ \ 0, \ 1, \ \ \alpha^2, 1, \ \ \alpha, \ \alpha^2, \alpha] \\
[0, \ \ 0, \ \alpha, \ \ 0, \ \ 0, \ \ 0, \ \ \alpha^2, 1]
\end{array} .$$

If we start with a generating set for $\mathcal{A}$, the resulting TOF still generates $\mathcal{A}$. Each generator has order p and generates $p-1$ nonidentity strings with the same span. Suppose two generators P and P' have spans $[L_1, R_1]$ and $[L_2, R_2]$, respectively, and both $R_1, R_2 \leq i \in \{1, \ldots, n\}$. Then all products of the p elements generated by P and the p elements generated by P' also have right index less than or equal to i. If there are no other generators with right index less than or equal to i, then these are all of the elements of $\mathcal{A}_{\mathfrak{p}_i}$. Extending this argument shows the following.

Proposition 2.3.20

Let $\mathcal{A}$ be a set of elements of $\mathcal{P}_n$ with generators $\mathcal{A}_i$ in TOF. Then

$$\dim \mathcal{A}_{\mathfrak{p}_i} = |\{\mathcal{A}_i \mid R(\mathcal{A}_i) \leq i\}| \tag{2.8}$$

$$\dim \mathcal{A}_{\mathfrak{f}_i} = |\{\mathcal{A}_i \mid L(\mathcal{A}_i) \geq i+1\}|. \tag{2.9}$$

Unfortunately, the previous argument also shows that the TOF is not unique since a generator may be replaced by any of its $p-1$ multiples. (In Example 2, replace S_1S_3 with, for example, $S_1S_2S_3$). However, in light in of Theorem 2.3.7 and Corollary 2.3.9, the entire structure of the trellis may be read-off directly from the generators of $\mathcal{S}^{\perp}$ when put into TOF. Proposition 2.3.20 also provides trivial proofs of results such as Lemma 2.3.10 by merely counting the number of possible new generators obtained when shifting indices in TOF.

Example 4. The stabilizers of the $[[5, 1, 3]]$ code appear in Example 2. The generators of $\mathcal{S}^{\perp}$ have a different TOF,

$$\begin{matrix} [1,\ \alpha^2, 1,\ \ 0,\ \ 0] \\ [\alpha, 1,\ \ \alpha,\ \ 0,\ \ 0] \\ [0,\ 1,\ \ \alpha^2, 1,\ \ 0] \\ [0,\ \alpha,\ \ 1,\ \ \alpha,\ \ 0] \\ [0,\ 0,\ \ 1,\ \ \alpha^2, 1] \\ [0,\ 0,\ \ \alpha,\ \ 1,\ \ \alpha] \end{matrix} .$$

Applying Proposition 2.3.20 to both sets we get Table 2.2.

Table 2.2: State and branch profiles for the $[[5, 1, 3]]$ code.

	$[[5, 1, 3]]$							
i	$\dim \mathcal{S}_{\mathfrak{p}_i}$	$\dim \mathcal{S}_{\mathfrak{f}_i}$	$\dim \mathcal{S}^{\perp}_{\mathfrak{p}_i}$	$\dim \mathcal{S}^{\perp}_{\mathfrak{f}_i}$	$\lvert V_i\rvert$	$\lvert E_i\rvert$	in	out
0	0	4	0	6	1	–	–	4
1	0	2	0	4	4	4	1	4
2	0	0	0	2	16	16	1	4
3	0	0	2	0	16	64	4	1
4	2	0	4	0	4	16	4	1
5	4	0	6	0	1	4	4	–

These may be compared with the trellis diagram for the code given in [170].

Remark: As discussed in the previous subsection, the structure of the trellis is invariant with respect to a change of stabilizers. One may compute a TOF of a set of generators, apply Proposition 2.3.20 to determine $|V_i|$, $|E_i|$, etc, and then construct the trellis with respect to the original set of stabilizers which, for example, may have been more beneficial experimentally.

2.3.3 The Viterbi Algorithm

It is worth clarifying exactly which decoding problem the trellis solves. Let $E \in \mathcal{P}_n$ be an error acting on a quantum stabilizer code $\mathcal{Q}$ with stabilizer group $\mathcal{S} = \langle S_1, \ldots, S_{n-k} \rangle$. The syndrome, $s = (s_1, \ldots, s_{n-k})$, is the ordered tuple of commutation relations of E with the

stabilizer elements, $s_j = \langle S_j, E \rangle$. The trellis solves Equation (1.48),

$$\hat{E}^{\mathrm{ML}}(s) = \underset{E \in \mathcal{P}_n}{\operatorname{argmax}} \Pr\left(L, S, T_s\right),$$

While this incorporates all of the information of the logical operators, it is technically not the degenerate decoder of (1.49). Pelchat and Poulin give a procedure for constructing a "degenerate" trellis to solve (1.49) for quantum convolutional codes [169]. We give an unrelated and different procedure in the next chapter based on the work developed here.

Proposition 2.3.21

The Viterbi algorithm solves (1.48) *for i.i.d. noise models.*

Proof. Let $\mu(v)$ be the value of the optimal (minimum weight) path at vertex v, initialize $\mu(\overline{0}) = 0$ at V_0, and denote by $\mathrm{wt}(e)$ the weight of edge e (see Algorithm 1). We proceed by induction on the depth i. By construction, all paths from V_0 to $v \in V_1$ consist of a single edge. The optimal path V_0 to v is given by the minimum weighted edge from V_0 to v. By Algorithm 1, $\mu(v) = \min_{\substack{e \in E_1 \\ t(e)=v}} \{\mu(\overline{0}) + \mathrm{wt}(e)\} = \mathrm{wt}(e)$ for the edge $e \in E_1$ of minimum weight, which is correct. Now assume the Viterbi algorithm computes the optimal path for all depths 1 to i and let $v \in V_{i+1}$. Then

$$\mu(v) = \min_{\substack{e \in E_i \\ t(e)=v}} \{\mu(s(e)) + \mathrm{wt}(e)\} = \min_{\substack{e \in E_i \\ t(e)=v \\ \text{path:} V_0 \to s(e)}} \{\mathrm{wt}(\text{path}) + \mathrm{wt}(e)\} = \min_{\substack{e \in E_i \\ t(e)=v \\ \text{path:} V_0 \to t(e)}} \{\mathrm{wt}(\text{path})\}.$$

Every path from V_0 to v is of this form. □

Remark: For independent X, Z, or depolarizing channels, the weights of the edges of the trellis are in correspondence with the Hamming weight of the vector, in which case the Viterbi algorithm returns the minimum Hamming weight correction. This is the same solution returned by algorithms like MWPM.

As mentioned in Section 2.2, the full decoding procedure consists of finding a pure error, shifting the trellis, and then applying the Viterbi algorithm. The shifting procedure requires at most n symplectic inner products followed by $|V| + |E|$ shifts. Fortunately, this operation is embarrassingly parallel, and the runtime may be significantly reduced depending on the implementation and available resources. For a potentially large computational speedup, one may skip shifting the vertices, as they play no role in the Viterbi algorithm. It is clear from Algorithm 1 that every edge incurs a single addition followed by incoming-degree comparisons for every vertex. For large codes, the vertices of V_i may be processed independently in parallel. The Viterbi algorithm may also be run simultaneously in a "forward pass" from V_0 to some V_i and a "backwards pass" from V_n to V_{i+1}. The choice of the optimal splitting depends highly on the left-right balance of the trellis. Further optimizations such as those based on the coset structure of the code (vertical symmetry) are generally code specific. In practice, the shifting and decoding can proceed one after the other inside a given section before moving onto the next, but for the sake of theoretical fundamentals we momentarily assume they are separate.

Theorem 2.3.22 (Theorem 2.10 [175])

The Viterbi algorithm requires $\Theta(|E|)$ arithmetic operations.

Proof. It suffices to characterize the operations of line 14 in Algorithm 1. For each vertex $v \in V_i$ for $1 \leq i \leq n$, $\deg_{\text{in}}(v)$ additions are preformed:

$$\text{total additions} = \sum_{i=1}^{n} \sum_{v \in V_i} \deg_{\text{in}}(v).$$

For simplicity we take a computational model where $x - 1$ minimums are computed for a list of size x. From this we then have

$$\text{total minimums} = \sum_{i=1}^{n} \sum_{v \in V_i} \deg_{\text{in}}(v) - \sum_{i=1}^{n} \sum_{v \in V_i} 1.$$

The first term in this sum is equal to the total number of edges, $|E|$, and the second term is total number of vertices minus V_0, $|V| - 1$:

$$\text{total additions} = |E|$$
$$\text{total minimums} = |E| - |V| + 1.$$

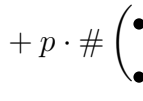
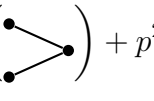
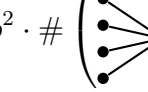
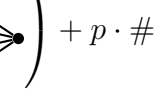
Therefore,

$$\text{total number of arithmetic operations} = \text{total additions} + \text{total minimums} = 2|E| - |V| + 1.$$

Since the trellis is connected, $|E| - |V| + 1 \geq 0$, so $2|E| - |V| + 1 \geq |E|$. The total number of arithmetic operations is hence upper bounded by $2|E|$ and lower bounded by $|E|$. $\square$

Theorem 2.3.7 shows that the syndrome trellis minimizes $|V_i|$ and $|E_i|$, but in light of the previous proof it would be useful to show that it also minimizes the quantity $|E| - |V| + 1$. That is, the trellis minimizes the number of algorithmic operations. Classically, Muder first showed that the analogous trellis minimizes $|V|$ [174], McEliece showed it minimizes $|E|$ [180, 175], and Vardy and Kschischang showed it minimizes $|E| - |V| + 1$ [181]. The proof of this follows for the quantum case with minor modifications (Proposition 2.3.25) and we include it here in Theorem 2.3.26 for completeness. We first establish a geometric interpretation of this quantity.

The quantity $|E|$ may be further resolved using the edge configurations in Table 2.1. While these are drawn for the special case $p = 2$, here we will use them to represent the corresponding diagrams for higher p. One may check that substituting $p = 2$ produces the correct coefficients in the formula below. We have,

$$|E| = 1 \cdot \#\left(\text{}\right) + p \cdot \#\left(\text{  }\right) + p^2 \cdot \#\left(\text{  }\right) + p \cdot \#\left(\text{  }\right) + p \cdot \#\left(\text{  }\right)$$

$$+ p^2 \cdot \#\left(\right) + p^2 \cdot \#\left(\right) + p^3 \cdot \#\left(\right) + p^2 \cdot \#\left(\right) + p^2 \cdot \#\left(\right)$$

(2.10)

$$+ p^3 \cdot \#\left(\right) + p^2 \cdot \#\left(\right) + p^3 \cdot \#\left(\right) + p^4 \cdot \#\left(\right).$$

Similarly, ignoring V_0 and counting the right hand sides,

$$|V| - 1 = 1 \cdot \#\left(\right) + 1 \cdot \#\left(\right) + 1 \cdot \#\left(\right) + p \cdot \#\left(\right) + 1 \cdot \#\left(\right)$$
$$+ p \cdot \#\left(\right) + p \cdot \#\left(\right) + p^2 \cdot \#\left(\right) + p^2 \cdot \#\left(\right) + 1 \cdot \#\left(\right)$$

(2.11)

$$+ p \cdot \#\left(\right) + 1 \cdot \#\left(\right) + p \cdot \#\left(\right) + p^2 \cdot \#\left(\right).$$

Definition 2.3.23

A vertex v with $\deg_{out}(v) > 1$ is called an expansion and a merger if $\deg_{in}(v) > 1$.

The number of mergers is graphically given by

$$\mathcal{M} = (p-1) \cdot \left(\right) + (p^2-1) \cdot \left(\right) + (p-1) \cdot \left(\right) + p(p-1) \cdot \left(\right)$$
$$+ p(p-1) \cdot \left(\right) + p^2(p-1) \cdot \left(\right) + (p^2-1) \cdot \left(\right) + p(p^2-1) \cdot \left(\right)$$

(2.12)

$$+ (p^2-1) \cdot \left(\right) + p(p^2-1) \cdot \left(\right) + p^2(p^2-1) \cdot \left(\right).$$

Subtracting (2.11) from (2.10) shows that this is exactly $|E| - |V| + 1$ [182], as expected

from the proof of Proposition 2.3.22. To match the classical proof in [181] we switch to from mergers to expansions $\mathcal{E}$. Repeating the arguments of the proof with $\deg_{\text{in}}(v)$ in place of $\deg_{\text{out}}(v)$ shows $\mathcal{E} = |E| - |V| + 1$ as well. This is intuitively clear from the fact that the trellis both starts and ends with a single vertex. Writing a similar expression to (2.12) for expansions and setting $\mathcal{E} = \mathcal{M}$, the diagrams which share a left-right symmetry cancel leaving only an equality of total asymmetric edge configurations.

Returning to our goal, it is clear from Section 2.3 that the number of paths from V_0 to any vertex in V_i is $p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_i}}$ and the total number of paths from V_0 to all of V_i is $p^{\dim \mathcal{S}^{\perp}_{|\mathfrak{p}_i}}$. Furthermore, for any definition of a trellis, the number of paths is at most $p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_i}}$ and the total number of paths is at least $p^{\dim \mathcal{S}^{\perp}_{|\mathfrak{p}_i}}$.

Lemma 2.3.24 (Lemma 5 [181])

Let $\mathscr{P}_i(v)$ denote the number of paths from V_0 to $v \in V_i$ and $\mathscr{P}_i = \sum_{v \in V_i} \mathscr{P}_i(v)$ be the number of paths from V_0 to all vertices in V_i. Then

$$\mathscr{P}_i = 1 + \sum_{j=0}^{i-1} \sum_{v \in V_j} \mathscr{P}_j(v) \left(\deg_{\text{out}}(v) - 1\right).$$

Proof. We proceed by induction. For $i = 1$, $\mathscr{P}_1 = \deg_{\text{out}}(\overline{0})$, as desired. Now suppose the formula is true for some i. Then

$$\begin{aligned}
\mathscr{P}_i &= \sum_{v \in V_{i-1}} \mathscr{P}_{i-1}(v) \deg_{\text{out}}(v) \\
&= \sum_{v \in V_{i-1}} \mathscr{P}_{i-1}(v) \left(\deg_{\text{out}}(v) - 1\right) + \sum_{v \in V_{i-1}} \mathscr{P}_{i-1}(v) \\
&= \sum_{v \in V_{i-1}} \mathscr{P}_{i-1}(v) \left(\deg_{\text{out}}(v) - 1\right) + 1 + \sum_{j=0}^{i-2} \sum_{v \in V_j} \mathscr{P}_j(v) \left(\deg_{\text{out}}(v) - 1\right) \\
&= 1 + \sum_{j=0}^{i-1} \sum_{v \in V_j} \mathscr{P}_j(v) \left(\deg_{\text{out}}(v) - 1\right).
\end{aligned}$$

□

The next result is used but not proved in [181]. We present the proof for the quantum case; the proof of their original expression follows from this by restricting to $\dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{i+1}}^{\perp} \leq 1$. Recall that $\{\dim \mathcal{S}_{\mathfrak{p}_i}^{\perp}\}$ is an increasing sequence whose value changes when a new right index is encountered in the TOF. Let $\{R_i\}$ be the locations of the right indices. The idea behind the following proof is to divide the index set $\{0, \ldots, n\}$ into intervals of constant past dimension and then make arguments about the locations $\{R_i\}$. To be explicit, in this notation $\dim \mathcal{S}_{\mathfrak{p}_{R_i}}^{\perp} < \dim \mathcal{S}_{\mathfrak{p}_{R_{i+1}}}^{\perp}$ and $\dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} = \dim \mathcal{S}_{\mathfrak{p}_{R_a}}^{\perp}$ if and only if $R_a \leq i < R_{a+1}$. The proof is purely algebraic.

Proposition 2.3.25

Let unprimed quantities be with respect to the syndrome trellis and primed quantities be with respect to any other trellis for the same code. Denote by

$$\mathcal{E}_j = \sum_{v \in V_j} (\deg_{\text{out}}(v) - 1)$$

the number of expansions at depth j and likewise for the primes, and define

$$\Delta_i \coloneqq \sum_{j=0}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right).$$

Then for $R_\kappa < i \leq R_{\kappa+1}$,

$$\sum_{j=0}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) = \Delta_i + \sum_{a=1}^{\kappa} p_{\kappa,a}(p_a - 1)\Delta_{R_a}$$

where $p_a = p^{\dim \mathcal{S}_{\mathfrak{p}_{R_a}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{R_a-1}}^{\perp}}$ and $p_{\kappa,a} = p^{\dim \mathcal{S}_{\mathfrak{p}_{R_\kappa}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{R_a}}^{\perp}}$.

Proof. We proceed by induction on κ. For $\kappa = 1$,

$$R_1 < i \leq R_2,$$

$$\dim \mathcal{S}_{\mathfrak{p}_{R_0}}^{\perp} = 0,$$

$$\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}} = \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_1}},$$
$$p_{1,1} = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_1}} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_1}}} = 1,$$
$$p_1 = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_1}} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_0}}} = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_1}}}.$$

Then,

$$\begin{aligned}
\sum_{j=0}^{i-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) &= \sum_{j=0}^{R_1-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) + \sum_{j=R_1}^{i-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) \\
&= p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_1}}} \sum_{j=0}^{R_1-1} \left(\mathcal{E}'_j - \mathcal{E}_j\right) + \sum_{j=R_1}^{i-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_j}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) \\
&= p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_1}} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_0}}} \sum_{j=0}^{R_1-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_0}}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) - \sum_{j=0}^{R_1-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_j}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) \\
&\quad + \sum_{j=0}^{R_1-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_j}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) + \sum_{j=R_1}^{i-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_j}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) \\
&= p_1 \sum_{j=0}^{R_1-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_j}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) - \sum_{j=0}^{R_1-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_j}}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) + \sum_{j=0}^{i-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_j}} (\mathcal{E}'_j - \mathcal{E}_j) \\
&= (p_1 - 1) \sum_{j=0}^{R_1-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_j}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) + \Delta_i \\
&= p_{1,1}(p_1 - 1)\Delta_{R_1} + \Delta_i.
\end{aligned}$$

Assume the result is true for $\kappa - 1$. We wish to show that

$$\sum_{j=0}^{i-1} p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) = \Delta_i + \sum_{a=1}^{\kappa} p_{\kappa,a}(p_a - 1)\Delta_{R_a},$$

where

$$R_\kappa < i \leq R_{\kappa+1},$$
$$p_{\kappa,a} = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_\kappa}} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_a}}},$$
$$p_a = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_a}} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{R_{a-1}}}}.$$

We have,

$$\begin{aligned}
\sum_{j=0}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) &= \sum_{j=0}^{R_\kappa - 1} p^{\dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) + \sum_{j=R_\kappa}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) \\
&= \sum_{j=0}^{R_\kappa - 1} p^{\dim \mathcal{S}_{\mathfrak{p}_{R_\kappa}}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) + \sum_{j=R_\kappa}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_{R_\kappa}}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) \\
&= p^{\dim \mathcal{S}_{\mathfrak{p}_{R_\kappa}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{R_{\kappa-1}}}^{\perp}} \sum_{j=0}^{R_\kappa - 1} p^{\dim \mathcal{S}_{\mathfrak{p}_{R_{\kappa-1}}}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) + \sum_{j=R_\kappa}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) \\
&= p^{\dim \mathcal{S}_{\mathfrak{p}_{R_\kappa}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{R_{\kappa-1}}}^{\perp}} \left(\Delta_{R_\kappa} + \sum_{a=1}^{\kappa-1} p_{\kappa-1,a}(p_a - 1)\Delta_{R_a}\right) + \sum_{j=R_\kappa}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) \\
&= p^{\dim \mathcal{S}_{\mathfrak{p}_{R_\kappa}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{R_{\kappa-1}}}^{\perp}} \Delta_{R_\kappa} + p^{\dim \mathcal{S}_{\mathfrak{p}_{R_\kappa}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{R_{\kappa-1}}}^{\perp}} \sum_{a=1}^{\kappa-1} p^{\dim \mathcal{S}_{\mathfrak{p}_{R_{\kappa-1}}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{R_a}}^{\perp}} (p_a - 1)\Delta_{R_a} \\
&\quad + \sum_{j=R_\kappa}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) \\
&= p_\kappa \Delta_{R_k} - \Delta_{R_k} + \Delta_{R_k} + \sum_{a=1}^{\kappa-1} p^{\dim \mathcal{S}_{\mathfrak{p}_{R_\kappa}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{R_a}}^{\perp}} (p_a - 1)\Delta_{R_a} + \sum_{j=R_\kappa}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) \\
&= (p_\kappa - 1)\Delta_{R_\kappa} + \sum_{j=0}^{R_\kappa - 1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) + \sum_{a=1}^{\kappa-1} p_{\kappa,a}(p_a - 1)\Delta_{R_a} + \sum_{j=R_\kappa}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) \\
&= p_{\kappa,\kappa}(p_\kappa - 1)\Delta_{R_\kappa} + \sum_{a=1}^{\kappa-1} p_{\kappa,a}(p_a - 1)\Delta_{R_a} + \sum_{j=0}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \left(\mathcal{E}_j' - \mathcal{E}_j\right) \\
&= \sum_{a=1}^{\kappa} p_{\kappa,a}(p_a - 1)\Delta_{R_a} + \Delta_i.
\end{aligned}$$

□

Theorem 2.3.26 (Theorem 6 [181])

Let $|E|$ and $|V|$ be with respect to the syndrome trellis and $|E'|$ and $|V'|$ be with respect to any other definition of a trellis for the same code. Then $|E'| - |V'| + 1 \geq |E| - |V| + 1$.

Proof. Let unprimed quantities be with respect to the syndrome trellis and primed quanti-

ties for the other trellis. For the syndrome trellis,

$$\mathscr{P}_i = 1 + \sum_{j=0}^{i-1} \sum_{v \in V_j} \mathscr{P}_j(v) \left(\deg_{\text{out}}(v) - 1\right) = 1 + \sum_{j=0}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \sum_{v \in V_j} \left(\deg_{\text{out}}(v) - 1\right),$$

and for any other trellis,

$$\mathscr{P}'_i = 1 + \sum_{j=0}^{i-1} \sum_{v' \in V'_j} \mathscr{P}'_j(v') \left(\deg_{\text{out}}(v') - 1\right) \leq 1 + \sum_{j=0}^{i-1} p^{\dim \mathcal{S}_{\mathfrak{p}_j}^{\perp}} \sum_{v' \in V'_j} \left(\deg_{\text{out}}(v') - 1\right).$$

Since $\mathscr{P}'_i \geq \mathscr{P}_i$, subtracting the two expressions gives $\Delta_i \geq 0$. Then

$$\begin{aligned}
(|E'| - |V'| + 1) - (|E| - |V| + 1) &= \sum_{j=0}^{n-1} \left(\mathcal{E}'_j - \mathcal{E}_j\right) \\
&= \frac{1}{p^{\dim \mathcal{S}_{\mathfrak{p}_{n-1}}^{\perp}}} \sum_{j=0}^{n-1} p^{\dim \mathcal{S}_{\mathfrak{p}_{n-1}}^{\perp}} \left(\mathcal{E}'_j - \mathcal{E}_j\right) \\
&= \frac{1}{p^{\dim \mathcal{S}_{\mathfrak{p}_{n-1}}^{\perp}}} \left(\Delta_n + \sum_{a=1}^{n-1} p_{n-1,a}(p_a - 1)\Delta_{R_a}\right) \\
&\geq 0,
\end{aligned}$$

where the last line follows because every term in the sum is positive. □

Since the syndrome trellis is a minimal representation of the code that is invariant with respect to the generators, following Theorems 2.3.22 and 2.3.26, it is often argued in the classical literature that the quantity $|E|$ should be regarded as a fundamental description of how hard it is to decode a given code, rivaling in importance with n, k, and d. Adopting this philosophy shows, for example, that the color codes are fundamentally more difficult to decode than the rotated surface code of the same distance without invoking projections or hypergraph matching. We provide quantitative data on this in Example 7 in the next section after introducing the CSS splitting of trellises.

Proposition 2.3.1 shows that the trellis essentially functions as a compact lookup table

for $\mathcal{S}^{\perp}$. The key to the efficiency of the Viterbi algorithm is its ability to make decisions about all of the elements of this set without computing every path individually. For a given $v \in V_i$, edge sharing in the trellis enables the algorithm to simultaneously check every element in $\mathcal{S}^{\perp}_{|\mathfrak{p}_i}$ ending at v. When it discards all outgoing edges for v, a total of $|\mathcal{S}^{\perp}_{\mathfrak{f}_i}|$ elements are eliminated from $\mathcal{S}^{\perp}$ for further consideration (see Figure 2.4).

Assuming $|E|$ scales at least cubically in n, the cost of the Viterbi algorithm also dominates that of finding the pure error T given the syndrome. Finding a pure error with the given syndrome is not difficult. The trellis decoder may use any valid T since the most likely solution is an element of the set LST enumerated by the paths of the trellis and will hence be found by the Viterbi algorithm. Given a potentially high-weight pure error, decoding may therefore be interpreted as a refinement process to the minimum-weight solution. Here, we use pseudoinverses for the syndromes to find T.

2.4 CSS Codes

CSS codes have the property that the generators of $\mathcal{S}$ ($\mathcal{S}^{\perp}$) split into those with only X or Z. It is common in QEC to decode each set of generators independently then combine the results into a single correction. This has the advantage of reducing decoding complexity and enabling parallelization at the expense of ignoring potential X-Z correlations. The same technique can, of course, be used with trellises. As a first example, consider the trellis diagrams in Figure 2.5. Figure 2.5c shows the trellis diagram of the distance three rotated surface code and Figures 2.5a and 2.5b show the effect on the trellis of considering the X and Z stabilizers separately. Figure 2.6 shows the same for the distance three color code. The reduction in trellis and decoding complexity is immediately apparent even for these small examples.

An immediate consequence of decoding the X and Z stabilizers separately is that moving from left-to-right in the TOF, the past (future) can only increase (decrease) by a maximum of one stabilizer generator: $\dim \mathcal{S}^{\perp}_{\mathfrak{f}_i} - \dim \mathcal{S}^{\perp}_{\mathfrak{f}_{i+1}} \leq 1$ and $\dim \mathcal{S}^{\perp}_{\mathfrak{p}_i} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{i-1}} \leq 1$.

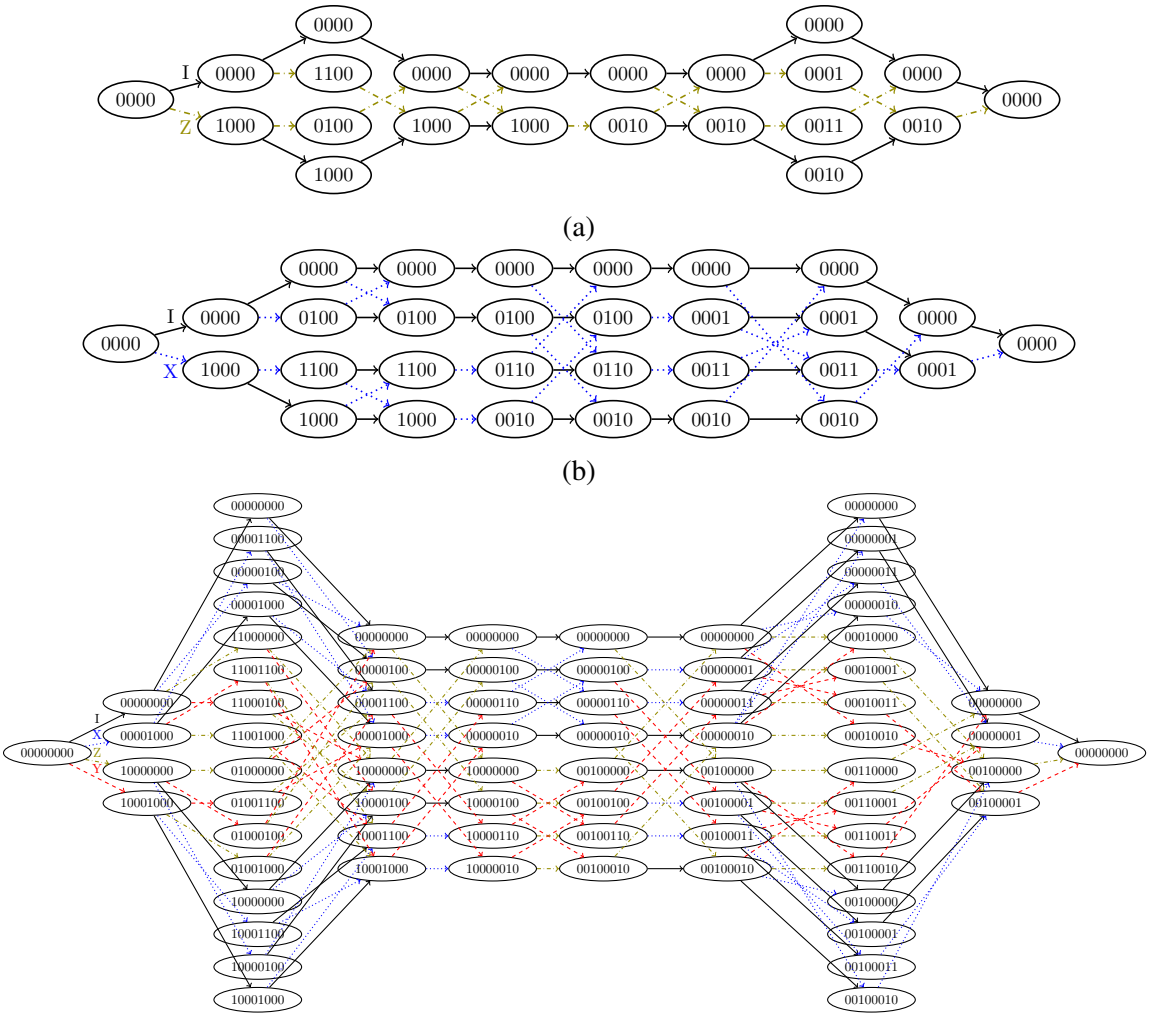


Figure 2.5: Trellis diagrams for the distance three rotated surface codes: (a) X stabilizers only, (b) Z stabilizers only, (c) the full code with vertices organized by the trellis product of (a) with (b).

Thus, in contrast to Corollary 2.3.9, none of the more complicated edge configurations in Table 2.1 with a two are allowed and the edge configurations with the highest contribution of edges do not occur, i.e.,

$$|V_{X/Z}|-1 = 1\cdot\#\left(\begin{array}{c}\bullet\!\!-\!\!\bullet\end{array}\right)+1\cdot\#\left(\begin{array}{c}\bullet\\ \bullet\!\!-\!\!\bullet\end{array}\right)+p\cdot\#\left(\begin{array}{c}\bullet\!\!-\!\!\bullet\\ \bullet\end{array}\right)+1\cdot\#\left(\begin{array}{c}\bullet\\ \bullet\end{array}\right)+p\cdot\#\left(\begin{array}{c}\bullet\!\!\times\!\!\bullet\\ \bullet\!\!\times\!\!\bullet\end{array}\right), \tag{2.13}$$

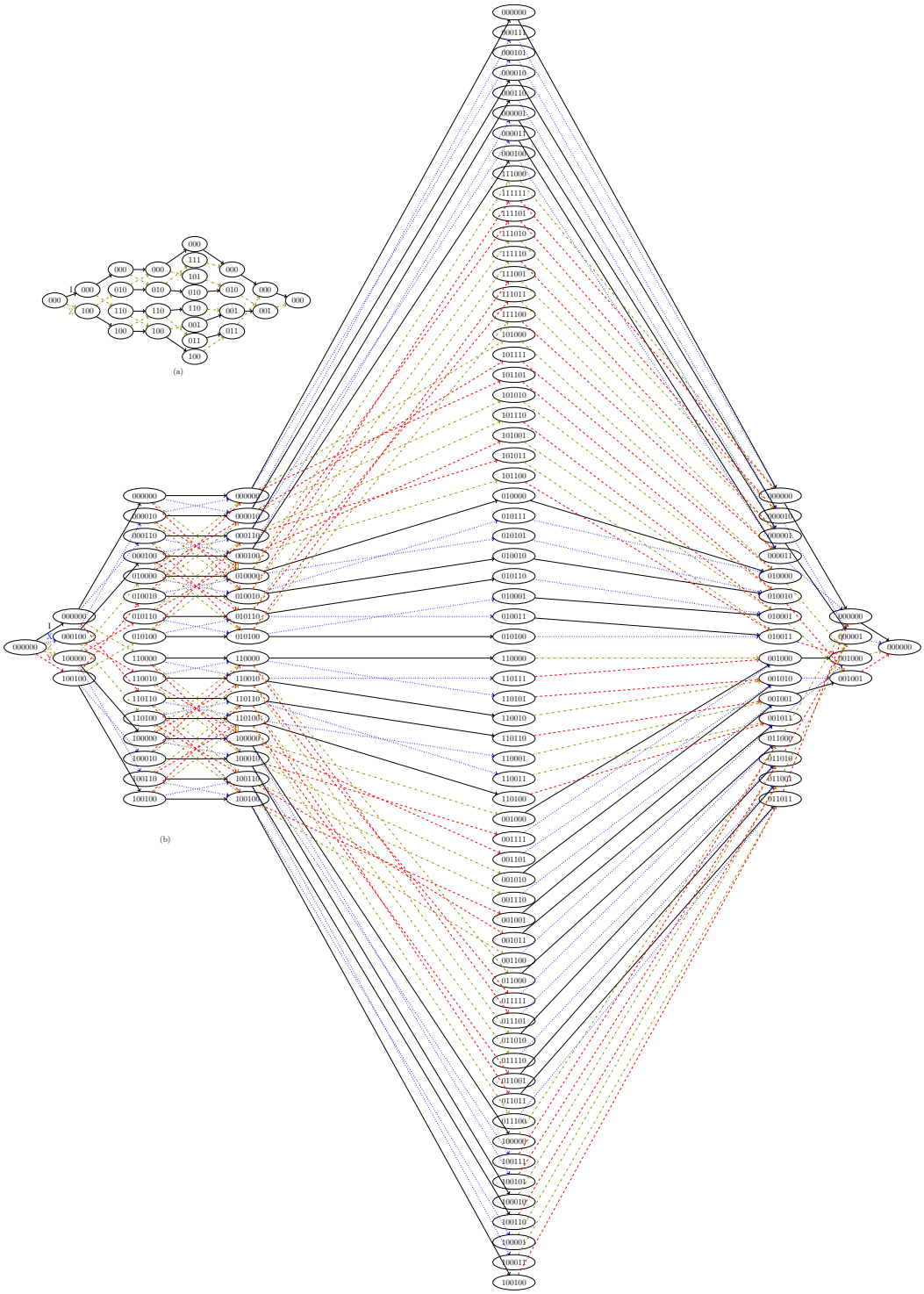
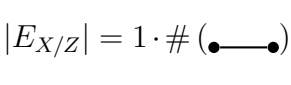


Figure 2.6: Trellis diagrams for the distance three color code: (a) X (or Z) stabilizers only, (b) the full code with vertices organized by the trellis product of (a) with itself.

and

$$|E_{X/Z}| = 1 \cdot \#\left(\right.$$  $$\left.\right) + p \cdot \#\left(\right.$$ $$\left.\right) + p \cdot \#\left(\right.$$ $$\left.\right) + p \cdot \#\left(\right.$$  $$\left.\right) + p^2 \cdot \#\left(\right.$$  $$\left.\right).$$ (2.14)

Thus, stabilizer CSS codes which have unit dimension change in the TOF of the X and Z stabilizers at the same index i will have higher dimension changes and hence more edges and will therefore be more difficult to decode than those which do not. It follows that self-dual codes are more difficult to decode than non-self-dual codes of the same parameters. Note that applying $\mathcal{E} = \mathcal{M}$ to CSS trellises gives [182]

$$\#\left(\right.$$ $$\left.\right) = \#\left(\right.$$  $$\left.\right).$$

Let $\mathcal{S}$ be a stabilizer code given by the CSS construction with $C_1 = [n, k_1, d_1]_p$, $C_2 = [n, k_2, d_2]_p$, and $C_2^T \subseteq C_1$. Denote by G_i and H_i the generator and parity-check matrices for C_i, respectively. The stabilizers of the quantum code are given, in symplectic form $(X \mid Z)$, by $H_1 \oplus H_2$. The set $\mathcal{S}^{\perp}$ is generated by the corresponding normalizer matrix [183, 184, 185]

$$\begin{pmatrix} 0 & G_1 \\ G_2 & 0 \end{pmatrix},$$

as can be seen immediately from the relationship between the generator and parity-check matrices of classical codes. The trellis for the X (Z) stabilizers only in a CSS code is therefore determined by the classical trellis whose paths are in one-to-one correspondence with the codewords of G_1 (G_2). Edge labels for CSS trellises should be restricted to I and Z or I and X with weights $-\log \Pr(I) - \log \Pr(X)$ and $-\log \Pr(Y) - \log \Pr(Z)$ for X and similarly for the Z stabilizers.

It is well-known in classical trellis theory that the dimension of V_i is equal to the dimension of the corresponding V_i for the dual code. Since one has p^{n-k} possible syndromes and the other p^k, we immediately get the following.

Corollary 2.4.1 (Wolf Bound For CSS Codes [173])

Let C_i be as above. Then for the X stabilizer trellis, $|V_{X,i}| \leq p^{\min\{k_1, n-k_1\}}$, and for the Z stabilizer trellis, $|V_{Z,i}| \leq p^{\min\{k_2, n-k_2\}}$.

Example 5. The distance three color code is equivalent to the well-known $[[7, 1, 3]]$ Steane code. This is a CSS code constructed with the $[7, 4, 3]$ binary Hamming code and its dual. Hence, $|V_{X/Z,i}| \leq 2^{\min\{3,4\}} = 8$, which agrees with Figure 2.6.

It is perhaps not surprising that the full trellis for a CSS code turns out to be a product of the two trellises of its component codes. For instance, the active generators of the full code are the active generators of the X code and the active generators of the Z code. Thus,

$$|V_i| = p^{\dim \mathcal{S}_{X,\mathfrak{a}} + \dim \mathcal{S}_{Z,\mathfrak{a}}} = p^{\dim \mathcal{S}_{X,\mathfrak{a}}} p^{\dim \mathcal{S}_{Z,\mathfrak{a}}} = |V_{X,i}||V_{Z,i}|.$$

Definition 2.4.2 (Trellis Product [179])

The trellis product of two trellises with vertices V, V' and edges E, E', respectively, is denoted by $\square_i$ and has vertices

$$V \square_i V' = \bigcup_{i=0}^{n} V_i \times V_i'$$

and edges

$$E \square_i E' = \{((v_i, v_i'), PP', (v_{i+1}, v_{i+1}')) \mid (v_i, P, v_{i+1}) \in E_i, (v_i', P', v_{i+1}') \in E_i'\}.$$

Remark: This is sometimes called the Shannon product [186] for historical reasons after Claude Shannon who first described the product of two channels operating at the same time or also the tensor product [187]. However, much of the classical literature on the trellis product is based on non-syndrome trellises and does not apply here. We will completely classify this product in Section 3.4.

Example 6. Using shorthand $(\Delta\dim \mathcal{S}_{\mathfrak{p}_i}^{\perp}, \Delta\dim \mathcal{S}_{\mathfrak{f}_i}^{\perp}) = (m, n)$ for edge configurations in

Table 2.1, it may be checked that $(m,n)\square_i(m',n') = (m+m', n+n')$. Such configurations occur in the full trellis as graph products of lower dimensional configurations in the CSS trellises. The vertices of Figure 2.5c are ordered by the trellis product of Figure 2.5a by Figure 2.5b and the vertices of Figure 2.6 (b) by the trellis product of (a) with itself.

Lemma 2.4.3

Let $\mathcal{S}$ be a CSS code with X stabilizers, $\mathcal{S}_X$, and Z stabilizers, $\mathcal{S}_Z$. Let V_X, E_X, $\deg_{in,X}$, and $\deg_{out,X}$ denote the appropriate quantities for the trellis of $\mathcal{S}_X$, and likewise for $\mathcal{S}_Z$. Then the trellis for $\mathcal{S}$ is given by the trellis product of the trellises for $\mathcal{S}_X$ and $\mathcal{S}_Z$. Furthermore,

(i) $|V_i| = |V_{X,i}||V_{Z,i}|$,

(ii) $|E_i| = |E_{X,i}||E_{Z,i}|$,

(iii) $\deg_{in,i} = |\deg_{in,X,i}||\deg_{in,Z,i}|$,

(iv) $\deg_{out,i} = |\deg_{out,X,i}||\deg_{out,Z,i}|$.

The proof of the enumerated parts of this lemma follow from Definition 2.4.2 and the larger claim is a restatement of the well-known fact that X and Z errors may be decoded independently for CSS codes. A rigorous proof is fairly trivial and we leave it to the reader. These results may be demonstrated with the diagrams above. Note that the Space Theorems 2.3.7 do not hold for CSS splittings. Following the proof of the theorem, the number of vertices is isomorphic to $|\mathcal{S}^\perp/\ker\sigma_i|$ but now the kernel with respect to $\mathcal{S}_X$ includes $\mathcal{S}_X^\perp$ as well as the past and future elements of $\mathcal{S}_Z^\perp$:

$$|V_{X,i}| = p^{\dim\mathcal{S}^\perp - \dim\mathcal{S}^\perp_{Z,\mathfrak{p}_i} - \dim\mathcal{S}^\perp_{Z,\mathfrak{f}_i} - \dim\mathcal{S}^\perp_X} = p^{\dim\mathcal{S}^\perp_Z - \dim\mathcal{S}^\perp_{Z,\mathfrak{p}_i} - \dim\mathcal{S}^\perp_{Z,\mathfrak{f}_i}}. \tag{2.15}$$

Similar equations hold for $|V_{Z,i}|$ and the edges. The same idea holds for counting vertices and edges for an arbitrary subset of generators of $\mathcal{S}$. This split trellis idea may be used

in more generality. The proof of the following with the clear associativity of the trellis product implies the main part of the lemma above.

Theorem 2.4.4

For $\mathcal{S} = \langle S_1, \ldots, S_{n-k} \rangle$, the minimal trellis for $\mathcal{S}$ is given by the trellis product of the minimal trellises for each S_i.

Proof. We show minimality, which is the non-trivial part of the proof.

We proceed by induction on the number of generators in the product. Let $\tilde{\mathcal{S}}_i^{\perp}$ be the subset of generators of $\mathcal{S}^{\perp}$ (in TOF) which have trivial partial syndromes with respect to S_i. Let $[L_1, R_1]$ and $[L_2, R_2]$ be the spans of S_1 and S_2, respectively, and construct trellises for each following Section 2.2 with vertex profiles

$$|V_i^1| = \begin{cases} 1 & \text{if } i \in [0, L_1) \\ p^{\dim \mathcal{S}^{\perp} - \dim \tilde{\mathcal{S}}_1^{\perp}} & \text{if } i \in [L_1, R_1) \\ 1 & \text{if } i \in [R_1, n] \end{cases} \qquad |V_i^2| = \begin{cases} 1 & \text{if } i \in [0, L_2) \\ p^{\dim \mathcal{S}^{\perp} - \dim \tilde{\mathcal{S}}_2^{\perp}} & \text{if } i \in [L_2, R_2) \\ 1 & \text{if } i \in [R_2, n] \end{cases}.$$

The vertex profile of the trellis product of the trellises for S_1 and S_2 are given by Lemma 2.4.3 (i) to be

$$\begin{aligned} |V_i^{1,2}| &= \begin{cases} 1 & \text{if } i \in [0, L_1) \\ |V_i^1| & \text{if } i \in [L_1, L_2) \\ |V_i^1||V_i^2| = p^{\left(\dim \mathcal{S}^{\perp} - \dim \tilde{\mathcal{S}}_1^{\perp}\right) + \left(\dim \mathcal{S}^{\perp} - \dim \tilde{\mathcal{S}}_2^{\perp}\right)} = p^{\dim \mathcal{S}^{\perp} - \dim \tilde{\mathcal{S}}_{1\cap 2}^{\perp}} & \text{if } i \in [L_2, R_1) \\ |V_i^2| & \text{if } i \in [R_1, R_2) \\ 1 & \text{if } i \in [R_2, n] \end{cases} \\ &= p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} - \dim \mathcal{S}_{\mathfrak{f}_i}^{\perp} - \dim \tilde{\mathcal{S}}_i^{\perp}}, \end{aligned}$$

where we have used that for sets $A, B \subset C$, $(C\backslash A) \cup (C\backslash B) = C\backslash(A \cap B)$, $\tilde{\mathcal{S}}^{\perp}_{1\cap 2} \coloneqq \tilde{\mathcal{S}}^{\perp}_{1} \cap \tilde{\mathcal{S}}^{\perp}_{2}$, the past and future are taken with respect to the generators S_1 and S_2, and the intersection $\tilde{\mathcal{S}}^{\perp}_{i}$ depends on i. Having recovered the minimal form of Theorem 2.3.7, assume the theorem holds for the trellis product of $S_1, \ldots, S_{n-k-1}$. It remains to show that the overlap $|V_i^{1,\ldots,n-k-1}||V_i^{n-k}|$ produces Equation (2.4). In this case $\tilde{\mathcal{S}}^{\perp}_{1\cap\ldots\cap n-k} = \varnothing$ so

$$|V_i^{1,\ldots,n-k-1}||V_i^{n-k}| = |V_i^{1,\ldots,n-k-1}|\, p^{\dim \mathcal{S}^{\perp} - \dim \tilde{\mathcal{S}}^{\perp}_{n-k}} = p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_i} - \dim \mathcal{S}^{\perp}_{\mathfrak{f}_i}}.$$

The proof for the edges is similar. □

This splitting idea may be used to decode in a similar manner to the CSS codes where each grouping of syndrome bits are corrected independently and then combined. The success of this approach is tied to finding an appropriate grouping of generators with respect to which the logical operators split "nicely" such that the combined corrections do not introduce a logical error not returned by the individual trellises. We return to this idea in Example 9.

If we adopt the idea that $|E|$ represents a fundamental description of how difficult it is to decode, then the two quantities of interest to compare are $|E_X| + |E_Z|$ and $|E|$, where the latter is with respect to the full stabilizer code. In particular, for a self-dual code $2|E_{X/Z}| \leq |E_{X/Z}|^2$. It follows that splitting the decoding of a self-dual CSS code is square-root easier than decoding the full code. A rather trivial bound on the more general case is immediate.

Proposition 2.4.5

Let $|E|$ be the total number of edges in a trellis diagram for a CSS code whose X and Z codes have trellises with total number of edges $|E_X|$ and $|E_Z|$, respectively. Then,

$$|E_X| + |E_Z| \leq |E| \leq |E_X||E_Z| - p^2 n(n-1) \tag{2.16}$$

Proof. Let $\xi_i = \log_p E_{X,i}$ and $\eta_i = \log_p E_{Z,i}$ such that $|E_X| = \sum_{i=1}^{n} p^{\xi_i}$ and $|E_Z| =$

$\sum_{i=1}^{n} p^{\eta_i}$, and recall that $\xi_i, \eta_i \geq 1$. By Lemma 2.4.3, $|E| = \sum_{i=1}^{n} p^{\xi_i+\eta_i}$. The left inequality is clear. For the right, we have

$$\begin{aligned}
(|E_X| + |E_Z|)^2 &= |E_X|^2 + |E_Z|^2 + 2|E_X||E_Z| \\
&= |E_X|^2 + |E_Z|^2 + 2\left[\left(\sum_{i=1}^{n} p^{\xi_i+\eta_i}\right) + \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} p^{\xi_i+\eta_i}\right] \\
&= |E_X|^2 + |E_Z|^2 + 2\left[|E| + p^2 \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} p^{\xi_i+\eta_i-2}\right] \\
&\geq |E_X|^2 + |E_Z|^2 + 2|E| + 2p^2 n(n-1),
\end{aligned}$$

where in the last line we have assumed that $\xi_i = \eta_i = 1$ for all i. □

The right-hand side of (2.16) is tight for the case that there are a minimal number of edges at each depth such that $|E_i| = 4$ for all i. This, of course, is rare, usually only occurring around the left and right ends of the trellis, and the more the trellis deviates from this value the worse this upper bound becomes.

It would be more useful to obtain bounds on the number of edges in any of the diagrams with only the values n, k, and d. We leave this for future work and simply note that several classical trellis bounds could potentially be exploited for CSS codes. An easy one in particular worth mentioning is the case of classical maximum distance separable (MDS) codes where it is well-known that the sequence $\{|V_i|\}$ has the following pattern [173]

$$\{1, p, p^2, \ldots, p^{\min\{k,n-k\}}, p^{\min\{k,n-k\}}, \ldots, p^2, p, 1\}.$$

Combining this with the fact that $E_i \geq V_i$, removing the contribution from V_0 and summing

the geometric series we get

$$|E_{X/Z}| \geq \frac{2p^{\min\{k,n-k\}} - p - 1}{p-1}, \tag{2.17}$$

which is tight for the (unrealistic) case that the outgoing degree at every i is one. Replacing p with p^2 in (2.17) provides the corresponding bound on $|E|$ for a self-dual CSS code. Combining the two loose bounds (2.16) and (2.17) does not appear useful.

Example 7. Returning to the proposal that $|E|$ represents a fundamental parameter for the code, vertex and edge counts for the 4.8.8 and 6.6.6 color codes, rotated surface codes, and their CSS splittings are given in the following tables. The non-CSS XZZX surface codes have the same values as the full rotated surface codes. A standard qubit numbering order was applied to the surface codes but the qubit numbering for the color codes was assigned greedily to minimize the trellis. The difference in values between distances may become more consistent given more consistent numbering schemes. Note that the total vertex and edge counts for the CSS trellises are the sum of the X and Z counts.

Table 2.3: Vertex counts by distance for common codes.

	3	5	7	9	11	13	15	17	19	21
4.8.8	122	4042	83402	2126282	8673370	108195242	2074018922	36433758250	618938698730	10475888643050
4.8.8 X/Z	26	230	1382	8198	20058	66710	327278	1498758	6610590	28827294
6.6.6	122	2522	49802	496010	4719242	54470282	604814042	5357974490	40005091802	326581575962
6.6.6 X/Z	26	170	974	3966	14414	48878	174170	617322	1728842	5102498
RSurf	74	1098	10058	73034	464202	2708810	14898506	78468426	399856970	1985303882
RSurf X	22	118	470	1590	4854	13814	37366	97270	245750	606198
RSurf Z	30	198	854	2998	9334	26870	73206	191478	485366	1200118

Table 2.4: Edge counts by distance for common codes.

	3	5	7	9	11	13	15	17	19	21
4.8.8	232	7080	143272	3559336	14506536	175628968	3358695592	60870993576	1031533604008	17421128805544
4.8.8 X/Z	36	316	1884	11100	27084	89628	439100	2020188	8901500	38785916
6.6.6	232	4648	89512	832936	7708072	89669032	1007967784	8733820456	64652391976	532838443048
6.6.6 X/Z	36	236	1340	5372	19388	65852	234956	828556	2316044	6847020
RSurf	152	2152	19688	143336	913384	5341160	29425640	155189224	791674856	3934257128
RSurf X	30	172	700	2388	7316	20852	56436	146932	371188	915444
RSurf Z	44	284	1228	4332	13548	39148	106988	280556	712684	1765356

The difference in sizes between the X and Z trellises for the rotated surface code family is an artifact of the particular numbering system used in this work. Consider the X_2X_3 stabilizer in Figure 1.2. This is active only at depth two since at depth three its syndrome is already zero. On the other hand, the Z_1Z_6 stabilizer is active over five depths and hence

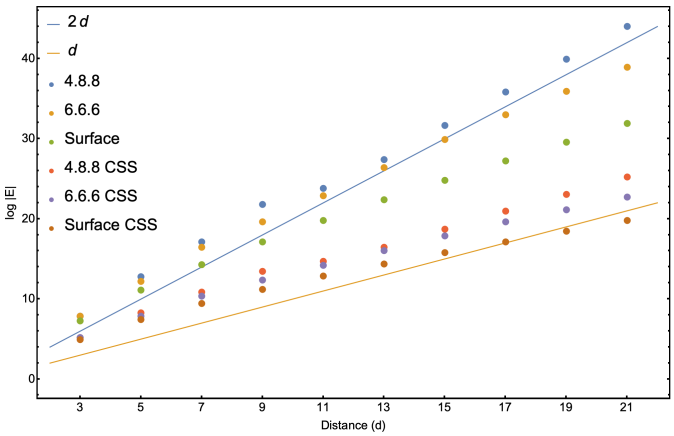


Figure 2.7: The scaling of the total edge counts for the trellises listed in Table 2.4.

contributes vertices over this entire range. Since the X and Z stabilizer measurements are independent, arranging the data with respect to two different numbering schemes will allow both the X and the Z trellises to be isomorphic. This demonstrates the strong effect permutations have on the trellis.

While an analytical formula for the edge scaling is currently missing, we can gain some numerical insights from Table 2.4. In Figure 2.7 we see that $|E|$ is exponential in the minimum distance d (compare to the lines $y = 2d$ (blue) and $y = d$ (orange)). The slight bend in the data points, most predominately seen for the CSS rotated surface codes, suggest a more complicated behavior. It is possible that the color codes also have the same trend but were not simulated to high enough distances to make this as visually apparent. It is also possible that the greedy algorithm used to assign qubits to the color code geometries was suboptimal enough to erase this effect.

Example 8. To further emphasize the effect permutations have on the trellis, draw the distance five and seven 4.8.8 color codes, choose an arbitrary boundary, and number the qubits from left to right, level by level. The minimum trellis for this configuration for the distance five code has $|V| = 5,242$ and $|E| = 9,000$ and has $|V| = 177,018$ and $|E| = 293,928$ for distance seven. Comparing to Tables 2.3 and 2.4, using this numbering scheme would increase the slope of the blue dots in Figure 2.7, potentially limiting the

ability to do large-scale simulations at a lower distance.

2.5 Simulations And Discussion

Numerical simulations were performed in the Julia programming language with pre-simulation computations in the MAGMA quantum coding theory library [188]. We begin by constructing the trellis. Historically, it is widely assumed, even to this day, that the trellises considered in this work are too large to ever build. Fortunately, the commonly referred to size estimates date back to the late 1980's, early 1990's when constructing the multi-gigabyte trellises required for color codes of distance 21 would have indeed been impossible. The construction algorithms in [167, 170] iterate through all p^{n+k} elements of $\mathcal{S}^{\perp}$. The theoretical guarantees of the Space Theorems 2.3.7 allow us to do better by only generating elements until we hit $|V|$ and $|E|$. Numerical trials using this method suggested the largest trellis considered in this work would take between one to two months to generate. Many classical construction algorithms are just as wasteful [173]. The theoretical results in this work provide a better algorithm, although we will also present a different algorithm in the next chapter based on trellis products of subcodes, which may be faster given the trellis for the subcodes have been constructed using the method described here.

Our algorithm is closest to [189] for classical codes, but is conceptually simpler and more efficient. The basic idea is to serially proceed through V_i picking a vertex and examining all possible outgoing edges for matches in V_{i+1}. This implicitly utilizes Corollary 2.3.2, although this result never appears in the literature. Here we rely on the theoretical guarantees provided by Corollary 2.3.2, Theorem 2.3.7, Lemma 2.3.8, and Theorem 2.3.12. By Theorem 2.3.7, the vertices at depth i are given by the set of all $(n-k)$-tuples whose bits corresponding to generators active at i range through all possible values. To construct the edges at section i, determine all of the vertices in V_{i-1} connecting to $\overline{0} \in V_i$ via Corollary 2.3.2. Then choose one of these $v \in V_{i-1}$ and determine all of the vertices in V_i to which it connects, again via Corollary 2.3.2. These are all of the vertices in the

edge configuration by Theorem 2.3.12, which may be completed with Corollary 2.3.2. This is the subgroup of E_i by Lemma 2.3.8. The rest of the section consists of all translations (cosets) of this subgroup. Our new proof of the classification of edge configurations allows us to only compute a small, bipartite graph for each depth to generate the entire trellis. The V_i are independent and may be constructed in parallel, after which the E_i may also be parallelized. For the codes considered in this work, only the distance 19 and 21 color code trellises were large enough to justify saving so as to not compute them on-the-fly later.

As previously mentioned, the vertex labels are required for the construction of and theoretical justification for the trellis, but an examination of the Viterbi algorithm shows that they serve no role in decoding. As such, we need not bother shifting the vertex labels for each measured syndrome, and generally we can safely discard them after the construction phase is completed. Even storing the labels as integers is similar to creating a lookup table and can take a non-trivial amount of memory starting at moderately sized codes. The storage requirements for the zero-syndrome trellis valid for any error model became slightly unwieldy to use for large-scale simulations for the distance 21 color codes, but this was implementation dependent. Choosing a specific error channel allowed the trellis to be translated to a data structure a fraction of the storage size of the original. For example, the distance 21, 4.8.8 CSS trellises come out to around 300 MB in contrast to the roughly 2 GB file size of the original data structure which includes all the information and generality. We leave a discussion of such improvements to the next chapter. However, since the trellis searches all elements of $\mathcal{S}^{\perp}$, the proper comparison to make here is to the size of a lookup table for the same code. Assuming the 2^{n+k} elements of $\mathcal{S}^{\perp}$ are stored as length-n strings of character size 1 byte, a lookup table for the $[[111, 1, 11]]$ code above would be on the order of $\sim 10^{17}$ exabytes.

To exemplify the versatility of trellis decoding, the row $n = 20$ was selected at random from the quantum error-correcting codes table at [171]. Extended codes (with weight one stabilizers) and codes with $k = 0$ or $d \leq 2$ were discarded, and code-capacity (memory

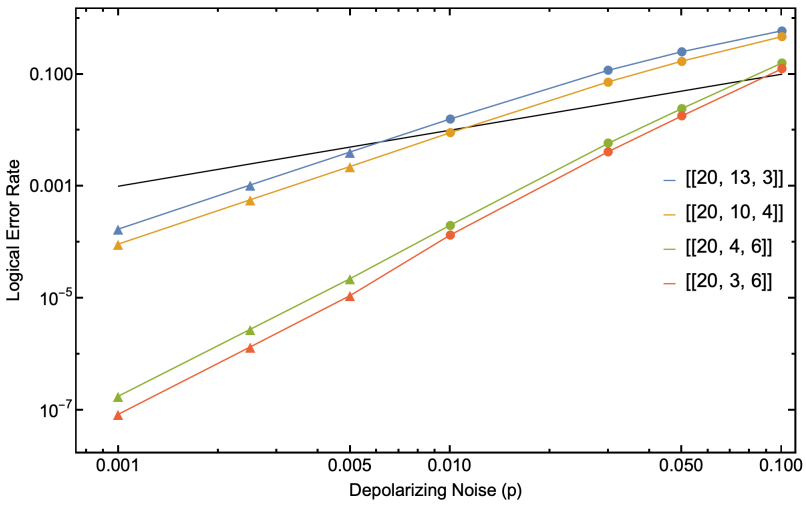


Figure 2.8: Simulated logical error rates for the four codes in the row $n = 20$ satisfying our requirements at [171]. Triangular data points are importance sampled to a tolerance of 10^{-9} and circular points are direct sampled; both methods were used and were found to agree at $p = 0.01$. The black line is $y = x$.

model) numerical simulations were performed on the remaining four codes: $[[20, 3, 6]]$, $[[20, 4, 6]]$, $[[20, 10, 4]]$, and $[[20, 13, 3]]$. These codes and their properties were previously unknown to the authors of this work and were never investigated. In particular, it is unknown if another decoding method is known for these codes and if there exist previous pseudothreshold results in the literature. Simulations consisted of a single round of error correction under depolarizing noise (1.46) following the discussion in the previous chapter. Results are presented in Figure 2.8, and the stabilizers for these codes are given below for convenience.

[[20, 3, 6]]

XIIIIIIIIYIXIYZYXZXI

ZIIIIIIIZXIZIYXYIXZY

IXIIIIIIZYZYXIXYYYIZ

IZIIIIIIIXYXZYXZZZIZ

IIXIIIIIZXIXYXZIYXXY

IIZIIIIIIZIZXXIYZXZI

IIIXIIIIIIXXXIZIIIXX

IIIZIIIIZIZZZZXZZZZX

IIIIXIIIIXYXXXXYXXXX

IIIIZIIIIZXZZZZXZZZZ

IIIIIXIIIYXZXZYZZYIX

IIIIIZIIZXZYZXYXXYIX

IIIIIIXIZYYZZXIZZIXY

IIIIIIZIIXXYYXYIIYZI

IIIIIIIXZXXIZZYXZIZZ

IIIIIIIZIZZIYIZXIYYZ

IIIIIIIIXIIIIXXXXXIZ

[[20, 4, 6]]

XIIIIIIIXYIXIZYZIYXZ

ZIIIIIIIZXIZIYXYIXZY

IXIIIIIIYYZYXXIZZZII

IZIIIIIIXXYXZZIYYYII

IIXIIIIIYXIXYIYXZIXX

IIZIIIIIXZIZXIXZYIZZ

IIIXIIIIXIXXXXYXXXXY

IIIZIIIIZIZZZZXZZZZX

IIIIXIIIIXYXXXXYXXXX

IIIIZIIIIZXZZZZXZZZZ

IIIIIXIIXYXZXYZYYZIY

IIIIIZIIZXZYZXYXXYIX

IIIIIIXIYYYZZIXYYXXX

IIIIIIZIXXXYYIZXXZZZ

IIIIIIIXYXXIZYZIYXZI

IIIIIIIZXZZIYXYIXZYI

[[20, 10, 4]]

XIIXZIIXYZIYIXZIZXIZ

ZIIZYIIZXYIXIZYIYZIY

IXIZZIIIYYXZXZIYIIZZ

IZIYYIIIXXZYZYIXIIYY

IIXZXIIXIXXXXYZIXZXY

IIZYZIIZIZZZZXYIZYZX

IIIIIXIXZZZZXIXYZIZY

IIIIIZIZYYYYZIZXYIYX

IIIIIIXZZXZXIXZZIZYY

IIIIIIZYYZYZIZYYIYXX

[[20, 13, 3]]

XIIZIXXIYZYIZYXZXIZY

ZIXIIXZYZIXZIXIYZYXY

IXXYIZZIXYXYXYXXIZII

IZXXIIIIYYYXIIXIYYZX

IIZYIXYZZXIIZYYIIXYY

IIIIXXXXXXXXXXXXXXX

IIIIZZZZZZZZZZZZZZZ

In order to quantitatively assess the success of trellis decoding, the same simulations were carried out for three common qubit code families, the rotated surface and 4.8.8, 6.6.6 color codes for the independent Z channel (1.52) up to at least distance 17. These codes and this noise model were chosen solely for the purpose of comparison with existing decoding literature. Due to the X-Z symmetry of the codes, the same results are obtained for independent X channel and depolarizing thresholds are at $3/2$ times the X or Z noise results decoded independently, which we numerically verified for low distances. There are, of course, many other highly-optimized decoders for topological codes, and these choices may not sufficiently demonstrate the value of trellis decoding which shines for moderately-sized QECCs for which there are no efficient decoding strategies. Data were taken at intervals of size 0.0025 around the suspected threshold to reduce the error caused by sampling and fitting.

Previous studies have reported a surface code Z threshold of 10.3% under MWPM [190] and 10.9% using the tensor network decoder of [111]. The statistical mechanical threshold is estimated to be 10.9% [110] without correlations and 12.6% with X and Z correlations taken into account [191]. Both the trellis and MWPM are minimum-weight decoders for this channel, so they should agree for uncorrelated noise. For correlated error models trellis decoding should beat MWPM since the minimum-weight correction on the full trellis will not always match the minimum-weight X correction combined with the minimum-weight Z correction returned by MWPM. For example, consider the error $Y_8Y_{13}Y_{18}$ on the distance five rotated surface code of Figure 1.2. The full trellis will return the correction $Y_8Y_{13}Y_{18}$, while the X- and Z-trellises will return Z_3Z_{23} and $X_8X_{13}X_{18}$, respectively; the latter of which is a logical error. The Viterbi algorithm takes the same number of steps every time and could be slower than using a small matching graph for a large code, so the expected number of errors should be taken into account when comparing the runtime of these two algorithms for uncorrelated error models. The results of our simulations are shown in Figure 2.9a.

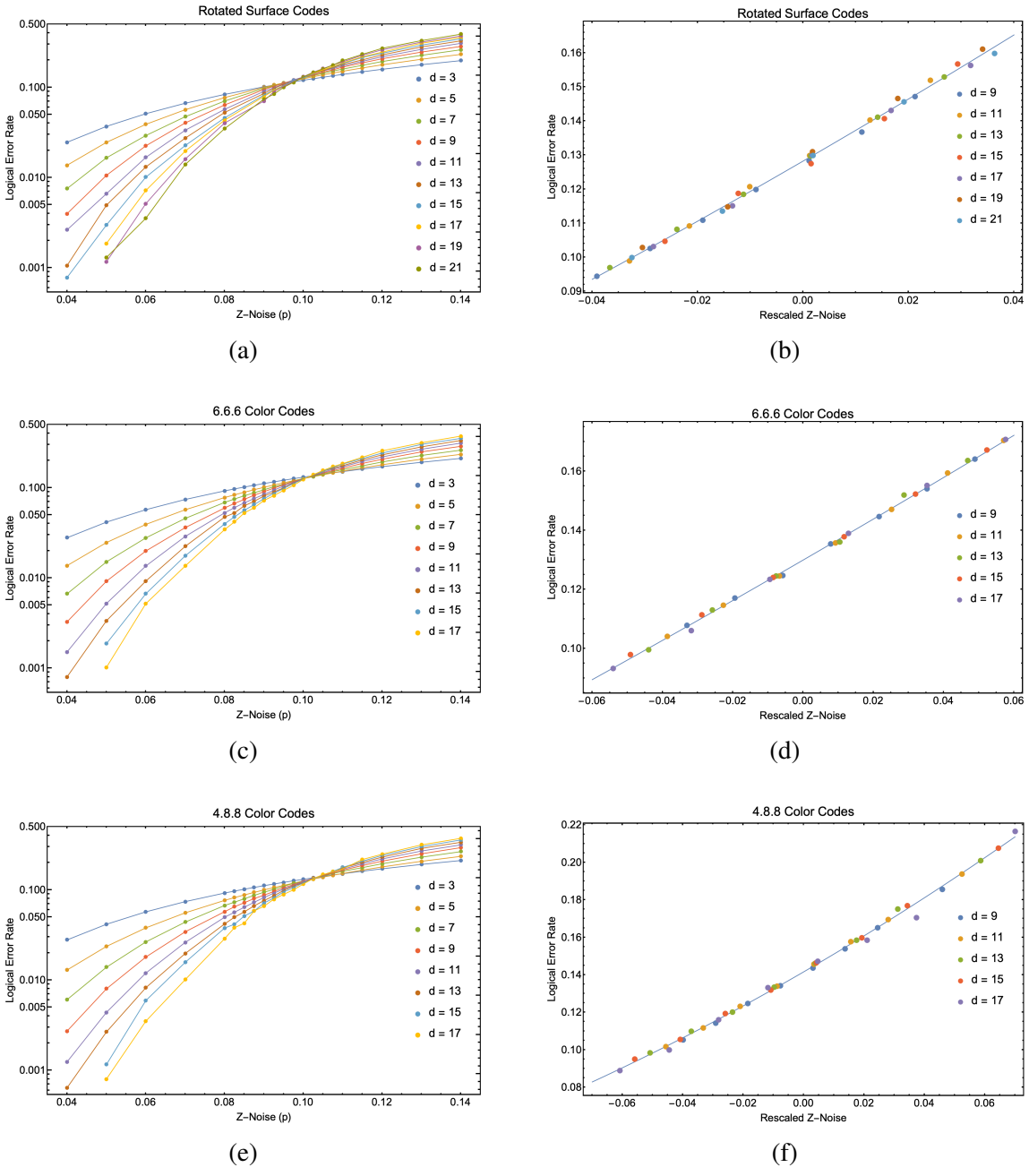


Figure 2.9: (a), (c), (e) - Threshold results for code capacity (memory model) simulations for the independent Z channel (1.52). (b), (d), (f) - Finite-size threshold analysis for the corresponding codes near threshold. The logical error rate is shown as a function of the rescaled Z error probability $x = (p - p_{\text{th}})d^{1/\nu}$. The solid line is the line of best fit to $A + Bx + Cx^2$. Note that the higher distance codes show some signs of under sampling for lower p.

The Restriction Decoder of [137] achieves a Z threshold of 10.2% for the 4.8.8 color codes with periodic boundary conditions. This was modified to include boundaries in [138]

which reports a full depolarizing noise threshold of 12.6% for the 6.6.6 lattice, which results in a Z threshold of roughly 8.4%. Reference [131] finds a Z threshold of roughly 8.7% for the 4.8.8 lattice. The statistical mechanical thresholds for both lattices are estimated to be 12.6% and 12.52% for the 6.6.6 and 4.8.8 families, respectively [192, 191]. The results of our simulations for the 6.6.6 and 4.8.8 families are shown in Figures 2.9c and 2.9e, respectively.

Following references such as [193] we perform a finite-size threshold analysis on our data by fitting to the ansatz

$$A + B(p - p_{\text{th}})d^{1/\nu} + C(p - p_{\text{th}})^2 d^{2/\nu}$$

for A, B, C, p_{th}, and ν starting with $d \geq 9$. For the rotated surface codes we find $p_{\text{th}} \approx 10.25\%$ ($\nu = 1.58$). It is noteworthy that [122] did not see convergence to this value below $d = 21$. For the color codes, it's possible that all of the distances reported in this work are within the finite-size-effect regime. Nevertheless, we see strong evidence for threshold-like behavior at 10.1% ($\nu = 1.29$) and 10.4% ($\nu = 1.51$) for the 6.6.6 and 4.8.8 families, respectively. Further simulations and analysis are required to better estimate the thresholds for these codes using this decoder. Figures 2.9b, 2.9d, and 2.9f show the logical error rate versus the rescaled Z noise probability $x = (p - p_{\text{th}})d^{1/\nu}$ for data near each threshold, along with the line of best fit.

Judging by the time it took to complete each simulation, we believe that the CSS, distance 21 trellis for the 4.8.8 color code ($[[421, 1, 21]]$) roughly represents the limit for large-scale simulations. Considering $|E|$ as a measure of the difficulty of decoding, trellis profiles for codes were compared to Tables 2.3 and 2.4 before using this method. We offer this as a rough benchmark but acknowledge that some of this may due in part to not properly leveraging the limited computational resources available for this work. For example, we noticed that a large percentage of the total simulation time for the distance 19 and 21 color

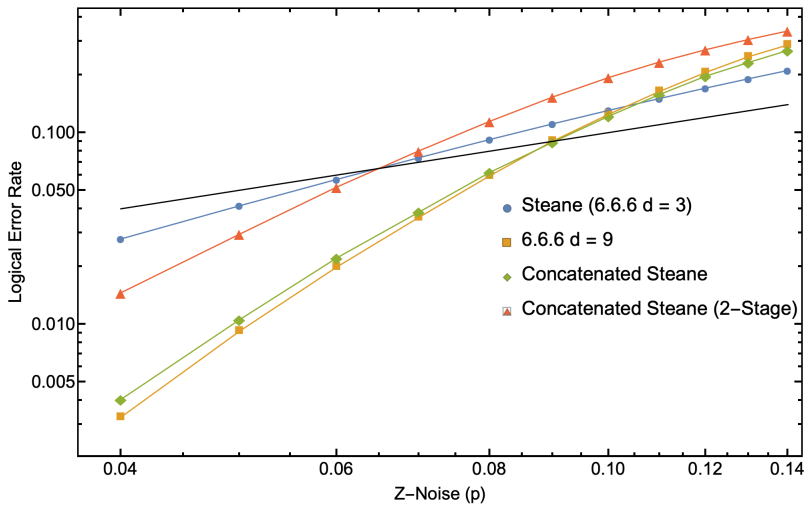


Figure 2.10: Simulated logical error rates for Example 9. The black line is $y = x$.

codes was consumed by distributing the trellis to all the processors. We have chosen to use this chapter as a baseline for trellis theory, including its limits. Reporting all improvements to the theory or its practical implementation has been relegated to the next chapter. The special structure of some codes may be utilized to reduce $|E|$ without the need for further theory, as demonstrated by the next example.

Example 9. Let H denote the X and Z stabilizer matrix for the $[[7, 1, 3]]$ Steane code,

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

The level-2 concatenated Steane code is the $[[49, 1, 9]]$ code whose X or Z stabilizers are given by the Pauli operators corresponding to $I \otimes H$ and $H \otimes \vec{1}$, where I is the 7×7 identity matrix and $\vec{1}$ is the length-7 all-ones vector. The trellis for this code has $|V| = 626$ and $|E| = 844$. Using Theorem 2.4.4 and the subsequent discussion, we consider a 2-stage, suboptimal decoder. The seven copies of H are decoded independently at a cost of 36 edges each. The resulting corrections are made then the syndrome updated and the bits

corresponding to $H \otimes \vec{1}$ are again decoded using the trellis for H. The correction returned from this trellis is tensored with $\vec{1}$ to map back to the original problem. Since the first seven decodes can be done simultaneously, the time complexity of this decoder is determined by only $36 + 36 = 72$ edges, a 91.5% savings. Numerical simulations similar to the ones above were performed for both this decoder and the normal, single stage trellis. Results are presented in Figure 2.10 where we see that this is indeed suboptimal, trading speed for accuracy.

The calculated pseudothreshold for the independent Z channel between the Steane code and concatenated Steane code using 2-stage block decoding is 6.44%, in agreement with the Z threshold of 6.46% estimated from the depolarizing channel threshold of 9.69% [194]. The calculated Z pseudothreshold between the Steane and the concatenated Steane code using the full trellis is 10.55%. This is known to be below the true threshold for continued concatenation based on message passing of 12.5% estimated from a depolarizing channel threshold of 18.8% [195]. The $[[7, 1, 3]]$ Steane code can also be considered a distance 3 6.6.6 color code. It is informative to compare the performance of the $[[61, 1, 9]]$ 6.6.6 color code to the level-2 concatenated Steane code. We see that the color code has a slightly better performance at the cost of 12 more qubits.

The numerical threshold estimates obtained in this work are competitive with, and sometimes even better than, current state-of-the-art decoders. We attribute this success to two main things. First, with the exception of organizing the data into logical cosets, the decoder is provided with close to the maximum amount of information possible of both the code and the error channel. Traditional decoders often only rely on a fraction of this information. Second, since every element of $\mathcal{P}_n$ represented by a path in the trellis has the same syndrome as was measured, any correction returned by the decoder will, at minimum, force the resulting state to have zero syndrome. If there are few or no logical operators of the weight of the true error, the decoder will therefore correct the majority of the correctable

errors of that weight regardless of the minimum distance of the code.

CHAPTER 3

APPLICATIONS OF TRELLISES TO QUANTUM ERROR CORRECTION

3.1 Introduction

In the previous chapter we established trellis decoding as a powerful technique for QECCs. The success of this decoding scheme comes at the price of speed and storage for codes with moderate values of combinations of the parameters n, k, and d. In order to be considered among practical decoders, it remains to reduce the computational complexity of trellis decoding in both space and time.

We start with a review of the classical theory of sectionalization and show how it applies to stabilizer codes. This merges edge sections E_i to reduce $|E|$, which saves both storage space and algorithmic runtime. We prove that sectionalized trellises can be computed using the algorithm of the previous chapter. We then show how discarding some information about the trellis allows for an often considerably smaller storage representation of the trellis using the adjacency matrix of the graph. The Viterbi algorithm is recast for this format. Having reduced space and time for the algorithms, we proceed to reduce the initial cost of trellis construction by further investigating the trellis product. We begin by classifying it in terms of the standard graph products, then use this to provide an efficient method of constructing a trellis using matrix multiplication of the adjacency matrices of subtrellises. A thorough analysis of this construction process offers new insights on previous results and completes the mathematical analysis of the internal structure of the graphs. This new understanding allows us treat the graph as a fundamental representation of the code, and we show how to decompose a syndrome trellis into subtrellises corresponding to individual generators in TOF.

Having increased the efficiency of and improved our understanding of trellises, we

move on to applications. We begin by switching the vertices and edges with respect to Chapter 2 and show how a well-known result from graph theory applied to the adjacency matrix of the trellis can be used to solve the degenerate decoding problem for any stabilizer code. To the author's knowledge, this is the first application of algebraic graph theory in QEC. Using the same technique to separate cosets, we show how to use the trellis to compute the minimum distance of both classical and stabilizer codes. This is a simple application of the Viterbi algorithm, and if the trellis has already been constructed for a previous purpose, the computation is essentially free. We compare this to state-of-the-art algorithms. Along a similar line, we show how to use the trellis to compute various weight enumerators and the set of elements of bounded weight. A new weight enumerator for stabilizer codes is introduced which takes the phases of elements $\eta^c X(a)Z(b)$ into account, and a new signed trellis is introduced to compute it.

We will continue to think of trellises as representations of codes in this chapter, but there is nothing stopping us from considering trellises as representations of appropriate modules. Codes are themselves are nothing more than finitely-generated $\mathbb{F}_p$-submodules of $\mathbb{F}_p^n$ with a finitely-generated dual taken with respect to a given annihilator (see Section 1.6). We have not pursued this generality here to make the results accessible to the widest audience, but with some work the proofs can be modified for this new language. Considering a trellis as a compact representation of every element of an appropriate module M, the previously mentioned topics show how to extract n, $|M|$, and a finite generating set.

Although we will not discuss it here, the techniques discussed in this chapter, in particular the minimum distance, words of minimum weight, and coset decoding, are the basis for attacks on cryptographic systems based on error-correcting codes.

3.2 Sectionalization

Sectionalization is the process in which multiple edge sections of the trellis are merged into one. The merging of sections E_i to E_j will be denoted by $E_{i:j}$, and indices i and $j+1$ are

called the section boundaries. This is a common technique in classical trellis theory and the literature is vast. We review the necessary ideas here in our notation and refer the reader to (e.g.) [189] for further details. To demonstrate the basic idea, consider the trellis for the $[[7, 1, 3]]$ Steane code ($d = 3$ color code) in Figure 2.6. The vertex and edge profiles are given in Table 2.3 and 2.4, respectively. As can be seen from both the figure and the tables, $|V_i|$ hits a maximum when the incoming and outgoing vertex degrees are both one. By merging E_4 and E_5 into a single section, $E_{4:5}$, where each edge is now labeled by a tuple of elements of $\mathcal{P}_1$, 64 vertices and edges are removed from the trellis for a 52% reduction in vertices and 27% in edges.

The Viterbi algorithm runs in $\Omega(|E|)$ time, so we are interested in sectionalization as a means to decrease $|E|$, as seen in the example above. Consider removing a vertex $v \in V_i$. The incoming and outgoing edges associated with this vertex must be combined into $\deg_{\text{in}}(v) * \deg_{\text{out}}(v)$ edges in the sectionalized trellis. This can increase $|E|$; for example, $E_{1:n}$ is simply a lookup table for $\mathcal{S}^{\perp}$. However, as observed in the example when both degrees were one, this can also decrease $|E|$. Sectionalizing larger trellises may result in a significant speedup in decoding if done properly and may slow down decoding if not.

Let $\{0 = h_0, h_1, \ldots, h_L = n\}$ be a set of ordered section boundaries. The case $h_i = i$ for all i corresponds to an unsectionalized trellis. By construction, vertex sets are either kept or removed but never modified, so the vertex formula remains the same

$$|V_{h_i}| = p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{h_i}} - \dim \mathcal{S}^{\perp}_{\mathfrak{f}_{h_i}}}.$$

Likewise, the number of edges becomes

$$|E_{h_{i-1}:h_i}| = p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{h_{i-1}}} - \dim \mathcal{S}^{\perp}_{\mathfrak{f}_{h_i}}}.$$

Let $\{S_\ell \mid h_{i-1} < L(S_\ell), R(S_\ell) < h_i\}$ be the set of stabilizer generators active only between two section boundaries. These paths will originate from $\overline{0} \in V_{h_{i-1}}$ and terminate at $\overline{0} \in V_{h_i}$;

hence, each edge in $E_{h_{i-1}:h_i}$ will have $p^{|\{S_\ell\}|}$ parallel edges. These may be thought of as subtrellises and exploiting these substructures has been useful in classical trellis theory. The degrees are given by

$$\deg_{\text{in}}(v) = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{p}_{h_i}} - \dim \mathcal{S}^{\perp}_{\mathfrak{p}_{h_{i-1}}}},$$
$$\deg_{\text{out}}(v) = p^{\dim \mathcal{S}^{\perp}_{\mathfrak{f}_{h_i}} - \dim \mathcal{S}^{\perp}_{\mathfrak{f}_{h_{i+1}}}}.$$

It remains to show that we can construct the sectionalized trellis efficiently. Since we are introducing this concept as a way to reduce storage space (and therefore time), it would be preferable to construct this without first constructing the full trellis then merging. The procedure of the previous chapter relied on the bipartite, complete structure of the edge configurations. The edge configurations of Table 2.1 become increasingly complex during the merging process and we do not attempt to enumerate them here. The next theorem shows that the merging of edge configurations is still bipartite, complete.

The proof of this can be taken in two ways. First, ignore the fact that sections E_i to E_j exist and simply define $E_{i:j}$ as the fundamental object. The proofs of the last chapter show that this is a group with subgroup the zero syndrome. The edge configuration proof then holds by replacing labels P with an appropriately defined ordered set of labels. On the other hand, one could treat the full trellis as fundamental and then view the proof as a merger of individual E_i. The latter is more complicated as one needs to define the group $E_{i:j}$ in terms of $E_i, \ldots, E_j$. Naïvely one could suggest $E_{i:j} = E_i \times \ldots \times E_j$, but this does not hold as edges $(e, e') \in E_i \times E_{i+1}$ might not have an appropriate terminus-source pairing to be a valid edge in the trellis. Viewed in this manner $E_{i:i+1} = \{ee' \mid e \in E_i, e' \in E_{i+1}, t(e) = s(e')\}$. It is easy to see that this set satisfies the group axioms because E_i and E_{i+1} do.

Theorem 3.2.1 (Merger Viewpoint)

Consider two edge sections E_i and E_{i+1} of a syndrome trellis with edge configurations

in Table 2.1. Let $e \in E_{i:i+1}$ *and define* $\mathcal{I}_i = \{v \in V_i \mid \exists e' \in E_{i:i+1}, t(e') = t(e)\}$ *and* $\mathcal{I}_{i+2} = \{v \in V_{i+2} \mid \exists e' \in E_{i:i+1}, s(e') = s(e)\}$. *Then the vertices of* $\mathcal{I}_i$ *and* $\mathcal{I}_{i+2}$ *form a completely-connected bipartite graph in* $E_{i:i+1}$ *and no other elements of* $V_i \backslash \mathcal{I}_i$ *or* $V_{i+2} \backslash \mathcal{I}_{i+2}$ *are connected to the vertices in* $\mathcal{I}_{i+2}$ *and* $\mathcal{I}_i$*, respectively. If there exists a parallel edge in* $E_{i:i+1}$ *then all edges in* $E_{i:i+1}$ *are parallel with the same number of edges in parallel.*

Proof. Let $e = E_{i:i+1}$ be arbitrary. Then there exists two edges $e_i = (a, P_{ax_1}, x_1) \in E_i$ and $e_{i+1} = (x_1, P_{x_1b}, b) \in E_{i+1}$ such that $e = e_i e_{i+1}$. The sets $\mathcal{I}_i$ and $\mathcal{I}_{i+2}$ are taken with respect to b and a, respectively. Let $c \in \mathcal{I}_{i+2}$ and $d \in \mathcal{I}_i$. We need to show there exists an edge in $E_{i:i+1}$ with source c and terminus d. By definition of $\mathcal{I}$, the exists edges $e_{cb} = (c, P_{cx_2}, x_2)(x_2, P_{x_2b}, b)$ and $e_{ad} = (a, P_{a,x_3}, x_3)(x_3, P_{x_3d}, d)$. Then

$$\begin{aligned}
&(c, P_{cx_2}, x_2)(x_2, P_{x_2b}, b)(-a, P_{ax_1}^{-1}, -x_1)(-x_1, P_{x_1b}^{-1}, -b)(a, P_{a,x_3}, x_3)(x_3, P_{x_3d}, d) \\
&= (c, P_{cx_2} P_{ax_1}^{-1} P_{a,x_3}, x_2 - x_1 + x_3)(x_2 - x_1 + x_3, P_{x_2b} P_{x_1b}^{-1} P_{x_3d}, d)
\end{aligned}$$

is the desired edge. As in Theorem 2.3.12, we know these inverses exist. We need to show that for $f \in V_i \backslash \mathcal{I}_i$ there does not exist a path from f to b and that for $g \in V_{i+2} \backslash \mathcal{I}_{i+2}$ there does not exist a path from a to g. The arguments are similar to the above and that in Theorem 2.3.12, so we omit the details here. The last statement similarly follows from Theorem 2.3.12. □

Corollary 3.2.2

The edge configurations of a sectionalized trellis are completely-connected, bipartite graphs.

The proof of this follows from repeated applications of the previous theorem.

The proof of the theorem shows that the edge configurations of a sectionalized trellis do not merge in such a way that their sizes grow exponentially. Thus, sectionalization shows that the edge configurations of E_{i+1} cannot randomly connect to E_i, but instead the combinatorics of the vector space cause them to align.

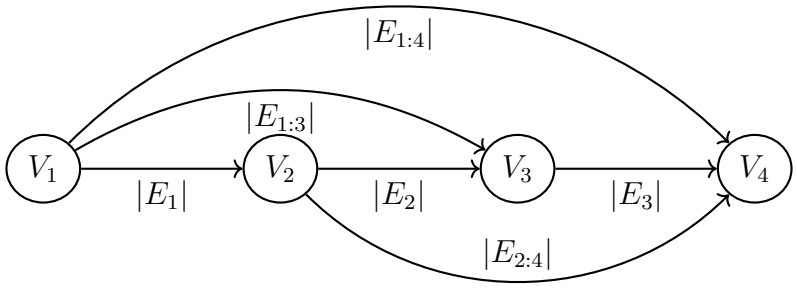


Figure 3.1: The optimal sectionalization graph from [196] for a four section trellis.

Corollary 3.2.3

The number of disjoint edge configurations in $E_{h_{i-1}:h_i}$ *is* $p^{\dim \mathcal{S}^\perp - \dim \mathcal{S}^\perp_{\mathfrak{p}_{h_{i+1}}} - \dim \mathcal{S}^\perp_{\mathfrak{f}_{h_i}}}$.

An optimal sectionalization is a trellis with the lowest $|E|$ taken over all possible sectionalizations. The optimal may not be unique, and it may the case, likely only for small examples, that the optimal sectionalization requires no merging at all. Since the objective of this trellis is decoding, we require that V_0 and V_n are fixed as section boundaries. Lafourcade and Vardy pointed out that the straightforward greedy algorithm to determine the optimal sectionalization of a trellis may be expressed as a graph algorithm [196]. Make an $n+1$ vertex graph arranged linearly and number them from left to right by $0, 1, \ldots, n$. Connect vertices 0 and 1 by a directed edge with weight E_1. Connect vertices 0 and 2 by a directed edge with weight $|E_{1:2}|$. Repeat this for all pairs of vertices $0 \leq i \leq j \leq n$ weighting with the value $|E_{i:j}|$, where $|E_{i:i}| = |E_i|$. The optimal sectionalization is then given by the minimum-weight path from vertex 0 to vertex n, which may be again found by the Viterbi algorithm. See Figure 3.1 for an example.

Heuristically, these kinds of basic optimizations should arguably always be made, although this may complicate computer implementations of algorithms. This should be done on a code-by-code basis. We will not explicitly work with sectionalized trellises in the rest of this chapter but instead implicitly assume the trellis is in the desired form.

3.3 The Adjacency Matrix Representation

A graph $\mathcal{G} = (V, E)$ consists of a set of vertices, V, and edges, E. It is bipartite if the vertices can be partitioned into disjoint sets $V = V_1 \sqcup V_2$ such that every edge has one end in V_1 and the other in V_2. A path of length ℓ from vertices v to v' is a sequence of ℓ distinct vertices starting with v and ending with v' such that each consecutive vertices are connected by an edge. A graph is equipathic if all paths with common endpoints have equal length. A cycle is a path where $v = v'$. A graph is acyclic if it contains no cycles. It will be convenient in this work to consider paths as sequences of edges instead of vertices. Edges whose vertices lack a distinct ordering are called undirected and those with an ordering directed. Undirected edges represent a symmetric relation - v is connected to v' implies v' is connected to v - while directed edges do not. Directed edges, also called arcs, are denoted with an arrow starting from the source and ending at the terminus. A graph whose edges are directed is called a directed graph, or a digraph for short. A digraph is weakly connected if there exist a path between any two vertices on the graph obtained by making all directed edges undirected. All edges in this work are directed, and so from here on out we will drop the adjective directed and "graph" will always mean "digraph". For a given vertex v, the number of edges whose terminus is v is called the in(coming) degree, and the number of edges whose source is v the out(going) degree. A vertex with in-degree zero is called a source and one with out-degree zero a sink. Finally, a graph is graded, or layered, if there exists a rank function, φ, on the vertices which can be used to construct a partial order on the edges by evaluating φ on the sources and termini.

Trellises are a strict structure and are slightly boring from an algebraic graph theoretical perspective. They are weakly-connected, equipathic, graded, acyclic, directed graphs with a unique source (V_0) and sink (V_n). The grading is known by construction and is given by the rank function $\varphi : V \to \{0, \ldots, n\}$ such that $\varphi(v) = i$ if and only if $v \in V_i$. Trellises have trivial spectrum, a major focus of algebraic graph theory, and all random

walks converge to V_n. Considered as a network, flow problems currently do not appear physically relevant.

An adjacency matrix of a graph $\mathcal{G}$, $A = A(\mathcal{G})$, is a $|V| \times |V|$ integer matrix with rows and columns indexed by V whose ij-th entry is equal to the number of edges from vertex i to j. If $\mathcal{G}$ is undirected, A is symmetric. All graphs in this work are simple graphs meaning no edge has the same source and terminus, i.e., there are no self-loops. Thus the diagonal entries of A are zero. If the edges of a graph are weighted, a weighted adjacency matrix of the graph has ij-entries equal to the (sum of the) weight(s) of the edge(s) from i to j. For the moment we will only deal with integer matrices. We will need the following well-known result, which we record as two separate statements solely for added clarity.

Lemma 3.3.1

(i) Let A be an (integer) adjacency matrix for a connected graph. The ij-th entry of A^m is the number of paths from vertex i to vertex j of length m.

(ii) Let A be a weighted adjacency matrix for a connected graph. The ij-th entry of A^m is the sum of the products of all weights of all paths from vertex i to vertex j of length m.

For the rest of this chapter, let A be an adjacency matrix for a trellis representing an error-correcting code, classical[1] or quantum. Considering the size of $|V|$ for the codes of Chapter 2, it is not clear A is a more useful representation of the trellis. To see why it might be, we need to understand the structure of the matrix. Begin by assigning an ordering to the vertices. The natural way to do this is to enumerate the vertices in the order $V_0, V_1, \ldots, V_n$, where the order in each V_i is arbitrary. Then vertex v_i connects to v_j only if $i < j$ with respect to this ordering. (This is another way of saying the trellis is graded, acyclic, and directed.) Hence A is upper triangular with zeroes along the diagonal with respect to this ordering. Since edges only exist from V_i to V_{i+1}, the adjacency matrix exhibits a block

[1]To build a trellis for a classical code, use the parity-check matrix in the role of $\mathcal{S}^{\perp}$ in Chapter 2.

structure of sizes $|V_i| \times |V_{i+1}|$, the ratio of which are determined by $\deg_{\text{in}}$ and $\deg_{\text{out}}$. Since $|V_i|/|V|$ is usually much less than one, A is typically sparse.

Example 10. The adjacency matrix for the trellis in Figure 2.1 is

$$A = \begin{pmatrix}
0 & 1 & 1 & 1 & 1 & & & & & & & & & & & \\
 & 0 & & & & 1 & & 1 & & & & & & & & \\
 & & 0 & & & & 1 & & 1 & & & & & & & \\
 & & & 0 & & 1 & & 1 & & & & & & & & \\
 & & & & 0 & & 1 & & 1 & & & & & & & \\
 & & & & & 0 & & & & 1 & 1 & & & & & \\
 & & & & & & 0 & & & 1 & 1 & & & & & \\
 & & & & & & & 0 & & & & 1 & 1 & & & \\
 & & & & & & & & 0 & & & 1 & 1 & & & \\
 & & & & & & & & & 0 & & & & 1 & 1 & \\
 & & & & & & & & & & 0 & & & 1 & 1 & \\
 & & & & & & & & & & & 0 & & 1 & 1 & \\
 & & & & & & & & & & & & 0 & 1 & 1 & \\
 & & & & & & & & & & & & & 0 & & 2 \\
 & & & & & & & & & & & & & & 0 & 2 \\
 & & & & & & & & & & & & & & & 0
\end{pmatrix}.$$

There is no widely agreed upon precise definition, but a square matrix is generally considered sparse if the number of nonzero entries is roughly the number of rows (columns). The ratio of nonzero entries to total entries is called the density of the matrix. It immediately follows from its upper triangular form that the density of the adjacency matrix of a trellis is less than 50%. Tables 2.3 and 2.4 of Chapter 2 provide vertex and edge counts, respectively, for the 4.8.8 and 6.6.6 color codes, the rotated surface codes, and their CSS splittings. The density of the corresponding adjacency matrices are $|E|/|V|^2$ and are given

in Table 3.1. A better understanding of the scaling of these variables for QECCs could allow for more analytical statements to be made about the density of A.

Table 3.1: Adjacency matrix densities $|E|/|V|^2$ by distance for the codes in Tables 2.3 and 2.4.

	3	5	7	9	11	13	15	17	19	21
4.8.8	1.6e-2	4.3e-4	2.1e-5	7.9e-7	1.9e-7	1.5e-8	7.8e-10	4.6e-11	5.8e-13	1.6e-13
4.8.8 X/Z	5.3e-2	6.0e-3	9.9e-4	1.7e-4	6.7e-5	2.0e-5	4.1e-6	9.0e-7	2.0e-7	4.7e-8
6.6.6	1.6e-2	7.3e-4	3.6e-5	3.4e-6	3.5e-7	3.0e-8	2.8e-9	3.0e-10	4.0e-11	5.0e-12
6.6.6 X/Z	5.3e-2	8.2e-3	1.4e-3	3.4e-4	9.3e-5	2.8e-5	7.7e-6	2.1e-6	7.7e-7	2.6e-7
RSurf	2.8e-2	1.8e-3	1.9e-4	2.7e-5	4.2e-6	7.3e-7	1.3e-7	2.5e-8	5.0e-9	1.0e-9
RSurf X	6.2e-2	1.2e-2	3.7e-3	9.4e-4	3.1e-4	1.1e-4	4.0e-5	1.6e-5	6.1e-6	2.5e-6

An immediate consequence of Lemma 3.3.1 (i) combined with the structure of the adjacency matrix is that A is nilpotent of order $n+1$. Each successive power of A becomes more sparse and at A^n there is only a single nonzero value in $A_{0,n}$ equal to the number of paths between V_0 and V_n. By the one-to-one correspondence of paths and elements in Proposition 2.3.1, this is also equal to the cardinality of the code. This can be used to check the trellis was constructed correctly.

There are two immediate questions concerning the computation of A^n required in the above algorithm: can it be done efficiently and is it numerically stable (for the weighted case)? The answer to the second question is yes, and the interested reader is referred to standard texts on numerical analysis. The answer to the first question comes in several pieces. First note that A^n can be done in $\Omega(\log n)$ matrix-matrix multiplications using recursive doubling. For example, to compute A^9, first compute $A^2 = AA$, then $A^4 = A^2A^2$, and $A^8 = A^4A^4$; finally, $A^9 = A^8A$. If n is a power of two, this terminates in exactly $\log_2 n$ steps. Otherwise, compute the $\lfloor \log_2 n \rfloor$ squares then use the binary representation of n to break A^n into a product of powers of two of A. In the worse case, the binary representation is all ones and an extra $\log_2 n - 1$ products need to be done for a total of $2\log_2 n - 1$ matrix-matrix multiplications. For example, for a code of length $n = 500$, compute that $500 \equiv 111110100_2$, so $A^{500} = A^{256}A^{128}A^{64}A^{32}A^{16}A^4$ for a total of 13 matrix-matrix multiplications.

The second piece regarding the efficiently of computing A^n is that sparse matrices may be stored in special formats such as the compressed sparse row/column formats (CSR and CSC, respectively), which are taken advantage of by sparse matrix algorithms. In these formats, matrix multiplication is essentially a loop over an array of the nonzero values of the matrix. Since A^{256} has fewer nonzero values than A, computing $A^{256}A^{256}$ is a faster than, say, AA. The associativity of the decomposition of A^n into matrix products should take this into account. GPUs may be useful in some matrix multiplications, but at a certain point the extreme sparsity of the matrix powers may make the overhead involved in this process an unnecessary bottleneck. As a final comment, we remark that the block compressed sparse row (BCSR) format may or may not be desirable depending on whether or not the underlying implementation requires the size of every block ($|V_i| \times |V_{i+1}|$) to be the same, as is common. The CSC is used in this work.

If not implemented properly, the adjacency-matrix representation of the trellis can lose vital information about the graph. For example, in the worst case scenario that multiple errors have equal weight, edge labels cannot be inferred from the weights so additional information must be kept to handle shifting and decoding. This can be done at the cost of an extra byte per edge, and is still a considerable savings over the standard 4-byte pointers on 32-bit systems and 8-byte pointers on 64-bit systems used in a straightforward linked-list based implementation of the trellis or the integer-based addressing scheme used in the previous chapter. We find it convenient to construct the trellis in full generality with vertex and edges structures holding numerous pieces of information and then cast down to the adjacency matrix for speed if necessary.

It remains to show how shifting and the Viterbi algorithm can be done using just the adjacency matrix. We will describe these separately but note that these can be combined into a single shift-and-decode step on each E_i, reducing the number of memory accesses to this data. The benefit of this depends on the specific choice of parallelism in the implementation of the algorithms but in all scenarios considered here was indeed faster. We

will assume that the edge labels can either be inferred from the weights or a modified-CSC format is used which allows them to be stored with each entry. Shifting merely requires one to keep track of the matrix indices corresponding to each V_i. The Viterbi algorithm is presented in this format in Algorithm 2 for completeness.

Algorithm 2: Adjacency matrix version of the Viterbi algorithm.

Input: A - An adjacency matrix for a trellis
Input: n - The depth of the trellis
Output: Correction vector $\in F_p^n$

$\mu s \in \mathbb{R}^{1\times|V|}$ - vertex weights
$prev \in \mathbb{N}^{1\times|V|}$
$\mu s[1] \leftarrow 0.0$
$prev[1] \leftarrow 0$
for $i \leftarrow 2$ ***to*** $\mathbf{ncols}(A)$ **do**
    $vals, rows \leftarrow \mathbf{nonzeros}(A[:, v]), \mathbf{rows}(A[:, v])$
    for $j \leftarrow 1$ ***to*** $|vals|$ **do**
        $vals[j] \leftarrow vals[j] + \mu s[row[j]]$
    end
    $min, arg \leftarrow \mathbf{argmin}(vals)$
    $\mu s[v], prev[v] \leftarrow min, rows[arg]$
end
$path \in \mathbb{F}_{p^2}^{1\times n}$
$v_i \leftarrow \mathbf{ncols}(\mathbf{A})$
$v_{i-1} \leftarrow prev[v_i]$
for $i \leftarrow n$ ***to*** 1 ***by*** -1 **do**
    $path[i] \leftarrow$ edge label for $A[v_{i-1}, v_i]$
    $v_i \leftarrow v_{i-1}$
    $v_{i-1} \leftarrow prev[v_i]$
end
return $path$

Remark: The adjacency matrix is an elementary and unoriginal tool to apply to a graph. It is therefore interesting to find it rarely mentioned in the classical trellis literature (outside of convolutional codes). We assume this is largely due to the fact that implementation details are usually ignored in theoretical publications. We mention it here for two reasons.

First, the previous chapter might leave the impression that the algorithm is completely unscalable, in which case we explicitly show how to reduce the overhead. Second, we have further computational uses for the adjacency matrix than just representing the graph in a compact representation. It is also interesting that the 2011 paper “Matrix Representations Of Trellises And Enumerating Trellis Pseudocodewords" introduced a matrix they called an adjacency matrix for a slightly different concept called a tail-biting trellis whose elements are multivariate polynomials [197]. This is entirely distinct from the common use of the term but allowed them to define new invariants. They also come up with a similar formula for a trellis product (below) but, again, it is unrelated to our use or derivation.

3.4 The Trellis Product

The TP introduced in Definition 2.4.2 has been studied in a few papers, but never in the context of a syndrome-based trellis. Many of the results and diagrams in the literature do not hold for the trellises considered here. Likewise, the results derived here are not valid for those trellises. Unraveling the definition, its consequences, and its inverse will further illuminate and complete our discussion of the internal structure of the trellis.

Recall the definition from 2.4.2.

Definition 3.4.1

The TP of two trellises $\mathcal{T}^1 = (V^1, E^1)$ and $\mathcal{T}^2 = (V^2, E^2)$ of depth n is denoted by $\Box_i$ and has vertices

$$V^1 \Box_i V^2 = \bigcup_{i=0}^{n} V_i^1 \times V_i^2$$

and edges

$$E^1 \Box_i E^2 = \{((v_i^1, v_i^2), P^1 P^2, (v_{i+1}^1, v_{i+1}^2)) \mid (v_i^1, P^1, v_{i+1}^1) \in E_i^1, (v_i^2, P^2, v_{i+1}^2) \in E^2\}.$$

The trellises $\mathcal{T}^1$ and $\mathcal{T}^2$ are called factors of the product. There are 256 ways to define the product of graphs in algebraically reasonable ways, only six of which are commutative,

associative, and have a unit. There are exactly four graph products in which at least one of the projections onto its factors is a weak-homomorphism [198]. These are the four standard graph products: Cartesian $\square$, direct $\times$, strong $\boxtimes$, and lexicographical $\circ$. Let $\mathcal{G}^1 = (V^1, E^1)$ and $\mathcal{G}^2 = (V^2, E^2)$ be connected graphs. The vertex set of all the graph products is the Cartesian product $V^1 \times V^2$; they are distinguished by the edges. For example, the directed Cartesian product $\mathcal{G}^1 \square \mathcal{G}^2$ has edge set

$$E(\mathcal{G}^1 \square \mathcal{G}^2) = \{(g^1, g^2) \to (h^1, h^2) \mid g^1 = h^1, g^2 \to h^2 \in E^2 \text{ or } g^1 \to h^1 \in E^1, g^2 = h^2\},$$

where $a \to b$ represents a directed edge with source a and terminus b.

We note immediately that there are easy isomorphisms showing the TP is both commutative and associative, $\mathcal{T}^1 \square_i \mathcal{T}^2 \cong \mathcal{T}^2 \square_i \mathcal{T}^1$ and $\mathcal{T}^1 \square_i (\mathcal{T}^2 \square_i \mathcal{T}^3) \cong (\mathcal{T}^1 \square_i \mathcal{T}^2) \square_i \mathcal{T}^3$, and has a unit, $\mathcal{I}$, the trellis for the identity of $\mathcal{P}_n$. This last point will become obvious in our exploration as will the projections onto factors, but for the moment we point out that although no attempt has been previously made to classify the TP, it has been incorrectly referred to as a Cartesian product or a tensor product. The Cartesian product is only *technically* incorrect as the Cartesian product of two trellises would contain $|V^1||V^2|$ vertices (with no edges) combining, for example, V_0^1 and V_n^2, which doesn't happen in the TP. For the CSS Steane code trellis of Figure 2.6 (a), this would give a full Steane code trellis with 676 vertices, far greater than the 122 vertices of Figure 2.6 (b). Comparing to Definition 2.4.2, we see that the TP is really a graded, directed Cartesian product taken independently over each depth/section.

Proposition 3.4.2

Let $\mathcal{T}^1 = (V^1, E^1)$ and $\mathcal{T}^2 = (V^2, E^2)$ be trellises of depth n, and let $\mathcal{G}_i$ be the subgraph $(V_i \cup V_{i+1}, E_i)$. Then $\mathcal{T}^1 \square_i \mathcal{T}^2 = \bigcup_{i=0}^{n} \mathcal{G}_i^1 \square \mathcal{G}_i^2$.

The current TP literature is devoted to trellis construction using smaller trellises (e.g. [186]), similar to our discussion of trellises for CSS codes in Section 2.4. However, no

explicit formula is given to aid in this process, reducing the usefulness of the procedure to those trellises for which one can manually draw and combine or brute-force loop over on a computer. Our identification of the TP with the Cartesian product allows us to fill this gap. It is well-known that the adjacency matrix of the Cartesian product is the Kronecker product of the adjacency matrices of each factor, $A(\mathcal{G}^1 \square \mathcal{G}^2) = A(\mathcal{G}^1) \otimes A(\mathcal{G}^2)$.

Corollary 3.4.3

Let $\mathcal{T}^1 = (V^1, E^1)$ and $\mathcal{T}^2 = (V^2, E^2)$ be trellises of depth n, and let $\mathcal{G}_i$ be the subgraph V_i, E_i, V_{i+1}. The adjacency matrix $A(\mathcal{T}^1 \square_i \mathcal{T}^2)$ is the $\sum_{i=0}^{n} |V_i^1||V_i^2| \times \sum_{i=0}^{n} |V_i^1||V_i^2|$ upper triangular matrix with blocks of size $|V_i^1||V_i^2| \times |V_{i+1}^1||V_{i+1}^2|$ given by $A(\mathcal{G}_i^1) \otimes A(\mathcal{G}_i^2)$.

Note that this relies on the assumption that the individual Cartesian products are taken consistently in a manner which does not rearrange the vertices between $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$. The Kronecker product is associative and commutative up to row and column permutations. Trellises created in this manner essentially treat vertices as tuples $(v_i^1, v_i^2) \in V_i^1 \times V_i^2$. This is interesting as it allows us to view trellises as higher-dimensional structures. However, vertices of the syndrome trellis are defined with labels $v_i^1 + v_i^2$, and it remains to show that this does not collapse the labels into less than $|V_i^1||V_i^2|$ vertices. We will return to this below where we provide conditions for when this happens.

The corollary is meant to be understood structurally; the edge labels need to be computed separately at each element of the Kronecker product if using an additive representation of $\mathcal{P}_n$ such as $\mathbb{F}_{q^2}^n$ or $\mathbb{F}_q^{2n}$. Information about vertex labels is lost using the Kronecker product but isn't necessary to store anyway. Labels can always be recreated by simply taking the partial syndrome of any path leading up to a given vertex. This is most likely the quickest way to construct a large trellis, beating out even the parallel algorithm of Chapter 2. The Kronecker product also has a specialized implementation for sparse matrix formats.

Remark: The TP was originally described as a "product of edges", which makes sense from the definition. It is interesting then that the formula for the TP is given in terms of

the adjacency matrix from the vertex perspective and not from the incidence matrix from the edge perspective. This is an $|E^1| \times |E^2|$ matrix with -1 and each source and 1 at each terminus. The TP product from this perspective must first split the matrices into positive and negative parts, then the Kronecker product of the two positive parts and of the two negative parts are taken separately before recombining them, being careful to remove any introduced self-loops.

With these preliminaries out of the way, we move to exploring the structure of the syndrome trellis using the TP. For this we will need to define an ET. We will again stick to trellises for stabilizer codes, noting that the same intuition and proofs hold classically as well. This will hold for both trellises with vertices $\mathcal{S}$ and edges $\mathcal{S}^\perp$ and trellises with vertices $\mathcal{S}^\perp$ and edges $\mathcal{S}$ (Section 3.6), so we will switch to using $\mathcal{A}$ and $\mathcal{A}^\perp$, where either $\mathcal{A} = \mathcal{S}$ or $\mathcal{A} = \mathcal{S}^\perp$ as desired.

Definition 3.4.4

Let $\mathcal{A} = \langle A_1, \ldots, A_\ell \rangle \leq \mathcal{P}_n$ with generators in TOF. An ET is a syndrome trellis for a single generator $\langle A_j \rangle$ with vertices taken with respect to $\mathcal{A}^\perp$.

Our goal is to build a trellis for $\mathcal{A}$ via $\mathcal{T}^1 \square_i \ldots \square_i \mathcal{T}^\ell$, where $\mathcal{T}^j$ is the ET for generator A_j.

Example 11. Figure 2.6 (a) gives the CSS trellis for the Steane code. This has vertices with respect to the stabilizer matrix

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

and edges

$$S_1^\perp = [1, 1, 1, 0, 0, 0, 0]$$

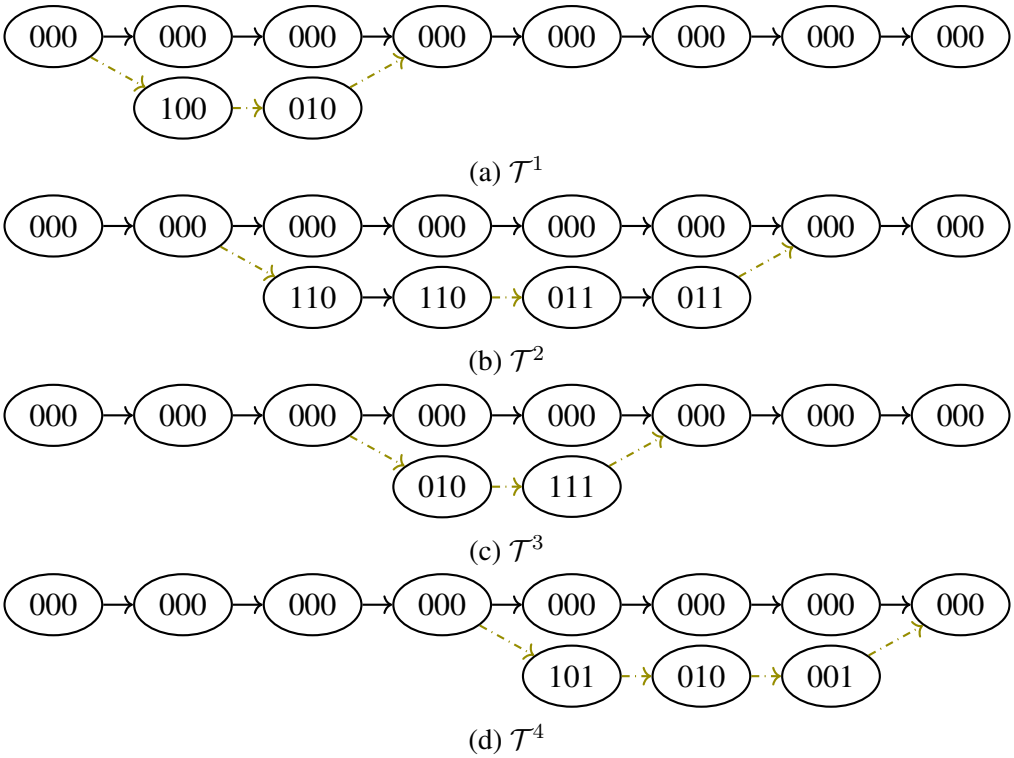


Figure 3.2: ETs for the CSS Steane code.

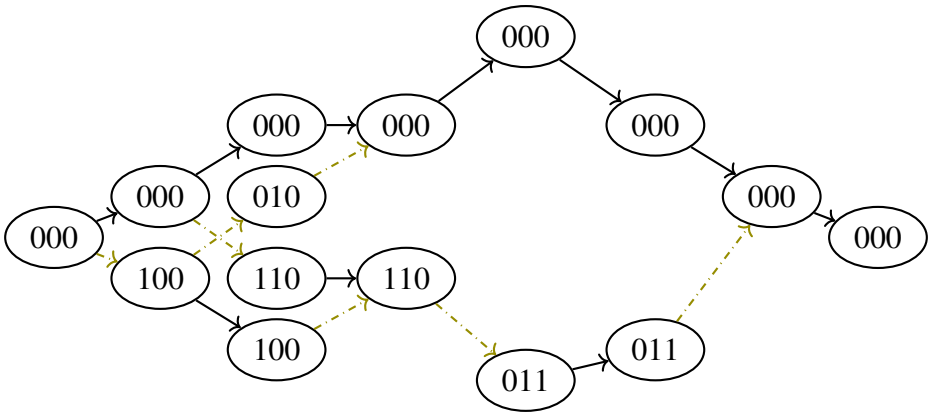
$$S_2^\perp = [0,1,0,1,0,1,0]$$

$$S_3^\perp = [0,0,1,1,1,0,0]$$

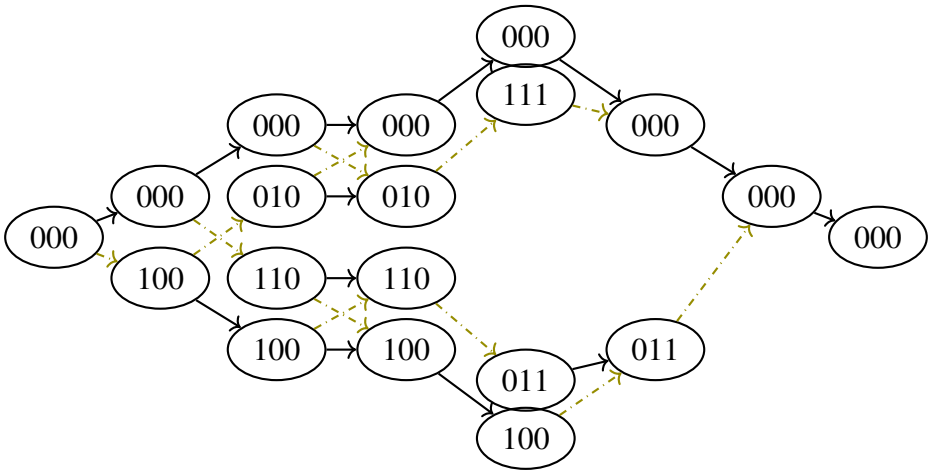
$$S_4^\perp = [0,0,0,1,1,1,1].$$

Letting $\mathcal{T}^j$ denote the ET for $S_j^\perp$, the ETs are given in Figure 3.2a. Figure 3.3 shows the construction of the full CSS trellis with the TP.

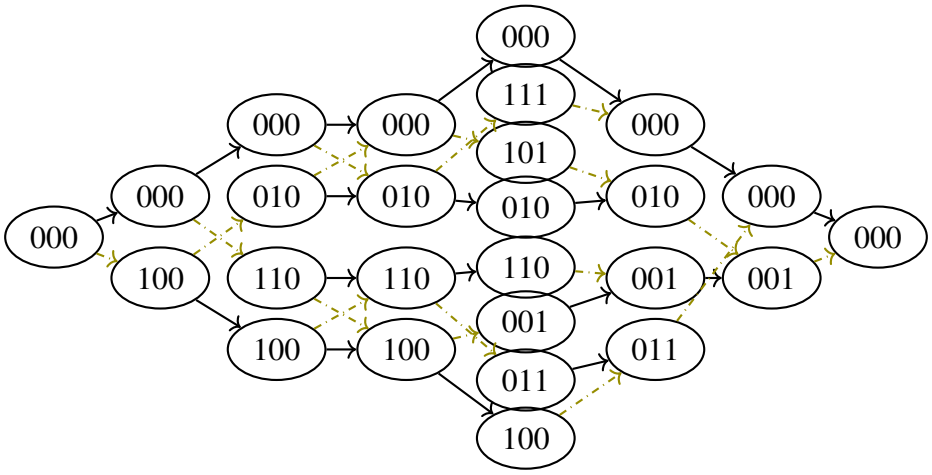
Consider the ET $\mathcal{T}^1$ in the previous example. The corresponding generator has left-right indices $[1,3]$, which can be seen directly in the trellis. The number of paths in $\mathcal{T}^1$ is $|\langle S_1^\perp \rangle| = |\{[0,0,0,0,0,0,0], [1,1,1,0,0,0,0]\}| = 2$. The edge $(\overline{0},0,\overline{0})$ interacts with any other edge (v_i, P_i, v_{i+1}) as $(\overline{0}+v_i, P_i+0, \overline{0}+v_{i+1}) = v_i, P_i, v_{i+1})$ and hence acts as the identity in its section. Hence, sections E_4 to E_7 of $\mathcal{T}^1$ do not interact with any other ETs in the product in these sections. Likewise, sections E_1 to E_3 are completely constructed in $\mathcal{T}^1 \square_i \mathcal{T}^2 \square_i \mathcal{T}^3$ because $\mathcal{T}^4$ has only the identity path until E_5. These are



(a) $\mathcal{T}^1 \square_i \mathcal{T}^2$



(b) $\mathcal{T}^1 \square_i \mathcal{T}^2 \square_i \mathcal{T}^3$



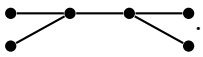
(c) $\mathcal{T}^1 \square_i \mathcal{T}^2 \square_i \mathcal{T}^3 \square_i \mathcal{T}^4$

Figure 3.3: Constructing the CSS trellis for the Steane code using the TP.

graphical equivalents to the statements about active and inactive generators made when discussing the TOF.

Since every ET contains the identity path, the TP always contains an exact copy of all of its factors, inducing a well-defined projector $\Pi_{\mathcal{T}^j}(\mathcal{T}^1 \square_i \ldots \square_i \mathcal{T}^\ell) = \mathcal{T}^j$. This further illustrates Example 6 and the discussion of CSS trellises of the previous chapter. The syndromes of Figures 2.5a, 2.5b, and 2.6 (a) should be longer but are written with respect to $\mathcal{S}_X$ or $\mathcal{S}_Z$ only for simplicity because they are identically zero for the other set. The $\mathcal{S}_X$ labels thus serve as the identity path for the $\mathcal{S}_Z$ labels and vice versa. By the previous discussion, there is an exact copy of both CSS trellises inside the full trellis plus terms with mixed labels coming from the TP, as expected.

Now consider a generator $A_j \in \mathbb{F}_{p^2}^n$ with $p > 2$. The ET contains paths for $\langle A_j \rangle = \{A_j, 2A_j, \ldots, (p-1)A_j\rangle$. Since the partial syndrome is a homomorphism, if v_i is the partial syndrome of A_j at depth i, then ℓv_i is the partial syndrome for ℓA_j for $2 \leq \ell \leq (p-1)$. These are distinct for $v_i \neq \overline{0}$. Suppose $v_i = \overline{0}$ for some depth not equal to the left or right index of A_j. Since the degree of a vertex is either 1 or p, $\deg_{\text{in}} \overline{0}_i = p$ for some i and $\deg_{\text{out}} \overline{0}_m = p$ for some $L(A_j) < i \leq m < R(A_j)$; for example,



But then there are at least p^2 paths in the trellis which is in one-to-one correspondence with the p elements of $\langle A_j \rangle$, a contradiction. We have shown the following result.

Lemma 3.4.5

The ET for A_j contains $p-1$ parallel paths which expand from and merge with the identity path only at the left and right indices of A_j, respectively, but otherwise do not intersect.

Lemma 3.4.6

Let A_1 and A_2 be generators in TOF with ETs $\mathcal{T}^1$ and $\mathcal{T}^2$, respectively. The paths of the

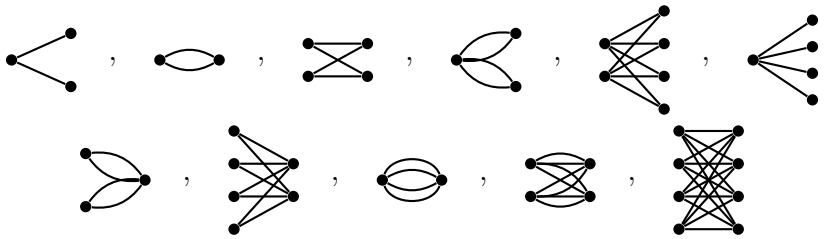
TP $\mathcal{T}^1 \square_i \mathcal{T}^2$ only intersect the identity path at the left and right indices of each generator. Equivalently, no two ETs for a set of generators in TOF contain the same nonzero vertex in the same vertex depth.

Proof. Suppose $\overline{0} \neq v \in V_i^1$ and $v \in V_i^2$. Then $\{v, 2v, \ldots, (p-1)v\}$ are also all in $V_i^{1,2}$. In particular, the additive inverse of v exists, producing a vertex $\overline{0}$ with degree p in $\mathcal{T}^1 \square_i \mathcal{T}^2$. There are $p^2 - 1$ paths going through this vertex. Every vertex in $V_i(\mathcal{T}^1 \square_i \mathcal{T}^2)$ will also have the same degree and number of paths, contradicting the fact that there are only p^2 total paths in $\mathcal{T}^1 \square_i \mathcal{T}^2$. □

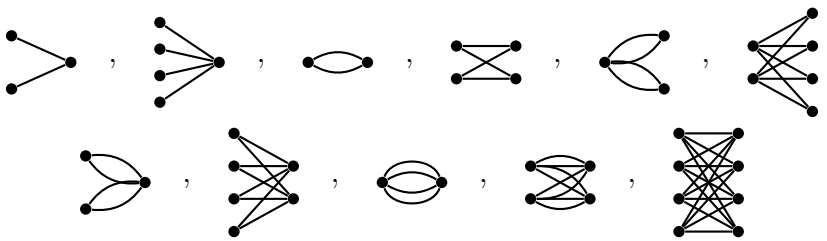
Corollary 3.4.7

Let $\mathcal{T}^1 = (V^1, E^1)$ and $\mathcal{T}^2 = (V^2, E^2)$ be two ETs. Then $|V_i(\mathcal{T}^1 \square_i \mathcal{T}^2)| = |V_i^1(\mathcal{T}^1)||V_i^2(\mathcal{T}^2)|$ and $|E_i(\mathcal{T}^1 \square_i \mathcal{T}^2)| = |E_i^1(\mathcal{T}^1)||E_i^2(\mathcal{T}^2)|$.

Re-examining the edge configurations in light of Lemma 3.4.5 and the TP we see that because any configuration is shared by $\overline{0}$ the following are only possible when there exists a generator with a left index at this section



and

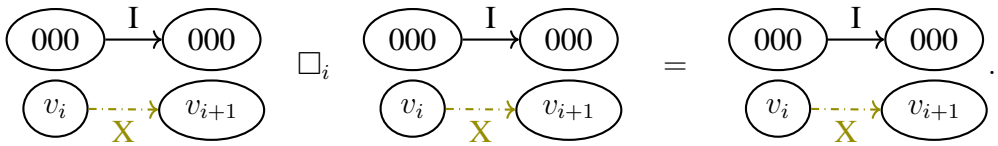


are only possible when there exists a generator with right index at that section. For all other

sections without either a left or right index,

is the only possible configuration, since the TP of parallel paths are parallel paths. These are the sections which can be merged during sectionalization for free. It follows that low-weight stabilizers permuted to be active over a large span, or high-weight stabilizers, such that the occurrences of left and right indices are infrequent may benefit the most from sectionalization.

Corollary 3.4.7 is one of the fundamental properties of the TP quoted throughout the literature. This was used for CSS trellises in Lemma 2.4.3, however, this does not hold in general. To see this, consider the TP of this edge configuration with itself over $\mathbb{F}_2$,



Here, all edges of the TP are

$$\{(\overline{0}+\overline{0}, 0+0, \overline{0}+\overline{0}), (\overline{0}+v_i, 0+1, \overline{0}+v_{i+1}), (v_i+\overline{0}, 1+0, v_{i+1}+\overline{0}),$$
$$(v_i+v_i, 1+1, v_{i+1}+v_{i+1}\}$$
$$= \{(\overline{0}, 0, \overline{0}), (v_i, 1, v_{i+1})\}.$$

This simplifies because vertex and edge label addition are commutative. If we consider edges labeled by elements $\eta^c X(a)Z(b) \in \mathcal{P}_n$ instead of $x \in \mathbb{F}_{p^2}^n$, we would need to be more careful due to the commutation relations between X and Z. There are many edge configurations which act as an idempotent for the TP. The reason for this is that the vertices form a discrete, additive subgroup and adding a subgroup still generates the same group.

Lemma 3.4.8

Let $\mathcal{T}^1$ and $\mathcal{T}^2$ be two trellises and suppose, without loss of generality, $V_i^1 \cap V_i^2$. Then $|V_i^2(\mathcal{T}^2)| = |V_i(\mathcal{T}^1 \square_i \mathcal{T}^2)| \neq |V_i^1(\mathcal{T}^1)||V_i^2(\mathcal{T}^2)|$.

Fortunately, Corollary 3.4.7 shows that $V_i^1 \cap V_i^2 = \{\}$ for ETs.

Remark: These results are in contrast with the non-syndrome-based trellises of the classical literature in which the paths can intersect in different ways. In fact, it was in this context that the edge configurations in Table 2.1 were first noted. It is therefore interesting that the same edge configurations not only exist for the syndrome trellis but are also provably the complete set.

Some of the previous results may seem like nothing more than a restatement of Table 2.1, which was deduced from the TOF, but interpreting this from a purely graph theoretic standpoint allows us to ask different questions. In particular, given only the trellis, can we decompose it into a set of ETs from which a set of generators in TOF follows? The answer to this is yes, which might not be surprising given that the Cartesian product is decomposable in each section. However, the standard decomposition may choose any of the non-identity elements of $\langle A_j \rangle$ and the choices may not line up from section to section to make connected paths from V_0 to V_n. Fortunately, we will see that the strict structure of the trellis as a directed graph allows us to extract this information almost immediately with little work and without relying on standard graph product decomposition algorithms.

Remark: Koetter and Vardy considered the related question of whether or not a (non-syndrome based) trellis represents a linear subspace and showed that a trellis represents a linear subspace if and only if can be constructed via the TP [199]. We are not concerned with this here; we assume that all the trellises dealt with in this section are constructed properly using the techniques of Chapter 2. This implies that they are depth n, weakly-connected, equipathic, graded, acyclic, directed graphs with a unique source and sink such

that every vertex has a connected path to both V_0 and V_n. We also assume the number of paths in the trellis is the correct cardinality of the space, and, for simplicity of discussion, there are no parallel edges. We have already shown that such trellises can be constructed via the TP of ETs in Theorem 2.4.4. The proof of the decomposition (factorization) in [199] relies on a simultaneous study of the trellis and a generator matrix for the space. Here, we suppose that the trellis is constructed then all information about the generators are subsequently lost and we wish to recover them.

We begin by noting that if the trellis was constructed using the TP in the first place, information about the ETs could be stored to make this decomposition immediate if desired. So we assume this is not the case. Consider the trellis $\mathcal{T}^1 \Box_i \mathcal{T}^2 \Box_i \mathcal{T}^3 \Box_i \mathcal{T}^4$ in Figure 3.3c. Using the previous discussion we see that the non-parallel structure of $E_1 - E_4$ imply the existence of a left index in the TOF at these locations. Likewise, there are right indices in the TOF at 3, 5, 6, and 7. If we can pair up the left and right indices, we could immediately identity the zero and nonzero entries of the TOF. By Definition 2.3.18 of the TOF, the left-right indices pair to minimize the total span length.

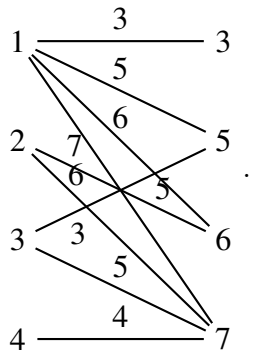
Definition 3.4.9

Two vertices are called admissible if there exists a path between them that does not include part of the identity path.

Let $\overline{0}_L$ and $\overline{0}_R$ be the zero vertices at the left and right indices L and R, respectively. To determine the pairings, construct a bipartite graph with a vertex for every left and right index. Connect each admissible pair of vertices with a weighted edge whose value is the minimum (Manhattan) distance between them. This can be found efficiently via depth-first search on the outgoing edges of $\overline{0}_L$. If it exists, the directed and graded properties of the trellis give that this distance is exactly $R - L + 1$. The structure of the TOF follows from

a MWPM on this graph.

Example 12. The matching graph for Figure 3.3c is



.

Two left-right pairs $(1, 3)$ and $(4, 7)$ are immediate and can be removed from the matching graph. MWPM for the other two give the TOF

$$\begin{pmatrix} * & * & * & 0 & 0 & 0 & 0 \\ 0 & * & * & * & * & * & 0 \\ 0 & 0 & * & * & * & 0 & 0 \\ 0 & 0 & 0 & * & * & * & * \end{pmatrix}.$$

Consider the ET for the first row in the TOF of the previous example: $A_1 = [*, *, *, 0, 0, 0, 0, 0]$. There are exactly $(p-1) = 1$ non-identity paths, and one of them begins with a 1, α, or α^2. The goal is to find $\mathcal{T}^1$ in Figure 3.3c, which is equivalent to understanding how to implement the projector $\Pi_{\mathcal{T}^1}$. Recall from Chapter 2 that the edge configurations of E_i is a collection of shifts of the edge configuration for $(\overline{0}, 0, \overline{0}) \in E_i$. The previous discussions make the origin and nature of the shifts more apparent: the edge configuration is copied onto every parallel path in the ET via the TP. The edge configuration copied onto the identity path is the location of $\mathcal{T}^1$. Thus, to extract $\mathcal{T}^1$ from $\mathcal{T}^1 \square_i \mathcal{T}^2 \square_i \mathcal{T}^3 \square_i \mathcal{T}^4$ it suffices to study the outgoing edges of $\overline{0}$.

Since A_1 is the only row of the TOF with this left index, these are the only outgoing

paths from $\overline{0} \in V_0$ and are immediately identifiable. If there were two rows of the TOF with this left index, there would be $p^2 - 1$ outgoing non-identity paths from $\overline{0}_1$, and we could single out those with edge labels 1 or α for A_1 and recognize the rest as the parallel paths of $\langle A_1 \rangle$. Having identified the first edge, $[1, *, *, 0, 0, 0, 0]$, we note that there can be no other path from $\overline{0} \in V_0$ to $\overline{0} \in V_3$ starting with this edge than the one representing A_1 because if there was we could use it to reduce the span of A_1, contradicting the definition of the TOF. Therefore, $A_2 = [1, 1, 1, 0, 0, 0, 0]$.

Repeating this for $A_2 = [0, *, *, *, *, *, 0]$, we find two possible generators $[0, 1, 0, 1, 0, 1, 0]$ and $[0, 1, 1, 0, 1, 1, 0]$. This occurs because the span of the third row, A_3, is contained in the span of A_2. Picking a path corresponds to choosing A_2 and A_3 or $A_2 A_3$ and A_3 as generators, either of which are fine. The rest of the rows have unique paths, producing a TOF which agrees with $\mathcal{S}^{\perp}$ above.

It is easy to see that the above logic produces a decomposition for any trellis.

Theorem 3.4.10

Any syndrome trellis is decomposable into a TP of its ETs.

3.5 Partial Ordering And Intersection Of Trellises

Thinking of a trellis as a set of elements of a set, the partial ordering of sets by inclusion induces a partial ordering of trellises.

Definition 3.5.1

Let $\mathcal{T}^1$ and $\mathcal{T}^2$ be two syndrome trellises constructed as in Chapter 2. Then

(i) *$\mathcal{T}^1 \prec \mathcal{T}^2$ if the set of elements represented by the edges of $\mathcal{T}^1$ is strictly contained in the set of elements represented by the edges of $\mathcal{T}^2$,*

(ii) *$\mathcal{T}^1 = \mathcal{T}^2$ if the set of elements represented by the edges of $\mathcal{T}^1$ is equal to the set of elements represented by the edges of $\mathcal{T}^2$,*

(iii) *$\mathcal{T}^1 \preceq \mathcal{T}^2$ if the set of elements represented by the edges of $\mathcal{T}^1$ is strictly contained in*

or is equal to the set of elements represented by the edges of $\mathcal{T}^2$,

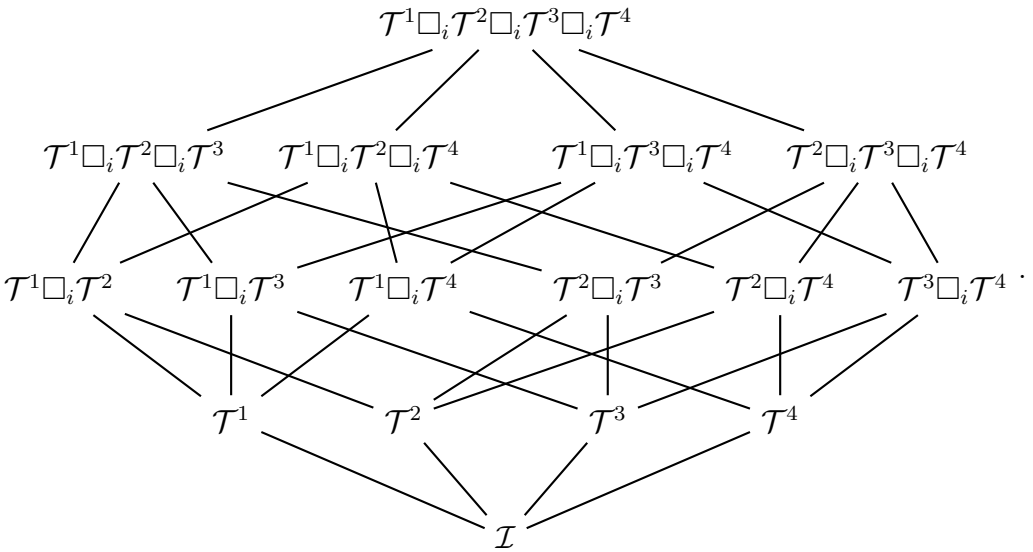
(iv) $\mathcal{T}^1$ and $\mathcal{T}^2$ are said to be incomparable if none of the above relationships hold.

Similar relationships hold for the symbols $\succ$ and $\succeq$.

This is clearly reflexive ($\mathcal{T} \preceq \mathcal{T}$), antisymmetric (if $\mathcal{T}^1 \prec \mathcal{T}^2$ and $\mathcal{T}^2 \prec \mathcal{T}^1$ then $\mathcal{T}^1 = \mathcal{T}^2$), and transitive (if $\mathcal{T}^1 \prec \mathcal{T}^2$ and $\mathcal{T}^2 \prec \mathcal{T}^3$, then $\mathcal{T}^1 \prec \mathcal{T}^3$).

In general, most trellises are incomparable. Those that are, are subcodes or have common subcodes. All codes (trellises) contain the identity, so $\mathcal{I}$ is a least element for the set of all trellises ($\nexists \mathcal{T}$ such that $\mathcal{T} \prec \mathcal{I}$). All codes considered here are contained in $\mathbb{F}_q^n$ (classical) or $\mathbb{F}_{q^2}^n$ (quantum), but we cannot build a trellis for the ambient space, as the dual space is zero. In this sense, there are maximal elements but no greatest element. More usefully, consider a trellis whose edges represent the elements $\mathcal{A} = \langle A_1, \ldots, A_\ell \rangle$. The power set of generators $\{A_j\}$ gives a Hasse diagram of trellises via the TP equivalent to the Hasse diagram of subcodes. While a partial ordering on trellises produces no new information about the objects or their relationships, it takes us one step closer to viewing the trellis as a fundamental representation of the code without requiring the simultaneous storage of other representations to understand it.

Example 13. A Hasse diagram for the CSS trellises for the Steane code of Example 11 is



A common operation in classical coding theory is the intersection of two codes, $\mathcal{C}_1 \cap \mathcal{C}_2$, defined as the set of common elements. The codes do not need to be related in any sense, but some formulas exist for families with special structure. For example, the intersection of two cyclic codes $\mathcal{C}_1 = \langle g_1(x) \rangle$ and $\mathcal{C}_2 = \langle g_2(x) \rangle$ is $\mathcal{C}_1 \cap \mathcal{C}_2 = \langle \mathrm{lcm}\{g_1(x), g_2(x)\} \rangle$. One may always compute the intersection of two finitely-generated modules using standard elementary linear algebra, so it is lamentable that, at the time of writing, many existing coding theory libraries do this by enumerating all possible elements of both codes, finding the intersection of the two lists, then building a finitely-generating set from it. Intersection is not a common operation in QEC, although it is important in subsystem codes and code switching (e.g. [200]). As a representation of all of the elements of a code, it is not surprising that we can define the intersection of two trellises.

Definition 3.5.2

Let $\mathcal{T}^1$ and $\mathcal{T}^2$ be two syndrome trellises of depth n constructed as in Chapter 2. Then $\mathcal{T}^1 \cap \mathcal{T}^2$ is the syndrome trellis whose paths are in one-to-one correspondence with the paths in both $\mathcal{T}^1$ and $\mathcal{T}^2$.

We note right away that this is distinct from the common operation of the intersection of two graphs which individually checks every source-edge label-terminus pairing. The intersection of trellises can only contain connected paths from V_0 to V_n. For any element represented by a path in the intersection, the full cyclic group generated by the element must be contained in both trellises and therefore the intersection as well. We can use such observations to see that the intersection of trellises is determined by their ETs and hence by common ancestors in their Hasse diagrams.

The decomposition of trellises into their ETs is the most efficient way to compute the intersection. In fact, the identification of the left-right indices is enough to determine when the intersection is trivial ($\mathcal{I}$). We may compare this to the more standard graph-theoretical approaches of depth-first search (DFS) and breadth-first search (BFS). For DFS, start with

$\overline{0} \in V_0$ in both trellises and choose the first outgoing edge with the same label that exists in both trellises. Move to V_1 and repeat. If V_n is reached or there are no common edges, backtrack to the previous depth and search for the next common edge label. If one trellis is completely contained in the other, say if we were comparing $\mathcal{S}^\perp$ with $\mathcal{Z}(\mathcal{S}^\perp) = \mathcal{S}$, then this will enumerate every element of $\mathcal{S}$, which we want to avoid.

Notice that the algorithm for DFS did not make use of the vertex information. This is because one can construct the same trellis with different vertex labels by, for example, replacing or permuting generators in the TOF. The edges are the invariant information. In the BFS approach, we analyze every edge in a section before moving on to the next section. In order to move from section to section, we require some way to identify how a path in one trellis corresponds to a path in the other, and to do this we introduce new vertex labels. Begin again with $\overline{0} \in V_0$ in both trellises and choose a common edge. Assign the terminus of these edges the same label in both trellises. Repeat this for everything in E_0. Moving to E_1, pick a newly labeled vertex in one trellis and its pair in the other trellis, then find all common edges and label termini as before. Doing this for all labeled vertices in E_1, we may repeat the process until E_n. In the worst case, this will enumerate all $|E|$ edges in one or both of the trellises; however, it makes better use of the edge sharing than DFS.

3.6 The Dual Trellis And Degenerate Decoding

As mentioned in the previous chapter, many results in classical coding theory rely on the dual trellis, which is defined to be the trellis of the dual code. The relationship between $\mathcal{S}$ and $\mathcal{S}^\perp$ is more complicated, but that does not prevent us from formally defining a dual trellis. The roles of the generator and parity-check matrices switch in $\mathcal{C}$ and $\mathcal{C}^\perp$, motivating us to switch the roles of $\mathcal{S}$ and $\mathcal{S}^\perp$ in the dual trellis for a quantum stabilizer code. For increased clarity of discussion, the original trellis of the previous chapter will be referred to as the primal trellis and equations with respect to the dual with be adorned with a superscript $\mathfrak{d}$.

Let $\mathcal{S}$ be a stabilizer group corresponding to an $[[n,k,d]]_p$ stabilizer code. First we note that this object is well-defined. The edges are given by elements of $\mathcal{S}$ and the vertices by partial syndromes with respect to $\mathcal{S}^\perp$. Since $\mathcal{S}$ commutes with $\mathcal{S}^\perp$, V_n consists of only a single vertex with label zero, as before. Following the same proofs as before, we find

$$|V_i^{\mathfrak{d}}| = p^{\dim \mathcal{S} - \dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{f}_i}} \tag{3.1}$$

$$|E_i^{\mathfrak{d}}| = p^{\dim \mathcal{S} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}} - \dim \mathcal{S}_{\mathfrak{f}_i}} \tag{3.2}$$

$$\deg_{\text{in}}^{\mathfrak{d}}(v) = p^{\dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}}} \tag{3.3}$$

$$\deg_{\text{out}}^{\mathfrak{d}}(v) = p^{\dim \mathcal{S}_{\mathfrak{f}_i} - \dim \mathcal{S}_{\mathfrak{f}_{i+1}}}. \tag{3.4}$$

In the previous chapter we remarked that we could have also proved the State Space Theorem using $\mathcal{S}$ instead of $\mathcal{S}^\perp$, resulting in

$$|V_i| = p^{\dim \mathcal{S}^\perp - \dim \mathcal{S}_{\mathfrak{p}_i}^\perp - \dim \mathcal{S}_{\mathfrak{f}_i}^\perp} = p^{\dim \mathcal{S} - \dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{f}_i}}. \tag{3.5}$$

Comparing this to (3.1) shows that $|V|$ is not reduced in the dual trellis. Since $|\mathcal{S}| < |\mathcal{S}^\perp|$, we would hope the total number of edges would decrease in the dual, but there is no obvious guarantee that the dual trellis shares edges as efficiently as the primal trellis. In fact, we find that $|E_i^{\mathfrak{d}}|$ may theoretically be even larger than the primal trellis.

Proposition 3.6.1

Let V, E be with respect to the primal trellis and $V^{\mathfrak{d}}, E^{\mathfrak{d}}$ be with respect to the dual. Then

$$p^{-2}|E_i| \leq |E_i^{\mathfrak{d}}| \leq p^2|E_i|. \tag{3.6}$$

Proof. Equating the exponents of (3.5) we have

$$\dim \mathcal{S}^\perp - \dim \mathcal{S}_{\mathfrak{p}_i}^\perp - \dim \mathcal{S}_{\mathfrak{f}_i}^\perp = \dim \mathcal{S} - \dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{f}_i}$$

or

$$\dim \mathcal{S} = \dim \mathcal{S}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} - \dim \mathcal{S}_{\mathfrak{f}_i}^{\perp} + \dim \mathcal{S}_{\mathfrak{f}_i} + \dim \mathcal{S}_{\mathfrak{p}_i}.$$

Substituting into Equation (3.2),

$$\begin{aligned}
|E_i^{\mathfrak{d}}| &= p^{\dim \mathcal{S} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}} - \dim \mathcal{S}_{\mathfrak{f}_i}} \\
&= p^{\dim \mathcal{S}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} - \dim \mathcal{S}_{\mathfrak{f}_i}^{\perp} + \dim \mathcal{S}_{\mathfrak{f}_i} + \dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}} - \dim \mathcal{S}_{\mathfrak{f}_i}} \\
&= p^{\left(\dim \mathcal{S}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp} - \dim \mathcal{S}_{\mathfrak{f}_i}^{\perp}\right) + \left(\dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} + \dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}}\right)} \\
&= \frac{|E_i|}{p^{\left(\dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp}\right) - \left(\dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}}\right)}}.
\end{aligned}$$

The sequence $\mathcal{A}_{\mathfrak{p}_i}$ is monotonically increasing with maximum difference two given by the number of new right pivots in the TOF. The denominator is maximum when $\dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp} = 2$ and $\dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}} = 0$, giving the lower bound. The upper bound corresponds to the opposite case $\dim \mathcal{S}_{\mathfrak{p}_i}^{\perp} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}}^{\perp} = 0$ and $\dim \mathcal{S}_{\mathfrak{p}_i} - \dim \mathcal{S}_{\mathfrak{p}_{i-1}} = 2$. □

Remark: For all of the cases we have tried so far, $|E_i^{\mathfrak{d}}| \leq |E_i|$, implying there may be something deeper going on similar to the discussion between Theorem 2.3.12 and Corollary 2.3.14. See Table 3.2.

Table 3.2: The percentage $|E^{\mathfrak{d}}|/|E|$ of the dual trellis to primal trellis for the codes of Chapter 2 rounded to the nearest whole number. The lower distance codes have so few edges that removing any is significant, while the opposite effect happens for the higher distances.

	3	5	7	9	11	13	15	17	19	21
4.8.8	79	89	91	94	95	98	99	95	95	95
4.8.8 X/Z	89	95	97	98	98	99	99	98	99	99
6.6.6	79	83	86	94	97	96	95	98	99	98
6.6.6 X/Z	89	93	95	98	99	98	98	99	99	99
RSurf	79	94	97	99	99	99	100	100	100	100
RSurf X	93	95	95	96	96	96	97	97	97	97

The first step in trellis decoding is to shift with respect to the length-$(n-k)$ measured syndrome. This works because the vertices of the primal trellis are partial syndromes with respect to the stabilizer generators. The vertices of the dual trellis are partial syndromes with respect to the $n+k$ generators of $\mathcal{S}^\perp$, which are not measured. We may compensate for this by constructing a pure error for the dual trellis given the pure error for the measured syndrome. Let s be a measured syndrome, $H^{\mathcal{S}}$ and $H^{\mathcal{S}^\perp}$ be the stabilizer matrices for generators of $\mathcal{S}$ and $\mathcal{S}^\perp$, respectively, and $H^{\mathcal{S}+}$ and $H^{\mathcal{S}^\perp+}$ be appropriate pseudoinverses. Then $T_s^{\mathcal{S}} = H^{\mathcal{S}+}s$ is a pure error for the primal trellis and $H^{\mathcal{S}^\perp}T_s^{\mathcal{S}}$ is the corresponding syndrome for the dual trellis. A pure error for the dual trellis is therefore $H^{\mathcal{S}^\perp+}H^{\mathcal{S}^\perp}T_s^{\mathcal{S}} = T_s^{\mathcal{S}}$. So the same pure error works for shifting both trellises, eliminating the need to compute and store $H^{\mathcal{S}^\perp}$ and $H^{\mathcal{S}^\perp+}$.

The obvious drawback of decoding using the dual trellis is that it contains none of the information about the logical operators that made the primal trellis so successful. The dual trellis only contains information concerning the identity coset of $\mathcal{S}$ in $\mathcal{S}^\perp$. Let $\{\ell_i\}_{i=1}^{2k}$ be coset representatives of the (physical) logical operators. In order to decode as well as before, we need to construct trellises for the rest of the cosets $\ell_i S$. We know from the last chapter's discussion of affine spaces that this is exactly the dual trellis shifted by ℓ_i. We could therefore repeatedly decode on all $2k$ shifted trellises then take the most likely result as the final answer. This is of course embarrassingly parallel and could represent a significant speedup if $|E^{\mathfrak{d}}| \ll |E|$. In all of the examples computed for this work, this was never the case, so the primal trellis was preferred. We will refer to the dual trellis plus its logical shifts collectively as coset trellises and this method of decoding as coset decoding.

The optimal use case for the coset trellises is solving the degenerate decoding problem Equation (1.49). As previously mentioned, the difficulty here lies in enumerating $\mathcal{S}^\perp$, organizing the elements into cosets, then computing the probabilities of each coset. The coset trellises solve the first two of these problems. The most probable coset is defined to be the greatest sum of all of the probabilities of every element in the coset, taken over

all cosets. Going back to Section 1.5, errors act i.i.d. on each qubit so that $\Pr(E) = \prod_{i=1}^{n} \Pr(E_i)$ for $E \in \mathcal{P}_n$. The straightforward approach of extracting every path from the trellis is infeasible and defeats the purpose of constructing the trellis in the first place. Fortunately, there are two ways to extract the summary information of each coset without explicit knowledge of its elements.

For both techniques, we change the weights from the additive log-likelihoods to standard multiplicative probabilities. The first way to do this is using a standard product-sum algorithm at each vertex: at each $v \in V_i$ for $1 \leq i \leq n$, pick an edge and multiply the weight of the source vertex by the weight of the edge, repeat for all edges, add the results, and store the final value at the vertex. The value at V_n is the desired probability.

Algorithm 3: Probability of a coset using forward dynamical programming.

Input: A vertex set, V, along with an edge set, E, corresponding to a valid trellis.
Output: Probability

```
Struct Vertex contains
    float value
end
Struct Edge contains
    Vertex source
    float weight
end
V_0 ∋ 0̄.value ← 0.0
for i ← 1 to n do
    foreach v ∈ V_i do
        foreach e ∈ E_i do
            v.value ← v.value + e.source.value * e.weight
        end
    end
end
return V_n ∋ 0̄.value
```

Proposition 3.6.2

Algorithm 3 computes the coset probability.

The proof is similar to that of Proposition 2.3.21.

The second technique to compute the coset probabilities is to use Lemma 3.3.1 (ii). Using a weighted adjacency matrix, A, to represent the (shifted) trellis, the coset probability is exactly the sole nonzero value $A_{0,n}$ of A^n. The previous discussion of the computation of A^n applies, and we note that, depending on the implementation, it is possible to compute this faster than product-sum in some circumstances.

In both techniques, the probability of each coset is computed, hopefully in parallel. The most probable coset is chosen, and the applied correction is the corresponding logical operator combined with the pure error T_s. The identity coset is always preferred and other ties are broken at random. One can use the Viterbi algorithm to find a most-likely element inside the most-likely coset, if desired, but this is unnecessary.

It is known that not all stabilizer codes are going to benefit from the extra effort required to solve the degenerate decoding problem. There are few examples of this in the literature since prior to this work there was no general degenerate decoder, and preliminary results using the technique developed here confirm this. Informally, the “large gains" expected with a degenerate decoder seem to be achieved by the primal trellis and account for the gap between many state-of-the-art decoders and traditional trellis decoding.

Example 14. Consider the Steane code under the action of the depolarizing channel. In this model, the weight of a coset $\mathcal{A}$ is

$$\sum_{A \in \mathcal{A}} \prod_{i=1}^{n} \Pr\left(A_i\right) = \sum_{A \in \mathcal{A}} (1-p)^{7-\mathrm{wt}(A)} p^{\mathrm{wt}(A)},$$

where $\mathrm{wt}(A)$ is the Hamming weight of A. For small enough p, low-weight terms completely dominate the sum. For example, when $p = 10^{-3}$, a single weight-one term has a probability

$$(1-10^{-3})^6 10^{-3} = 9.9 \times 10^{-4},$$

whereas the weight of a coset whose every element is weight-two is

$$64(1-10^{-3})^5(10^{-3})^2 = 6.4 \times 10^{-5}.$$

This particular code is both the easiest and the least interesting to analyze due to its triangular symmetry. To demonstrate the effect of symmetry here, note that for every X there is an element with a Z and hence a Y in its place. The complete weight enumerator (CWE) of $\mathcal{S}$, computed using the technique in the proceeding sections, is

$$1 + 7x^4 + 7y^4 + 7z^4 + 42x^2y^2z^2.$$

The weight enumerator of $T_s\mathcal{S}$ for $T_s = X_1$ is

$$x + 4x^3 + 3x^5 + 3xz^4 + 3xy^4 + 4y^3z + 4yz^3 + 12xy^2z^2 + 12x^2yz^3 + 12x^2y^3z + 6x^3y^2z^2$$

By symmetry, Z_1 and Y_1 errors give instead

$$z + 4z^3 + 3z^5 + 3x^4z + 3y^4z + 4x^3y + 4xy^3 + 12xy^3z^2 + 12x^2y^2z + 12x^3yz^2 + 6x^2y^2z^3$$

and

$$y + 4y^3 + 3y^5 + 3x^4y + 3yz^4 + 4x^3z + 4xz^3 + 12xy^2z^3 + 12x^2yz^2 + 12x^3y^2z + 6x^2y^3z^2,$$

respectively. Hence it suffices to only consider X errors. There are also only three unique single qubit errors up to symmetry: X_1, X_2, and X_5. The CWE of each of these errors are the same and the logical X cosets are

$$3x^2+4x^4+x^6+3x^2z^4+12xy^3z+12xyz^3+12x^2y^2z^2+3x^2y^4+4x^3yz^3+4x^3y^3z+6y^2z^2.$$

Thus there is only one weight-one solution, which is an element of the most likely coset, so both and primal and dual trellises will always return the same correction. One can check that this holds for all single qubit errors.

Example 15. The previous example assumed the pseudoinverse of the syndrome returned the minimum-weight pure error. If this were always the case, we would not need decoders. Suppose instead that an X_1 error was treated as X_4X_7. This causes the identity and logical cosets to switch with respect to the previous example. The logical coset is now most likely and contains the weight-one solution X_1. The correction is therefore $(X_4X_7)X_L = X_1$, as desired.

The tensor network decoder of [111] also solves the maximum-likely coset problem for surface codes (only). This decoder is complex and a proper description of it is beyond the intended scope of this work. It is able to compute the exact coset probabilities for independent X or Z error channels but must resort to approximations for the depolarizing channel. The algorithm may also take significant time and resources and requires a practical limit on the distance between correlated qubits. Reference [111] reports the ratio of the degenerate decoder to MWPM is no more than around 1.8 for the independent X channel, which is a negligible effect for such small numbers. This is in agreement with preliminary findings using the dual trellis for various codes. They detected a discernible difference between the two decoding methods for the depolarizing channel and $d = 25$. Further numerical simulations are needed to thoroughly investigate this parameter regime using the dual trellis. Trellis decoding is simpler than tensor network decoding and should theoretically achieve the same results.

3.7 Minimum Distance

Since the paths of a trellis can encode all of the elements of $\mathcal{C}$, S, $S^\perp$, or a coset $\mathcal{S}^\perp/\mathcal{S}$, it is not surprising that we can use it to compute the minimum distance of a code. This is surely

not a novel idea for classical codes; however, it is suspiciously absent from the literature. The QEC literature is almost completely silent on the issue of computing the minimum distance. A 2022 paper by Pryadko et al [201] claimed that "The lack of available software has caused researchers in the field of QECC to either skip the minimum distance calculations altogether (...), or develop their own suboptimal algorithms. In particular, Bravyi and Hastings (...) used an exhaustive search over all non-trivial codewords for calculating the minimum distances.". Here, we briefly describe the current minimum-weight algorithms, why a trellis may not currently be considered one of them, and when a trellis approach can be useful. This is valid for both classical and quantum codes; we begin with the classical case.

3.7.1 Classical

Recall that for linear codes the minimum distance is equal to the minimum weight. It is therefore always valid to compute every element of the code one-by-one by constructing all possible combinations of the generators of the generator matrix, computing their weights, and then returning the minimum. This takes exponential time. Vardy showed that the computation of the minimum distance is NP-Hard and the corresponding decision problem is NP-Complete [202], so we can't expect a polynomial-time algorithm but we can do better than brute force. There are two main minimum-weight algorithms in the literature: the Brouwer-Zimmermann (BZ) algorithm and Leon's algorithm [203]. The latter is probabilistic and returns the minimum-weight with high confidence for a binary linear code. The BZ algorithm is implemented in MAGMA over any finite field [204], in the GAP coding theory package GUAVA over $\mathbb{F}_2$ and $\mathbb{F}_3$ [205], and the library in Sagemath is a Python interface to GAP [206]. The MAGMA implementation is the fastest and is considered the standard.

The history of the Brouwer algorithm is reported in [204, 207]. Brouwer developed the algorithm in the 1980's and it remains unpublished. He used this to publish tables

of best upper and lower bounds of the minimum distance of linear codes (which are now maintained by Grassl [171]). Brouwer visited the MAGMA group in 1992 and explained the algorithm which was then added to the proprietary MAGMA C kernel. Perhaps due to this, a large number of coding theory libraries currently available use the brute-force solution, adding Brouwer-Zimmermann type algorithms only relatively recently if at all. In 1999, the MAGMA coding theory project formed to find explicit codes realizing Brouwer's bounds. Most knowledge of the BZ algorithm comes from reports from this project by Grassl [204] and White [207] in 2006, the latter of which cites the original C code, although Zimmermann's refinement was published in German in 1996 [208] and 1998 [209].

Let $\mathcal{C}$ be an $[n, k]_q$ code with standard form generator matrix $G = (I \mid A)$, where I is the $k \times k$ identity matrix. For $v \in \mathbb{F}_q^k$, $c = vG = (v \mid vA)$, so $\mathrm{wt}(c) \geq \mathrm{wt}(v)$, which we can use to reduce the number of elements we have to enumerate. One is always a lower bound on the minimum distance and an upper bound is given by the Singleton bound, $d \leq n - k + 1$. Enumerate all elements of $v \in \mathbb{F}_q^k$ with Hamming weight equal to the lower bound r and compute the set $W = \{vG\}$. For any element $c \in \mathcal{C} \backslash W$, $\mathrm{wt}(c) \geq r + 1$ since $\mathrm{wt}(c) \geq \mathrm{wt}(v)$ which must be at least $r + 1$ or else it would be in W. If an element of W has weight equal to the lower bound, we are done. Otherwise, set the upper bound equal to the minimum weight of the set, increase the lower bound by one, and repeat.

If $d \geq k + 1$, this will still enumerate everything. To get around this we must increase the lower bound faster. Brouwer used the fact that any set of k linearly independent vectors generate $\mathcal{C}$ and simultaneously utilized multiple generator matrices to compute multiple sets of encoded vectors. The key is to come up with m generator matrices with disjoint information sets. Since $G = (I \mid A)$, the first k columns form an information set and the others must be found in the columns of A. The lower bound in this case can start at m. Given m disjoint information sets, Gauss-Jordan[2] can be used to put the identity matrix in these columns producing matrices $\{G_2, \ldots, G_m\}$. We then proceed as before but now the

[2]We can ignore any column permutations because it doesn't affect the weight.

upper bound becomes the minimum of the previous upper bound and the weights of the union of the sets $\{vG_i\}$ for $1 \leq i \leq m$. After enumerating all elements of weight w, the lower bound becomes $m(w+1)$. Unfortunately m can be as low as one for many codes of interest, reducing to the previous algorithm. Also, if $k \nmid n$, then there are $n \bmod k$ columns not being used to produce information for the lower bound.

Zimmermann adapted Brouwer's algorithm to use overlapping information sets: Start with the standard form generator matrix $G_1 = (I_k \mid A_1)$. If the matrix A_1 is nonzero it has nonzero rank $0 < k_2 \leq k_1 \coloneqq k$. Define G_2 by reducing A_1, allowing the identity of G_1 to change,

$$G_2 = \left(\begin{array}{c|c|c} A_2' & I_{k_2} & A_2 \\ \hline & 0 & 0 \end{array} \right).$$

Repeat this for each A_i until either A_i is empty or zero,

$$G_m = \left(\begin{array}{c|c|c} A_m' & I_{k_m} & A_m \\ \hline & 0 & 0 \end{array} \right).$$

The last matrix A_m has $n - k_1 - \ldots - k_m$ columns, so G_m has this many zero columns such that $\mathrm{wt}(vG_m) \leq k_1 + \ldots + k_m$. We then proceed as before but one can show that the lower bound at each step is now

$$\sum_{\substack{i=1 \\ k-k_i \leq j}}^{m} (j+1) - (k - k_i).$$

More matrices mean more enumeration but also a faster growing lower bound. MAGMA chooses which algorithm to use by using an estimate on the number of elements it will need to generate. See the references for details.

There are several improvements one can make but the underlying idea of computing sets $\{vG_i\}$ and updating bounds remains constant. Lisoněk and Trummer used Edmond's matroid partitioning algorithm to construct information sets with optimal properties and

showed this has a dramatic effect on the runtime of BZ [210]. In an extreme case, their new information sets were able to generate all minimum weight codewords of a $[1344, 128]$ code used in an attack on the block cipher PICARO [211] in only 94 seconds, while MAGMA estimated it would complete in roughly 10^6 years. Bouyuklieva and Bouyukliev extended the definition of information set to further reduce the number enumerated elements [212]. References [204, 213, 214] consider implementation improvements from a computer science perspective leading to dramatically shorter runtimes than the current MAGMA standard.

For many codes of practical use, the number of elements which must be enumerated before the upper and lower bounds meet is low enough for this to be usable. For other codes one can consider other techniques. Sometimes the number of elements is greater than the number of elements of the code itself, so computing the weight distribution is better. Recall that $\mathcal{C}^\perp$ has parameters $[n, n-k]$ and that $|\mathcal{C}| = q^k$ and $|\mathcal{C}^\perp| = q^{n-k}$. If the number of enumerated elements is expected to be greater than $|\mathcal{C}^\perp|$, it is more efficient to compute the weight distribution of $\mathcal{C}^\perp$, apply the MacWilliams identities (1.2) to get the weight distribution of $\mathcal{C}$, and then return the lowest exponent of x. Known symmetries of the code can also be extremely useful. Permutations can also dramatically change the runtimes of algorithms. MAGMA heuristically permutes the generator matrix if it detects its information set algorithm does not produce a short enough expected runtime [204].

A trellis representation of $\mathcal{C}$ can return the minimum distance by weighting all edges with nonzero labels with a one and the rest with zeros. The Viterbi algorithm then computes not only the minimum weight but also an element of minimum weight, if desired, at negligible extra cost. This latter point is key as it may take the BZ algorithm an extra enumeration cycle to produce an element after its upper and lower bounds meet. As with decoding, the runtime cost for this is $\mathcal{O}(|E|)$ and this holds true for both low-distance and high-distance codes. When n is large, an enumeration-based method could return quicker than the trellis can be constructed, although the adjacency matrix approach presented in this chapter should improve this. On the other hand, enumeration-based methods may be

inappropriate when d is large.

A fair comparison between the two algorithms is difficult. Explicit runtimes are implementation, machine, and resource dependent, and trellises have not shared the same level of interest and optimization as MAGMA. The first performance analysis of the BZ algorithm and its variants was conducted in 2006 by White [207]. The enumeration step overwhelmingly dominates the runtime of these algorithms, so all other operations are assumed to cost unit time and are ignored. This includes the numerous comparisons between the bounds and the weight of the newly generated vector, but comparisons make up the total of the Viterbi algorithm. Yet even for the largest trellis of the previous chapter, unoptimized code benchmarked Viterbi in the microseconds. So while this is this $\mathcal{O}(|E|)$, the constants matter.

The construction phase of the trellis overwhelmingly dominates the runtime of this approach. Unfortunately, this is also extremely implementation dependent. The BZ algorithm can forget an encoded vector immediately after generating it, keeping storage costs minimal. The trellis is of course saved in memory and accessing it depends highly on the size, location, and structure of the underlying data structure in memory. A single V_i may not fit in the cache and building E_i may be dominated by cache misses. As we've seen above, simplifying the data structure can improve this dramatically. A full computer science analysis is beyond the intended scope of this work, so we will not discuss this further.

So why isn't this considered amongst the possible minimum-distance algorithms? One guess is the era during which these algorithms were developed. Trellises can take a significant amount of storage memory which was just simply not available in the late 1980's, early 1990's. Also, without the new construction algorithms discussed in this work, trellises were largely generated by enumerating elements of $\mathcal{C}$ until complete. In many if not most cases, the BZ algorithm will terminate before the trellis is even constructed. However, this may no longer hold true. We hope to have made a convincing argument that there are cases in which trellises should be considered as a realistic minimum-distance algorithm.

3.7.2 Quantum

The situation is much different in the quantum case. Recall that the minimum distance is given by the minimum weight of a non-trivial logical operator (1.29). This generally has nothing to do with the minimum distance of the corresponding stabilizer code considered as a classical, additive code. Note that $\mathcal{C}_{\mathcal{P}_n}(\mathcal{S})\backslash\mathcal{S}$ is a set-difference of size $p^{n+k}-p^{n-k}$ and not a quotient module of size p^{2k}. Constructing a basis for the centralizer is an easy row reduction but enumerating its elements are not as easy as before. The brute-force method always works, but the concept of an information set is now more complicated since one cannot row reduce down to the identity, as we saw with the TOF. White and Grassl tackled this in [207, 215] where they map the additive code onto a linear code in a way that the minimum of distance of the additive code may be implied from that of the linear code. This mapping increases the parameters from n to $3n$ and d to $2d$, dramatically increasing the overall runtime of the BZ algorithm. Further complicating the quantum case, once a minimal weight vector is detected, one must check to see if it is an element of S.

To see how classical intuition can be harmful here, recall that the rotated surface code of Figure 1.2 has many weight-two elements but is distance five. In general, low-weight elements are necessary for quantum codes to perform well against certain types of errors [216]. The Steane code also has minimum distance three despite having all elements of weight four. These apparent inconsistencies go back to the fact that stabilizer codes are specified by their parity-check matrices but the distances are determined by the dual.

The previous section on coset trellises show how to organize the elements of $\mathcal{C}_{\mathcal{P}_n}(\mathcal{S})$ by cosets of $\mathcal{S}$. To determine the minimum weight, we may simply exclude the $\mathcal{S}$-only trellis corresponding to the identity coset. The steps are therefore 1) construct the coset trellis with $\mathcal{S}$ on the edges, $\mathcal{S}^{\perp}$ on the vertices, and weights 1 for any non-identity label and 0 otherwise, 2) shift with respect to the first logical operator (ℓ_1), 3) apply Viterbi to find a minimum weight element, 4) shift with respect to the next logical operator ($\ell_2\ell_1^{-1}$), 5) repeat steps 3) and 4) until all logical operators have been used, 6) return the minimum

of all the outputs of the Vitberi algorithm.

The algorithm presented in [201] was developed independently of and simultaneously to the one in this work and is probabilistic for a specific family of stabilizer codes called low-density parity-check (LDPC) codes. Since the trellis can be constructed for any stabilizer code, the method described here can work for any stabilizer code.

3.8 Weight Distributions & Enumerators

3.8.1 Classical

Let $\mathcal{C}$ be an $[n, k, d]_q$ code and recall from the first chapter that the weight enumerator of $\mathcal{C}$ is the bivariate polynomial

$$W(\mathcal{C}; x, y) = \sum_{i=0}^{n} A_i x^i y^{n-i},$$

where A_i is the number of elements of $\mathcal{C}$ with weight i. The weight distribution of $\mathcal{C}$ is the ordered sequence $\{A_i\}_{i=0}^{n}$. This is sometimes called the homogeneous weight enumerator since we can define this without the y if desired. If we wanted to be more descriptive, we could extend the polynomial to have $|\mathbb{F}_q|$ variables and count the number of occurrences of each element of $\mathbb{F}_q$ in each vector $v \in \mathcal{C}$. For example, define a CWE over $\mathbb{F}_4$ by

$$w^{\mathrm{wt}_0(v)} x^{\mathrm{wt}_1(v)} y^{\mathrm{wt}_{\alpha^2}(v)} z^{\mathrm{wt}_\alpha(v)},$$

where the subscript on the weight counts the number of occurrences of that symbol; then $v = [1, 0, \alpha, \alpha, 1, \alpha^2]$ corresponds to wx^2yz^2. The homogeneous and complete weight enumerators coincide for $\mathbb{F}_2$, and the homogeneous is derivable from the complete over higher fields by summing the exponents not corresponding to the identity: $wx^2yz^2 \mapsto x^5y$. Complete are more difficult to analytically predict than the homogeneous case, but this usually isn't a problem since $\mathbb{F}_2$ codes are dominate. Enumeration-based algorithms which

compute the weight enumerator should be able to compute both with no extra effort. Thus, in this work we adopt the approach that the CWE should always be computed and anything else should be derived from this.

In addition to BZ-type enumeration algorithms, there are a few probabilistic, genetic, and Walsh-Hadamard transform algorithms to obtain the weight distribution of a linear code in the academic literature. However, as before, the brute-force algorithm is overwhelmingly used in current software. Trellises may also be used for this purpose, and are the only way to compute the weight enumerator for convolutional codes, which we do not discuss in this work. We are unfamiliar with the literature in this direction. Note that to compute a weight enumerator from a BZ-type algorithm, one must either enumerate the entire code or compute enough values of the weight distribution that the rest may be inferred from symmetries. As before, one may compute the weight enumerator of the dual if easier and then convert back to the desired code using MacWilliams-like identities.

Let $\mathcal{C}$ be an $[n, k, d]_q$ code and assume its trellis representation has been constructed. Every vertex will be assigned a multivariate polynomial in $\mathbb{Z}[x_0, \ldots, x_{m-1}]$ with the exponents of each variable assigned to an element of the field. Initiate the polynomial at V_0 to 1. Now pick a $v \in V_1$ and consider an edge in $e = (s(e), L, v) \in E_1$ with $L \in \mathbb{F}_q$. Multiply the polynomial at the vertex $s(e)$ by the x_i assigned to L. Repeat this for every edge, sum the results, and assign the (reduced) polynomial to v. Do this for every vertex in the trellis. The polynomial at V_n is the CWE of C. See Algorithm 4. This an example of a sum-product algorithm, which are widely used in decoding methods for linear codes.

Theorem 3.8.1

Algorithm 4 computes the CWE of an $[n, k, d]_q$ *code* $\mathcal{C}$.

Proof. Enumerate the elements of $\mathbb{F}_q$ by integers in $\{1, \ldots, m\}$. We've already shown there's a one-to-one correspondence between paths in the trellis and elements of $\mathcal{C}$. It remains to show that the operation at each vertex is correct. Since $n \geq 1$, initiate the polynomial at V_0 to one. Since $\deg_{\text{in}}(v) = 1$ for all $v \in V_1$, the polynomial at any vertex

Algorithm 4: CWE

Input: A vertex set, V, along with an edge set, E, corresponding to a valid trellis.
A fixed ordering of symbols of $\mathbb{F}_q$ corresponding to variables $\{x_0, \ldots, x_m\}$.

Output: The CWE.

```
Struct Vertex contains
    Edge edge
    Polynomial polynomial
end
Struct Edge contains
    float weight
    char label
end
V_0.polynomial ← 1
for i ← 1 to n do
    for (j, v) ← Enumerate(V_i) do
        poly ← 0
        for e ← E_i[j] do
            prev ← V_{i−1}[e.outvertex].polynomial
            for k ← e.label do
                for (ℓ, x) ← Enumerate(F_q) do
                    if k is equal x then
                        for term in prev do
                            term ← term * x_ℓ
                        end
                        break
                    end
                end
            end
        end
        poly ← poly + prev
    end
    v.polynomial ← Simplify(poly)
end
return V_n.polynomial
```

is x_j corresponding to the jth element in the enumeration of $\mathbb{F}_q$, which is correct. Now suppose $v \in V_{i+1}$ for some $1 \leq i < n$ and pick an edge $e = (s(e), P, t(e) = v) \in E_i$. Denote the path from V_0 to $s(e) \in V_i$ by $P_1 \ldots P_i$. Including the edge extends the path to $P_1 \ldots P_i P$. The weight enumerator at $s(e)$ must increase the exponent corresponding

to the label $P \in \mathbb{F}_q$, which is equivalent to multiplying by the appropriate x_j; store this at v. Picking another edge with terminus v, if any, represents a new path which needs to be distinguished from the previous. Multiplying with respect to the edge label and then adding the result to the weight enumerator from the previous edge accomplishes this. The two terms will only be combined if they are the same, which is correct since the weight enumerator only distinguishes paths up to weights and not labels. Since there are only n depths, the total degree of each term must be less than or equal to n with equality only if the homogeneous version is being computed. □

Remark: After the completion of this work, the author stumbled upon a 1994 paper cited almost exclusively in a small Japanese coding theory community which proposes an algorithm for generating the weight distribution of binary linear code using a sectionalized trellis [217]. Their algorithm is equivalent to the one presented here when specialized to that case but their approach is sufficiently different. Their approach emphasizes the set of paths between pairs of arbitrary vertices as the fundamental object, whereas we have chosen V and E. The path view is dominant in the classical trellis literature and is natural since distinct paths leaving $v \in V_i$ cannot reintersect until at least V_{i+d}, where d is the minimum distance. This does not hold in the quantum case, where the nature of d is more complicated. As such, the quantum case is currently missing analogous results to much, if not most, of the classical theory. Their approach is two-sided, working from both V_0 and V_n and appealing to Figure 2.4.

We have so far skipped a rigorous analysis of our algorithms here because our emphasis is on the comparison to enumeration-based algorithms, which is not an apples-to-apples comparison for storage heavy algorithms with repeated memory accesses. The analysis in [217] counts the number of additions and multiplications but the result is highly dependent on the chosen sectionalization. In enumeration-based algorithms, basic arithmetic operations are assumed to have negligible cost. For the construction phase, the analysis simply sets the cost to $|E|$ and cites the original classical construction technique of Wolf [173].

This method generates all possible edges at every vertex until depth N and then goes backwards and removes all paths and vertices which don't end at the zero syndrome vertex. This is clearly exponential and is cited heavily throughout the literature, although enumeration techniques are used to generate all possible parallel edges during sectionalization. It's likely enumeration is also used to construct trellises by generating vectors until the bounds on V_i, E_i are satisfied, but we have no explicit reference for this. The analysis in [217] claims $|E|$ is less than $2n \min\{2^k, 2^{n-k}\}$, where the latter is the Wolf bound mentioned in Corollary 2.4.1. The claim is that this proves efficiency, but this is unclear as it is clearly not scalable.

Interestingly, [217] points out that the weight distribution of shortened codes could be computed by simply skipping appropriate edges. Their comment should extend to punctured codes as well. Trellis symmetries are also considered to speed up the algorithm, which we do not consider here.

3.8.2 Quantum

Weight enumerators play a significantly less important role in QEC than in classical coding theory. For starters, the role of the dual is more complicated in QEC, but also the emphasis in quantum is the circuits and gates and less on the combinatorial structure of the distribution of non-identity operators in the stabilizer group. Quantum weight enumerators, now called the Shor-Laflamme enumerators, were first defined by Shor and Laflamme in 1997 in terms of arbitrary operators and their Hibert-Schmidt inner product with the basis $\mathcal{P}_n$ [218]. They also defined a formal dual by applying the MacWilliams identity and interpreted the result in relation to (what we have described in this work as) the success and failure probabilities of decoding. Rains reworked the Shor-Laflamme enumerators in 1998 into a new definition with made their properties more explicit [219]. Both of these definitions essentially define the weight enumerator of a QECC to be that of its stabilizer code considered as a classical additive code over $\mathbb{F}_{q^2}$. This is not enough for many problems in quantum theory.

Here, we would like to encode the full group structure $\eta^c X(a)Z(b)$. Knill and Laflamme (2001) [220] and Rall (2017) [221] defined signed homogeneous weight enumerators for qubit codes by allowing the coefficients A_i to be negative. Such weight enumerators often include a lot of cancelations and there's nothing preventing this definition from returning the zero polynomial. Here, we extend the definition of weight enumerators to construct an invariant of the code which keeps better track of the combinatorial structure of the group.

Definition 3.8.2 (Signed Complete Weight Enumerator)

Let $\mathcal{A} \subseteq \mathcal{P}_n$ with elements of the form $\pm X(a)Z(b)$, where $a, b \in \mathbb{F}_2^n$. Denote the elements of $\mathbb{F}_4$ by $\{0, 1, \alpha, \alpha^2\}$. Consider the map $\phi : \mathcal{P}_n \to \{\pm 1\} \times \mathbb{F}_4^n$, $\pm X(a)Z(b) \mapsto \pm P$, where $P \in \mathbb{F}_4^n$ and $P_i = a_i + b_i\alpha$ for $1 \leq i \leq n$, as before. For $P \in \mathcal{P}_n$ denote the number of occurrences of $j \in \mathbb{F}_4$ by $\mathrm{wt}_j(P)$. Define $A(t_0, t_1, t_\alpha, t_{\alpha^2})$ to be the number of elements $+P \in \phi(\mathcal{A})$ with $t_0 = \mathrm{wt}_0(P)$, $t_1 = \mathrm{wt}_1(P)$, $t_\alpha = \mathrm{wt}_\alpha(P)$, $t_{\alpha^2} = \mathrm{wt}_{\alpha^2}(P)$. Likewise, denote $B(t_0, t_1, t_\alpha, t_{\alpha^2})$ to be the number of elements $-P \in \phi(\mathcal{A})$ with corresponding weights. The signed complete weight enumerator of $\mathcal{A}$ is the Laurent polynomial

$$W_{\mathcal{A}}(x, y, z) = \sum_{(t_0,t_1,t_\alpha,t_{\alpha^2})\in\mathcal{I}} A(t_0, t_1, t_\alpha, t_{\alpha^2})x^{t_1}y^{t_{\alpha^2}}z^{t_\alpha} + B(t_0, t_1, t_\alpha, t_{\alpha^2})x^{-t_1}y^{-t_{\alpha^2}}z^{-t_\alpha}, \tag{3.7}$$

where $\mathcal{I} = \{(t_0, t_1, t_\alpha t_{\alpha^2}) \mid 0 \leq t_i \leq n, \sum t_i = n\}$. The corresponding signed homogeneous, complete weight enumerator of $\mathcal{A}$ is the Laurent polynomial

$$\begin{aligned} W_{\mathcal{A}}(x, y, z) = \sum_{(t_0,t_1,t_\alpha,t_{\alpha^2})\in\mathcal{I}} & A(t_0, t_1, t_\alpha, t_{\alpha^2})w^{t_0}x^{t_1}y^{t_{\alpha^2}}z^{t_\alpha} \\ & + B(t_0, t_1, t_\alpha, t_{\alpha^2})w^{-t_0}x^{-t_1}y^{-t_{\alpha^2}}z^{-t_\alpha}. \end{aligned} \tag{3.8}$$

There may be situations in which we want to physically distinguish the cases where an XZ error occurs, which is why this is not counted as one X and one Z in the weight enumerator. The case in which we don't want to do this is derivable from our definition

by adding or subtracting the exponent of y to that of x and z appropriately and then removing the symbol. It is also sometimes useful to know only the X terms or only the Z terms, which may be derived by distributing the exponent of y then dropping the other unwanted symbols. If the code is CSS, this is equivalent to the standard classical weight enumerator of the linear code whose generator matrix is given by the X or Z stabilizer matrices. The previous definitions of signed weight enumerators in the QEC literature may be recovered by mapping $B(t_0, t_1, t_\alpha, t_{\alpha^2})$ to $-B(-t_0, -t_1, -t_\alpha, -t_{\alpha^2})$ and simplifying. To the author's knowledge this is the only weight enumerator in the literature with negative exponents. This makes sense since there are no phases in classical codes, and unlike the previous quantum literature we have, for better or for worse, abandoned the concept of a MacWilliams identity. We are comfortable with using the term "weight enumerator" since our definition satisfies the intention of this name.

Despite the lack of attention to weight enumerators in QEC, it is a bit ironic that in the short paper introducing the quantum syndrome trellis, Ollivier and Tillich ended with a paragraph description of how to use a trellis to compute a weight enumerator of a stabilizer code by treating it as a classical code [167]. While we very naturally stumbled upon the same basic algorithm, our result extends theirs by redefining the trellis of Chapter 2 to include signs. We spare the reader of going through all of the technicalities of that chapter and instead describe the signed trellis by pointing out what changes.

It is not so simple to keep track of the overall phase of a path due to edge sharing in the trellis. Instead, we have to think deeper about the physics of the operator. Let $\mathcal{S}$ be the stabilizer group of a quantum stabilizer code $\mathcal{Q}$. Phases cannot be assigned randomly to elements $\eta^c X(a)Z(b) \in \mathcal{S}$; the η^c must be chosen in a consistent manner to maintain group closure and a non-trivial eigenspace for $\mathcal{Q}$. Physically, the η^c are determined by the overall phases of the individual qubits. Each qubit can be "in phase" or "out of phase" with respect to X-axis rotations, Z-axis rotations, or both.

Definition 3.8.3 (Character Vector)

A character vector for an n-qudit system is a length-$2n$ complex vector, χ, consisting of all the X phases then the Z phases on each qudit in a fixed order,

$$\chi = \left[\eta_X^{c_1}, \ldots, \eta_X^{c_n}, \eta_Z^{c'_1}, \ldots, \eta_Z^{c'_n}\right].$$

Assuming $\mathcal{Q}$ is non-trivial, for a given stabilizer $\eta^c X(a)Z(b)$,

$$\eta^c = \prod_{i \in \mathrm{supp}(a)} \chi_i \prod_{j \in \mathrm{supp}(b)} \chi_j, \tag{3.9}$$

where $\mathrm{supp}(v) = \{i \mid v_i \neq 0\}$ is vector support.

To unambiguously add signs to the trellis, modify every edge label with appropriate signs using χ. The correctness of this method follows from the one-to-one correspondence between paths in the trellis and elements of the subgroup and (3.9). Given the signed trellis, we want to compute the signed complete weight enumerator of the subgroups $\mathcal{S}$, $\mathcal{C}_{\mathcal{P}_n}(\mathcal{S})$, and $\mathcal{C}_{\mathcal{P}_n}(\mathcal{S})\backslash\mathcal{S}$ for a stabilizer group over $\mathbb{F}_2$. The weight enumerator for $\mathcal{C}_{\mathcal{P}_n}(\mathcal{S})\backslash\mathcal{S}$ may be computed by summing the weight enumerators of each coset trellis since cosets are disjoint. The idea is to modify Algorithm 4 such that any term with positive exponents flips to one with negative exponents and vice versa when encountering an edge with a negative phase. Positive exponents stay positive and negative exponents stay negative when encountering an edge with a positive phase. Algorithm 5 assumes the fixed ordering of $\mathbb{F}_4 = \{0, 1, \alpha, \alpha^2\}$ corresponding to variables $\{w, x, z, y\}$, respectively.

Theorem 3.8.4

Algorithm 5 correctly computes the signed complete weight enumerator of a subset $\mathcal{A} \subset \mathcal{P}_n$ defined by the trellis.

The proof follows from the proof of Theorem 3.8.1 while appealing to Equation (3.9).

As an application of the new weight enumerator, let ρ be the density matrix of a quantum system encoded in some quantum stabilizer code $\mathcal{Q}$ and U be an operator in $\mathcal{M}(2^n, \mathbb{C})$.

Algorithm 5: Signed Complete Weight Enumerator

Input: A vertex set, V, along with an edge set, E, corresponding to a valid signed trellis.
Output: The signed complete weight enumerator.

```
Struct Vertex contains
    Edge edge
    Polynomial polynomial
end
Struct Edge contains
    char label
    Int sign
end
V_0.polynomial ← 1
for i ← 1 to n do
    for (j, v) ← Enumerate(V_i) do
        poly ← 0
        for e ← E_i[j] do
            prev ← V_{i−1}[e.outvertex].polynomial
            for k ← e.label do
                if k is equal 0 then
                    if e.sign is equal +1 then
                        if any exponents are negative then
                            for term ← prev do
                                term ← term * w^{−1}
                            end
                        else
                            for term ← prev do
                                term ← term * w
                            end
                        end
                    else
                        for term ← prev do
                            // Flip signs of all exponents
                            if any exponents are negative then
                                term ← term * w^{−1}
                            else
                                term ← term * w
                            end
                        end
                    end
                end
                // Repeat for rest of symbols
            end
        end
        poly ← poly + prev
    end
    v.polynomial ← Simplify(poly)
end
return V_n.polynomial
```

Recall from Equation (1.23) that U may be written as a linear combination of $\{X(a)Z(b)\}$ with coefficients $U_{a,b} \in \mathbb{C}$,

$$U = \sum_{a \in \mathbb{F}_q^n} \sum_{b \in \mathbb{F}_q^n} U_{a,b} X(a) Z(b).$$

We wish to consider the action of the special case

$$U_Z = \sum_{b \in \mathbb{F}_q^n} U_b X(0) Z(b)$$

on ρ via $U_Z \Pi_{\mathcal{S}} \rho \Pi_{\mathcal{S}}^* U_Z^*$, where $\Pi_{\mathcal{S}}$ is a projector onto the subspace fixed by the stabilizer group $\mathcal{S}$ of $\mathcal{Q}$. Suppose this is a CSS code with X and Z stabilizers given by generator matrices of the classical codes $\mathcal{C}_2$ and $\mathcal{C}_1^\perp$, respectively. Since $\Pi_{\mathcal{S}} = \Pi_{\mathcal{S}_X} \Pi_{\mathcal{S}_Z}$ and $U_Z \Pi_{S_X}$ simply produces a phase factor canceled out by swapping $\Pi_{S_X}^*$ and U_Z^*, it remains to study $U_Z \Pi_{\mathcal{S}_Z}$. Hu, Liang, and Calderbank show that

$$U_Z \Pi_{\mathcal{S}_Z} = \frac{1}{|\mathcal{C}_1^\perp|} \sum_{\mu \in \mathbb{F}_q^n / \mathcal{C}_2^\perp} \sum_{\gamma \in \mathcal{C}_2^\perp / \mathcal{C}_1^\perp} A_{\mu,\gamma} \sum_{u \in \mathcal{C}_1^\perp + v} \omega^{c_u} Z(u),$$

where μ are pure errors for Z, γ are logical operators for Z, and

$$A_{\mu,\gamma} \coloneqq \sum_{v \in \mathcal{C}_1^\perp + \mu + \gamma} \eta^{-c_v} U_v$$

are the so-called generator coefficients of U_Z [89]. In the case $U_Z = \left(e^{-i\theta Z/2}\right)^{\otimes n}$,

$$A_{\mu,\gamma}(\theta) = \sum_{v \in \mathcal{C}_1^\perp + \mu + \gamma} \eta^{-c_v} \left(\cos\frac{\theta}{2}\right)^{n - \mathrm{wt}(v)} \left(-i \sin\frac{\theta}{2}\right)^{\mathrm{wt}(v)}.$$

To compute $A_{\mu,\gamma}(\theta)$, one needs to know not only the Hamming weight but also the phase η^{-c_v} of each operator in $\mathcal{C}_1^\perp + \mu + \gamma$. This is easily accomplished with the trellis since the signed CWE for the trellis for $\mathcal{C}_1^\perp$ shifted by a fixed $\mu + \gamma$ contains all of this in-

formation. Note that if U_Z preserves the code space, it can contain no pure errors ($\mu = 0$), making the operator more physically interesting and the sum more manageable. The examples in [89] either assume $\eta^{c_v} = 1$ for all v, in which case the weight enumerator of the classical code suffices, or the codes are small enough to brute-force enumerate every element. Generator coefficients have been shown to be useful in the study and design of QECCs [76, 216, 222], and the application here may be extended past our simple example.

Remark: Subtracting the weight enumerator of $\mathcal{S}$ from the weight enumerator of $\mathcal{S}^\perp$ provides an alternative method of computing the minimum distance of a stabilizer code. This avoids the repeated shifting and processing of each coset in Section 3.7 for a faster runtime at the expense of not returning a vector of minimum weight. Additionally, since the weight enumerator never needs to traverse the trellis in both directions, one may compute the trellis section-by-section, never storing more than one section in memory at a time. This further reduces the storage cost and can be useful if the trellis is not going to be used twice.

3.9 Words Of Bounded Weight

Similar to the previous two applications, it is often useful to compute the set of elements of a code, classical or quantum, with (Hamming) weight between some upper and lower bound. Often, we want to know if there exist elements with a specific weight and if so what they are in which case the upper and lower bound coincide. Conceptually, one could do this using a trellis by simply applying DFS from V_n to V_0. Each vertex must now store a list of size $\deg_{\text{in}}$ of all path weights ending at that vertex, which is already computed by the Viterbi algorithm. DFS is well-understood so we do not discuss this further. Note however, this is unlikely to be parallelizable due to potential simultaneous memory access with intersecting paths, and having a large gap between the upper and lower bounds will require enumerating a significant amount of elements which could behave like a brute-force enumeration algorithm. In these cases, a BZ-type algorithm will be more efficient

due to being able to use more than one generator matrix at a time. The catch is determining whether a generator matrix has computed an element that another generator matrix has already done. See [223] for an example solution. Note that the straightforward use of the BZ algorithm for this is only effective for low weights, an adaption can be made for high weights, and brute-force is faster for a range of weights in the middle [207]. This difficulty is not present in the trellis-based algorithm.

Since computing the weight enumerators is typically so hard, it is often beneficial to compute a partial weight distribution of the code using a words-of-bounded-weight algorithm. This can give information about the number of correctable errors beyond the $\lfloor (d-1)/2 \rfloor$ guarantee. The trellis algorithm proposed here has the same runtime for the partial and full weight enumerator, rendering this issue moot.

While we have not discussed it in this work, one of the most important pieces of information about a code is its automorphism group (see Section 1.1). The most well-known and widely implemented algorithm for this (GUAVA and MAGMA) is due to Leon [203]. The input to this algorithm is the set of elements of minimum weight. The original implementations for this used the brute-force algorithm. More recent updates ran BZ once to find the minimum weight, again to compute all the elements of minimum weight, then ran Leon's algorithm. At least MAGMA has combined the first two steps such that the elements of lowest known weight are stored until a new minimum weight is discovered. This is also inefficient. The author has never implemented Leon's algorithm, but [207] reports that this often dominates the runtime of Leon's algorithm. This again is not a problem for the trellis-based approach. The forward pass (V_0 to V_n) of the Viterbi algorithm computes the minimum weight and then DFS on the backwards pass completes the set of minimum weight elements.

3.10 Subfield Subcodes

Let $\mathcal{C}$ be an error correcting code over $\mathbb{F}_{p^m}$, and recall from Section 1.1 that the subfield subcode of $\mathcal{C}$ over $\mathbb{F} \leq \mathbb{F}_{p^m}$, $\mathcal{C}|_{\mathbb{F}}$, is the collection of codewords of $\mathcal{C}$ whose components lie entirely in $\mathbb{F}$. This is surprisingly easy to compute: chose a basis for $\mathbb{F}_{p^m}/\mathbb{F}$, replace each element of the parity-check matrix of $\mathcal{C}$, H, with a column vector representing its basis expansion, delete any linearly dependent rows, then the result is the parity-check matrix of $\mathcal{C}|_{\mathbb{F}}$. Using the terminology of Section 1.1, this is the transpose of the expansion of H^T, since the expansion described there was row-wise (for generator matrices). A dual basis may be need to be computed to determine the expansion of each element; see Section 1.1 for details.

The subfield subcode is similarly easy to compute directly from the trellis. If the trellis is the only available information about a code, one does not need to extract a generating set, compute a parity-check matrix, and then construct a trellis for the subcode. Instead, one need only select the length-n paths of the trellis whose edge labels are elements of $\mathbb{F}$. To do this, set $\overline{0} \in V_0$ as “marked" and examine all outgoing edges. Set any edges and corresponding terminal vertices as marked if the edge label is in the subfield. Now move to V_1 and repeat the same procedure for all of the marked vertices. Continue this until V_{n-1}. Any path from V_0 to V_n consisting of entirely marked edges and vertices is in the subfield subcode, and it is clear from construction that this is the entire subcode. This is never empty as the identity is always in both $\mathcal{C}$ and $\mathcal{C}|_{\mathbb{F}}$.

This is a BFS approach. It may be desirable to remove the unmarked and “dead end" paths and vertices which don’t make it to V_n before requiring an element of the extension field. Instead of “deleting" or “unmarking", one may extract the subcode with BFS, working backwards from V_n along the marked paths. We refer to the trellis for the subfield subcode as the subfield trellis.

Generators for $\mathcal{C}_{\mathbb{F}}$ may not have any discernible relationship to those of the supercode;

however, the above procedure makes clear that the set of left and right indices for the TOF of $\mathcal{C}|_{\mathbb{F}}$ must be contained in the set of left and right indices for the TOF of $\mathcal{C}$. This is stronger than the statement that the generators of $\mathcal{C}$ may be chosen as the union of generators for $\mathcal{C}|_{\mathbb{F}}$ and generators for $\mathcal{C}\backslash\mathcal{C}|_{\mathbb{F}}$, and may point to the use of the TOF as a more important canonical form than currently used in the literature.

The appeal of trellis algorithms for computing these quantities is in their absolute simplicity. A single sweep through the trellis can generate the weight enumerators at the same price as computing the minimum distance, so one might as well get the extra information. If the trellis is going to be constructed for another purpose, it does not make sense to use another algorithm. The NP-Hard runtime of these algorithms gets transferred to the one-time construction and the exponential storage requirement. Given these two aspects can be accommodated on a particular computer system, the trellis allows for quick computation of numerous desirable objects and properties without repeated enumerations of elements.

CHAPTER 4

FUTURE DIRECTIONS

Despite the progress of the last two chapters, there are more open theoretical questions about trellises than we have answered, leaving plenty of room for future exploration. We hope the internal structure of the trellis will one day be useful in understanding the internal structure of QECCs and their design. It is likely this will stem from a better understanding of the relationship between the trellis, its dual (cosets), and their subcodes. Some of this in the classical literature relies on the relationship between the trellis and the minimum distance, which we have just established. The vertex and edge scaling of families of QECCs is particularly important to understand for the practicality of trellis decoding of stabilizer codes moving forward.

We have attempted to reduce both the storage and time requirements of the trellis. In some situations, large trellises for classical codes are not stored but are computed as needed. More advanced decoding techniques are also used to circumvent the high computational complexity of the Viterbi algorithm; see [224] for a review. Both are fruitful avenues to explore for QEC.

With respect to the material contained in this work, more examples and simulations are needed. While we have not emphasized this here, simulations are the most important aspect discussed here for the physics community. Decoders are typically benchmarked against the depolarizing channel, as we have done; however, this is the worst case scenario for the trellis as all non-identity edges have equal weight and ties are broken randomly. Both depolarizing and biased error channels (e.g. $p_X < p_Z$) are experimentally relevant; yet experimental characterizations of current quantum technology demonstrate that noise does not affect each qubit equally. We should therefore be using an independently and non-identically distributed (i.ni.d.) error channel. This may be taken into account in MWPM by

updating the edge weights in the matching graph, as before. Instead of using the Manhattan metric on the lattice, the edge weights of the matching graph should be the maximum-likely path between the two syndrome bits appropriately incorporating p_X, p_Y, and p_Z in the edges of the lattice. This optimal path must be computed and requires standard graph algorithms, such as A^*, to be run for every edge in the matching graph, which is a potential bottleneck. Incorporating an i.ni.d. error channel for any stabilizer code with the trellis is easy: simply use a different error model for each E_i for no added complexity.

A recent 2022 paper [225] simulated both the standard and modified MWPM algorithms for a standard topological code for $d = 3$, 5, and 7 using the i.ni.d. error channel

$$\mathcal{E}(\rho) = (1 - p_X - p_Y - p_Z)\rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z,$$

where

$$p_X = p_Y = \frac{1}{4}\left(1 - e^{-t/T_1}\right) \qquad , \qquad p_Z = \frac{1}{4}\left(1 + e^{-t/T_2} - 2e^{-t/T_2}\right),$$

and T_1 and T_2 are physical parameters called the relaxation time and dephasing time, respectively. Their $d \times d$ lattices are firmly in the finite-size effect regime and one does not expect the results to hold as d goes to infinity (for example, note the behavior of the lower distance lines in Figure 2.9). Nevertheless, they find the standard MWPM algorithm performs poorly under the new error model and report significant gains for modified MWPM using Dijkstra's algorithm.

Modifying the edge weights of the trellis may also be the key to extending the decoder to new paradigms. For example, erasures can be handled by setting the weight of the identity such that it is never chosen in a given section. Bayes' Theorem could be used to incorporate flag information in circuit-level syndrome extraction. It's still unclear at this preliminary stage if this is the best method or if, for example, mimicking MWPM's 3D matching graph is more useful. In general, we are positively hopeful both spatial and temporal correlations

can be handled by minimal modifications to the previous chapters.

It remains to be seen how trellis decoding can benefit qudit codes such as those in [92].

4.1 Stabilizer Codes For Modular Architectures

Attempts to increase the (psuedo)threshold with more detailed simulations and better decoders should be performed in tandem with device specific QECC design for best results. One area of practical interest where more detailed modeling and decoding can provide notable improvements is modular quantum computing architectures. Current state-of-the-art designs and fabrication techniques combined with physical constraints dramatically limit the number of qudits (currently qubits) in near-term quantum devices. It is expected that near-term intermediate scale quantum (NISQ) technologies will consist largely of smaller hardware modules interconnected in a larger plug-and-play type system.

Significant attention has been paid in recent years to optimizing quantum algorithms for NISQ devices. All physical implementations of quantum computers large enough to run quantum algorithms are going to suffer from some form of limited qubit connectivity [226]. Hardware specific compiling and scheduling have a profound effect on circuit depth (size), overhead reduction, and error rate. This will likely be necessary for the foreseeable future as the community continues to develop both efficient high-level and low-level quantum programming languages. However, many of these optimizations are implicitly running on encoded qubits. Often excluded in the analysis is the underlying error-correcting code. QECCs have long been studied with respect to realistic error models, parallel measurement scheduling, resource minimization, circuit-level optimizations, and transversal gates, but it is less common to see codes developed for a specific qubit connectivity. This constraint is less likely to be important for many of today's popular codes which tend to have small length and support only a single or few logical qubits, but certain applications of quantum computers may require longer, high-rate codes to increase feasibility.

There are numerous obstacles to scaling current quantum technologies to larger devices.

The distributed quantum computing (DQC) approach to overcoming this is to network a number of high-fidelity, smaller modules which are easier to manufacture and control. Each module should have its own storage and ancilla qubits and a communication interface to at least one other module. The interface could share classical or quantum information and can move quantum data or teleport gates between modules. A sizable number of works have aimed to optimize intermodule entanglement protocols, especially in the context of one-way or measurement-based quantum computing where probabilistic methods are more tolerable.

Traditional discussions of DQC are usually described as either top-down (completely theoretical) or bottom-up (realistic modeling of a physical device). QECCs are mostly top-down, although they may be simulated against realistic error models, and assume the entire code fits inside a single module and any single-qubit (multi-qubit) operation has equal fidelity (between any pair of qubits). Depending on the length of the code, this may or may not be realistic. It is then the experimentalist's job to come up with an implementation of a given code on their system. Here we consider a meet-in-the-middle approach where we would like to stay as theoretical as possible while also acknowledging the underlying distributed architecture.

To accomplish this, we propose studying codes whose stabilizer matrix is in an ABDF where each block corresponds to a single module. The code cannot be decomposable (a direct sum), the modules need to be entangled; so the stabilizers cannot be put into a true block-diagonal form, but we attempt to minimize the number of stabilizers requiring slow, expensive, and noisy intermodule communication. This decreases the overall error rate and increases parallelism since measurements may be made in each module independently. We remain completely agnostic as to the specific technology used to implement each module, and the modules need not be homogenous. We ignore network topology and assume a pair of modules are connected if necessary. Stabilizers contained in a module are allowed to be any weight. Some modules, qudits, or stabilizers may be more noisy than others,

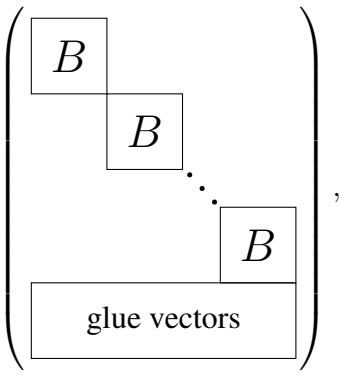


Figure 4.1: The ABDF targeted in this section.

introducing a space-like component to the standard time-like error models, as previously discussed.

Block-diagonal forms are common in mathematics, but ABDFs are not. To demonstrate how this can be done, we present two approaches of constructing ABDF codes: using subcodes of existing codes (Sections 4.1.1) and using existing codes on each module then constructing a larger code which includes this as a subcode (Section 4.1.2). As an example of the former, we examine CSS codes whose stabilizers use a lesser-known almost block-diagonal decomposition from classical coding theory. Unfortunately, the examples here are instructive but not yet promising. More work is required to comb through the sizable space of potential codes. Perhaps the most interesting example, the celebrated 15 qubit QRM code, is a negative result, exemplifying the need for more study. We discuss the open questions stemming from this in Section 4.2.

Formally, consider n_h hardware modules each contributing n_q qubits to the implementation of a single stabilizer code of length $n = n_h n_q$. We would like both the X and Z stabilizers to be put into the ABDF of Figure 4.1, where there are n_h copies of some length n_q code B. The remaining stabilizers serve to “glue" the smaller codes together.

4.1.1 Modular Quantum CSS Codes From Expanded RS Codes

Vardy and Be'ery proved in 1991 that expanded, cyclic RS codes have exactly the structure of Figure 4.1 [227].

Theorem 4.1.1 (Vardy-Be'ery decomposition (VBD))

Let $\mathcal{C}$ be a cyclic RS code over $\mathbb{F}_{p^m}$ for $p, m \geq 2$. Using column permutations, the expanded generator matrix, G_ϕ (Equation (1.3)*), can be put into the form of Figure 4.1, where B is the generator matrix of the corresponding BCH subfield subcode $\mathcal{B}$.*

The proof of this is easy. Instead of expanding $\mathcal{C}$ in its entirety, first perform the expansion on $\mathcal{B} \subset \mathcal{C}$ only. As a subfield subcode, the codewords of $\mathcal{B}$ are closed in $\mathcal{C}$ under scalar multiplication in $\mathbb{F}_{p^m}$. Choose a basis, β, of $\mathbb{F}_{p^m}/\mathbb{F}_p$ and let $\mathcal{B}_i = \{\beta_i b \,:\, b \in \mathcal{B}\}$. Clearly the $\mathcal{B}_i$ are disjoint subcodes of $\mathcal{C}$. For $b = (b_1, \ldots, b_n)$, a row of the generator matrix of $\mathcal{B}$, the expansion of $\beta_i b$ gives m rows of the form

$$\begin{aligned}
&(b_1, 0, \ldots, 0, b_2, 0, \ldots, 0, b_n, 0, \ldots, 0),\\
&(0, b_1, 0, \ldots, 0, b_2, 0, \ldots, 0, b_n, 0, \ldots, 0),\\
&(0, 0, b_1, 0, \ldots, 0, b_2, 0, \ldots, 0, b_n, 0, \ldots, 0),
\end{aligned} \tag{4.1}$$

where we have used the fact that $\mathrm{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_p}$ is $\mathbb{F}_p$-linear and $\mathrm{Tr}_{\mathbb{F}_{p^m}/\mathbb{F}_p}(b_\ell \beta_i \beta_j^\perp) = b_\ell \delta_{ij}$. There are $m-1$ zeros (cyclically) between each non-zero element. Permuting columns put these into the form

$$\begin{aligned}
&(b_1, \ldots, b_n, 0, \ldots, 0, 0, \ldots 0),\\
&(0, \ldots, 0, b_1, \ldots, b_n, 0, \ldots 0),\\
&(0, \ldots, 0, 0, \ldots 0, b_1, \ldots, b_n).
\end{aligned}$$

Repeating this for all of the rows of the generator matrix of $\mathcal{B}$ then permuting rows completes the m factors of $B \oplus \ldots \oplus B$.

If $\mathcal{C}$ has dimension k and $\mathcal{B}$, k', then $m(k-k')$ more "glue vectors" are required to span the expanded code, $\phi_\beta(\mathcal{C})$. These must be inside of $\phi_\beta(\mathcal{C})$ but with a nonzero component outside out $\mathcal{B}_i$ if we want the generator matrix to be full rank. The remaining vectors are therefore a basis of the row space of $\phi_\beta(\mathcal{C})/\left(\oplus_{i=1}^{m} B_i\right)$. The literature often describes the glue vectors as sums of minimum-weight coset leaders of $\mathcal{B}$ considered as polynomials also satisfying the zeros of $\mathcal{C}$ [228]; however, the standard coset leaders algorithm makes this difficult to use for even small codes. Instead, these may be computed, even for large codes, using elementary linear algebra using the same algorithm one would for computing the quotient space of two modules. It may be experimentally advantageous to further enforce that the glue vectors be of specific weight or as low-weight as possible. This can be done by selecting appropriate elements from $\phi_\beta(\mathcal{C})/\left(\oplus_{i=1}^{m} B_i\right)$ using the techniques of the previous chapter.

Vardy and Be'Ery provide no insight as to when a non-trivial decomposition exists in the sense that we desire $k \neq 0$, $3 \leq d < n$. There are too many parameters in the problem to prove a general statement about this. It's possible to prove some number-theoretic statements when the length of the code is of a specific form; however, we do not see the utility in performing this exercise here. Instead, fixing all the parameters we demonstrate how to easily determine when a non-trivial VBD exists. We do this over the low-dimensional qudit systems $\mathbb{F}_2, \mathbb{F}_3$, and $\mathbb{F}_5$ and stick to codes of moderate length $n \leq 200$. Codes over $\mathbb{F}_{2^6}, \mathbb{F}_{3^4}, \mathbb{F}_{5^3}$, and larger fields produce expanded codes that are too long.

We prove the first two statements of the following proposition. The proofs for (iii) - (vii) follow the same pattern but are tedious, since instead of finding contradictions one must now keep track of all of the examples constructed by checking all possible parameter choices. The RS codes using the field extensions in (i) and (ii) are too small to fit non-trivial subfield subcodes. These cases are a bit contrived as one would not, for example, make modules of length three to implement a code of length six. We consider them here to demonstrate the proof technique. Examples are plentiful in (iii) - (vii), however, not all

of them are good in the sense that the ratio of glue vectors to the total number of rows could easily be over 50%. These codes defeat the purpose of this application, so instead we search for codes with the number of glue vectors less than some threshold.

Proposition 4.1.2

(i) There are no stabilizer codes with both X and Z stabilizers supporting a non-trivial VBD for $\mathbb{F}_4/\mathbb{F}_2$.

(ii) There are no stabilizer codes with both X and Z stabilizers supporting a non-trivial VBD for $\mathbb{F}_8/\mathbb{F}_2$.

(iii) There are four VBDs for $\mathbb{F}_{16}/\mathbb{F}_2$ where less than 20% of the rows are glue vectors. All of them have between 15% - 17% glue vectors.

(iv) There are eight VBDs for $\mathbb{F}_{32}/\mathbb{F}_2$ where less than 20% of the rows are glue vectors. All of them have between 10% - 14% glue vectors.

(v) There are six VBDs for $\mathbb{F}_9/\mathbb{F}_3$ where less than 20% of the rows are glue vectors. All of them have between 16% - 20% glue vectors.

(vi) There are 40 VBDs for $\mathbb{F}_{27}/\mathbb{F}_3$ where less than 20% of the rows are glue vectors. All of them have between 8% - 18% glue vectors.

(vii) There are 88 VBDs for $\mathbb{F}_{25}/\mathbb{F}_5$ where less than 20% of the rows are glue vectors. All of them have between 4% - 20% glue vectors.

Proof. Let the superscript $C_s^{p^m}$ denote the q in q-coset, and recall that the cyclotomic cosets of the RS codes over the extension fields are always of size one. RS codes with $\delta = 2$ have a single cyclotomic coset and therefore parameters $[n, n-1, 2]_{p^m}$, which we ignore as we require $3 \leq d < n$. We also define codes to have support on all of their bits, so all binary $[n, 1]$ codes are necessarily the $[n, 1, n]$ repetition code, which we are also not interested in.

(i) For $\mathbb{F}_4$ we have $C_0^2 = \{0\}$ and $C_1^2 = \{1, 2\}$. To ensure $d \leq 3$, $\delta \geq 2$. If $\delta > 2$, then all cyclotomic cosets are chosen for the generator polynomial and the resulting code

has dimension 0. Therefore $\delta = 2$ and either $b = 0$ or $b = 1$. If $b = 0$, the RS code is $[3, 2, 2]_4$, the BCH subcode is $[3, 2, 2]_2$, and the expanded code is $[6, 4, \geq 2]_2$. There are therefore no glue vectors and the code is decomposable. If $b = 1$, we may choose $\mathcal{T}_1 = C_1^4$ or $\mathcal{T}_2 = C_1^4 \cup C_2^4$. The code $\mathcal{T}_1$ has parameters $[3, 2, 2]_3$ with subcode $[3, 1, 3]_2$ and expanded code $[6, 4, \geq 2]_2$. This has two glue vectors. Considering this generator matrix as either X or Z stabilizers leaves the other with exactly one stabilizer with support over all six qubits. This could be beneficial if so desired, but does not satisfy the ABDF requirement chosen for our model. The code $\mathcal{T}_2$ is $[3, 1, 3]_4$ with subcode $[3, 1, 3]_2$, and the $[6, 2, \geq 3]_2$ expanded code has no glue vectors. Alternatively, we could have disqualified any $[n, 1, n]$ repetition code for not meeting our criteria.

(ii) The 2-cyclotomic cosets modulo 7 are $C_0^2 = \{0\}$, $C_1^2 = \{1, 2, 4\}$, and $C_3^2 = \{3, 5, 6\}$.

For $b = 0$: $2 < \delta < 5$ since for $\delta \geq 5$ the subfield subcode has dimension 0.

— $\delta = 3 \rightarrow \mathcal{T}_{\text{RS}} = C_0^8 \cup C_1^8$

RS code: $[7, 5, 3]_8$, subcode: $[7, 3, \geq 3]_2$, expanded code: $[21, 15, \geq 3]_2$, # glue: 6

— $\delta = 4 \rightarrow \mathcal{T}_{\text{RS}} = C_0^8 \cup C_1^8 \cup C_2^8$

RS code: $[7, 4, 4]_8$, subcode: $[7, 3, \geq 3]_2$, expanded code: $[21, 12, \geq 3]_2$, # glue: 3

For $b = 1$: $2 < \delta < 4$ since for $\delta \geq 4$ the subfield subcode is a repetition code or dimension 0.

— $\delta = 3 \rightarrow \mathcal{T}_{\text{RS}} = C_1^8 \cup C_2^8$

RS code: $[7, 5, 3]_8$, subcode: $[7, 4, \geq 3]_2$, expanded code: $[21, 15, \geq 3]_2$, # glue: 3

For $2 \leq b \leq 4$: all cases lead to the repetition code or dimension 0.

For $b = 5$: $2 < \delta < 5$.

— $\delta = 3 \rightarrow \mathcal{T}_{\rm RS} = C_5^8 \cup C_6^8$

RS code: $[7, 5, 3]_8$, subcode: $[7, 4, \geq 3]_2$, expanded code: $[21, 15, \geq 3]_2$, # glue: 3

— $\delta = 4 \rightarrow \mathcal{T}_{\rm RS} = C_0^8 \cup C_5^8 \cup C_6^8$

RS code: $[7, 4, 3]_8$, subcode: $[7, 3, \geq 3]_2$, expanded code: $[21, 12, \geq 3]_2$, # glue: 3

For $b = 6$: $2 < \delta < 4$.

— $\delta = 3 \rightarrow \mathcal{T}_{\rm RS} = C_0^8 \cup C_6^8$

RS code: $[7, 5, 3]_8$, subcode: $[7, 3, \geq 3]_2$, expanded code: $[21, 12, \geq 3]_2$, # glue: 6

Used as either an X or Z stabilizer, the $[21, 15]_2$ code supports a maximum of five stabilizers of the other type and the $[21, 12]_2$ a maximum of eight. Hence, neither code can be used if we require both the stabilizer types to support the VBD.

□

For the reader's convenience in deriving the results in (iii) - (vii):

— The 2-cosets modulo 15 are:

$$C_0^{16} = \{0\} \quad , \quad C_1^{16} = \{1, 2, 4, 8\} \quad , \quad C_3^{16} = \{3, 6, 9, 12\}$$
$$C_5^{16} = \{5, 10\} \quad , \quad C_7^2 = \{7, 11, 13, 14\}.$$

— The 2-cosets modulo 31 are:

$$C_0^{32} = \{0\} \quad , \quad C_1^{32} = \{1, 2, 4, 8, 16\} \quad , \quad C_3^{32} = \{3, 6, 12, 17, 24\},$$
$$C_5^{32} = \{5, 9, 10, 18, 20\} \quad , \quad C_7^{32} = \{7, 14, 19, 25, 28\} \quad , \quad C_{11}^{32} = \{11, 13, 21, 22, 26\},$$

$$C_{15}^{32} = \{15, 23, 27, 29, 30\}.$$

— The 3-cosets modulo 8 are:

$$C_0^9 = \{0\} \quad , \quad C_1^9 = \{1, 3\} \quad , \quad C_2^9 = \{2, 6\} \quad , \quad C_4^9 = \{4\} \quad , \quad C_5^9 = \{5, 7\}.$$

— The 3-cosets modulo 26 are:

$$C_0^{27} = \{0\} \quad , \quad C_1^{27} = \{1, 3, 9\} \quad , \quad C_2^{27} = \{2, 6, 18\} \quad , \quad C_4^{27} = \{4, 10, 12\},$$
$$C_5^{27} = \{5, 15, 19\}, \quad , \quad C_7^{27} = \{7, 11, 21\} \quad , \quad C_8^{27} = \{8, 20, 24\},$$
$$C_{13}^{27} = \{13\} \quad , \quad C_{14}^{27} = \{14, 16, 22\} \quad , \quad C_{17}^{27} = \{17, 23, 25\}.$$

— The 5-cosets modulo 24 are:

$$C_0^{25} = \{0\} \quad , \quad C_1^{25} = \{1, 5\} \quad , \quad C_2^{25} = \{2, 10\} \quad , \quad C_3^{25} = \{3, 15\},$$
$$C_4^{25} = \{4, 20\} \quad , \quad C_6^{25} = \{6\} \quad , \quad C_7^{25} = \{7, 11\} \quad , \quad C_8^{25} = \{8, 16\},$$
$$C_9^{25} = \{9, 21\} \quad , \quad C_{12}^{25} = \{12\} \quad , \quad C_{13}^{25} = \{13, 17\} \quad , \quad C_{14}^{25} = \{14, 22\},$$
$$C_{18}^{25} = \{18\} \quad , \quad C_{19}^{25} = \{19, 23\}.$$

Remark: The process of mapping a cyclotomic coset from an extension field to a subfield is sometimes referred to as coset aliasing.

This method of obtaining an ABDF is highly constrained. For a given n_h, the RS code has length $p^m - 1$, and the expanded code has length $n_h(p^m - 1)$, which is why we did not need to specify the lengths of the codes in the statement of the previous theorem. To build a stabilizer code, we may use the CSS constructions CSS$(\mathcal{C}_2, \mathcal{C}_1)$, CSS$(\mathcal{C})$, or otherwise ensure that the stabilizers are symplectic orthogonal. Section 1.3.3 shows that this requires nested codes and Section 1.1 gives conditions for when two cyclic codes are nested. The

expansion must preserve the orthogonality of the original codes, so the discussion of self-dual bases in Section 1.1 applies. As mentioned in that section, different bases may give expanded codes with different properties, and this extends to quantum properties such as transversal gates as well.

In all cases (iii) - (vii), finding “good" CSS codes with both X and Z supporting a non-trivial VBD requires

(i) enumerating all possible VBDs

(ii) enumerating all nested combinations of these

(iii) enumerating all bases

(iv) computing the properties of every nested combination with every basis.

We could relax the requirement that both X and Z must support a VBD to only one must support a VBD to significantly increase the number of possible codes. We can also derive non-CSS stabilizer codes from classical codes using the Hermitian construction, but we did not cover this in Chapter 1 and therefore will not discuss this further here.

4.1.2 Modular Quantum CSS Codes From BCH Codes Of Composite Length

In a separate 1994 paper [229], Vardy and Be'ery show that binary, primitive BCH codes and binary BCH codes of composite block length may also be put into ABDF. We will also refer to this as a VBD since the proper technique should be clear from context. For primitive BCH codes, they extended the code and then split the zeros into partitions satisfying certain properties. The direct-sum subcodes are then obtained by puncturing on the set complement of the indices corresponding to the defining sets of each partition. This applies directly to RM codes. We will not use this approach here but instead consider BCH codes of composite block length. The two approaches are almost identical except that in the latter case the partitions are immediate from the structure of the code. The following applies to

cyclic codes in general, however, we stick to BCH codes here for the minimum distance guarantees provided by the BCH bound.

Let $\mathcal{C}$ be a BCH code over $\mathbb{F}_2$ of composite length $n = n_h n_q$ with defining set $C_b^n \cup \ldots \cup C_{b+\delta-2}^n$. Consider the sets $\mathcal{I}_i = \{1 + j + i \cdot n_h\}$ where $0 \leq j \leq n_h - 1$ for fixed $0 \leq i \leq n_q - 1$.

Theorem 4.1.3 (Proposition 2 [229])

The code obtained from $\mathcal{C}$ punctured on the complement, $\mathcal{I}_i^c$, is a BCH code of length n_q with defining set $C_b^{n_q} \cup \ldots \cup C_{b+\delta-2}^{n_q}$.

For example, consider the BCH code with $b = 0$ and $\delta = 5$ constructed with $n_h = 3$ hardware modules of $n_q = 15$ qubits each. The 2-cosets modulo 45 are

$$C_0^{45} = \{0\} \quad , \quad C_1^{45} = \{1, 2, 4, 8, 16, 17, 19, 23, 31, 32, 34, 38\} \quad , \quad C_3^{45} = \{3, 6, 12, 24\},$$

so this has parameters $[45, 28, 6]$. Then

$$\mathcal{I}_1 = \{1, 4, 7, \ldots, 43\} \quad , \quad \mathcal{I}_2 = \{2, 5, 8, \ldots, 44\} \quad , \quad \mathcal{I}_3 = \{3, 6, 9, \ldots, 45\}.$$

The BCH subcode has the defining set $C_0^{15} \cup \ldots \cup C_3^{15}$. The 2-cosets modulo 15 are

$$C_0^{15} = \{0\} \quad , \quad C_1^{15} = \{1, 2, 4, 8\} \quad , \quad C_3^{15} = \{3, 6, 9, 12\},$$
$$C_5^{15} = \{5, 10\} \quad , \quad C_7^{15} = \{7, 11, 13, 14\}.$$

giving a $[15, 6, 6]$ code. Permuting the indices of $\mathcal{I}_1$ to indices 1 - 15, $\mathcal{I}_2$ to 16 - 30, and $\mathcal{I}_3$ to 31 - 45 completes the direct-sum subcode.

While it is not mentioned in [229], the first VBD may seen as a special case of the second. It is well-known in the study of cyclic codes that expanded RS codes are BCH codes. The RS code has length $n_q = p^m - 1$ and the expansion mn_q, where we can identify $n_h = m$. The sets $\mathcal{I}_i$ are the non-zero locations of (4.1).

We can use the previous theorem to construct codes in a similar way to the previous subsection; however, we again run into an exhaustive blind search for a code with good properties. Instead, we propose a novel application of the previous theorem: start with a desirable subcode and then build a BCH supercode around it. At the moment, it is unclear which subcodes are "desirable". Two of the most popular QECCs are the Steane and 15-qubit QRM code, and it is interesting to consider them as subcodes in a toy model. There are a few different ways one could define a family of CSS QRM codes, of which the previous two codes are one example. To use them in the previous theorem, we first show that these RM codes are equivalent to BCH codes. While the RM-BCH relationship is well-known in classical coding theory, the fact that the 15-qubit code is cyclic has not appeared in the QEC literature. The end result of our example is that it cannot be done without expurgating some of the resulting stabilizers, but it is easy to see that this stems from the particular parameters of the code, and other QRM codes in this family are unlikely to run into the same problem. We consider expurgating in the next section.

Let $n = p^m - 1$. The p-weight of an integer $0 \leq a \leq n$ is $\mathrm{wt}_p(a) = \sum_{j=0}^{m-1} a_j$, where $a = \sum_{j=0}^{m-1} a_j p^j$, $0 \leq a_j \leq p-1$ is the p-adic expansion of a. Equivalently, we may interpret a vector in $\mathbb{F}_p^m$ as the coefficients of a p-adic expansion and define the p-weight as the sum of the elements. Consider monomials of the form $x_1^{i_1} x_2^{i_2} \ldots x_m^{i_m}$ for $i_1 + \ldots + i_m \leq r$. Interpreting $(i_1, \ldots, i_m)$ as a p-adic expansion, all cyclic shifts are also valid monomials of total degree less than r, have constant p-weight, and generate the p-coset C_i where $i = \sum_{j=0}^{m-1} i_j p^j$. In this way we establish a correspondence between multivariate polynomials of $\mathrm{RM}^*(r, m)$ and univariate polynomials of BCH codes.

Theorem 4.1.4 ([230])

Let α be a primitive root of $\mathbb{F}_{p^m}^\times$ and define $g_{r.m}^*(x) = \prod(x - \alpha^a)$ where $0 < \mathrm{wt}_p(a) \leq m(p-1) - r - 1$. Then $\mathcal{RM}_{p^m}^*(r, m)$ is permutation equivalent to the subfield subcode of $\mathcal{C}_{r.m}^* = (g_{r.m}^*(x))$ over $\mathbb{F}_p$.

Since the defining set of $\mathcal{C}_{r.m}^*$ is comprised of complete p-cosets, $g_{r.m}^* \in \mathbb{F}_p[x]$ and hence

$\mathcal{C}^*_{r.m} = \mathcal{C}^*_{r.m} \cap \mathbb{F}_p$.

The 15-qubit QRM code is formed from the shortened RM codes $\overline{\mathcal{RM}}(1,4)$ and $\overline{\mathcal{RM}}(2,4)$ with X stabilizers given by $\overline{G}(1,4)$ and Z stabilizers by $\overline{G}(2,4)$. Recalling the relationship between shortened and punctured codes, the X stabilizers are equivalent to a parity check matrix for $\mathcal{RM}^*(2,4)$ and the Z stabilizers are equivalent to a parity check matrix for $\mathcal{RM}^*(1,4)$. It's instructive to construct these explicitly.

We begin with $\mathcal{RM}^*(1,4)$. The set of all integers a with Hamming weight $0 < \mathrm{wt}_2(a) \leq 2$ is

$$\{1,2,3,4,5,6,8,9,10,12\} = C_1^{16} \cup C_3^{16}.$$

The corresponding generator polynomial is $g^*_{1,4}(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$. For $\mathcal{RM}^*(2,4)$, the set of all integers a with Hamming weight $0 < \mathrm{wt}_2(a) \leq 1$ is $\{1,2,4,8\} = C_1^{16}$. The corresponding generator polynomial is $g^*_{2,4}(x) = 1 + x^2 + x^3 + x^4$. The dual codes have generator polynomials $(g^*_{1,4})^\perp(x) = 1 + x^2 + x^4 + x^5$ and $(g^*_{2,4})^\perp(x) = 1 + x^3 + x^4 + x^6 + x^8 + x^9 + x^{10} + x^{11}$. Generator matrices for these are

$$\overline{G}(1,4) = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{4.2}$$

and

$$\overline{G}(2,4) = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}. \tag{4.3}$$

A set of explicit stabilizers for the 15-qubit QRM code are given in [231] as

$$G_X = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{4.4}$$

and

$$G_Z = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{4.5}$$

One may check that $\mathrm{rowspace}(\overline{G}(1,4)) \cong \mathrm{rowspace}(G_X)$ and $\mathrm{rowspace}(\overline{G}(2,4)) \cong \mathrm{rowspace}(G_Z)$ via the permutation $(3\ 5\ 9\ 15\ 13\ 14\ 12\ 7\ 11\ 8\ 4)$.

It remains to show that these two codes can form direct-sum subcodes of BCH codes such that the number of glue vectors is greater than zero but less than some reasonable percentage of the overall code. The X code has parameters $[15, 4]$ and Z, $[15, 10]$. The number n_h cannot be even because cyclic codes require $\gcd(n, p) = 1$, so the first test case is $n_h = 3$. The 2-cosets modulo 45 are given in the previous section. The duals have defining sets $\mathcal{T}_X = C_0^{15} \cup C_1^{15} \cup C_3^{15} \cup C_7^{15}$ ($b = 0, \delta = 7$) and $\mathcal{T}_Z = C_0^{15} \cup C_1^{15}$ ($b = 0, \delta = 4$), so the corresponding BCH supercodes for X has parameters $[45, 22]$ and $[45, 32]$ for Z. There are therefore ten X stabilizer glue vectors and Z has two. However, the combined stabilizer code encodes no logical qubits.

The problem with this code is there are too many Z stabilizers. The coset C_0 always has a single element, and the cardinality of all other cosets must divide $\mathrm{ord}_n(p)$, which is exactly $|C_1|$. Since the dimension of the code is $n - |\mathcal{T}| = n - 1 - \mathrm{ord}_n(p)$, reducing the number of stabilizers requires $\mathrm{ord}_n(p)$ to be large, which simply does not occur for the n_h in the ranges we are interested in. Thus, in order to make a stabilizer code with these X

modules and these Z modules, we would have to expurgate glue vectors. The question then becomes deeper: which stabilizers can be removed and which cannot? We briefly discuss this next.

4.2 Gluing Theory For Stabilizer Codes

To answer the questions asked at the end of the last section, we must answer the more fundamental question of: what is the role of each stabilizer in a code? This question does not make sense in classical coding theory as there is no equivalent concept of encoded operations. For a stabilizer code, one can ask, for example, whether or not a transversal gate remains transversal after a given stabilizer is added or removed. A generator-by-generator study or building of a code is currently missing in the QEC literature. We call this “gluing" in analogy to the way classical self-dual codes can be put together to create larger self-dual codes. The ABDF of the previous section may also be seen in this light. motivating the glue vector terminology. In fact, many QECCs may be seen as descending from this procedure.

The codes of the previous sections had the distinctive feature that the supercode already existed and we merely permuted it to expose a desired form. This has the advantage that the glue vectors are pre-determined and are computable by elementary linear algebra. However, as we’ve seen, this also has its drawbacks. For starters, it may be unclear if a given code may be put into ABDF. When it can, the supercode might not have desirable properties such as transversal gates. The constructions of the previous sections also forced each block to be the same. We tried to improve this by working backwards from a desirable subcode, but the construction was highly specific and still left us with a fixed supercode.

Now we consider the more general scenario of gluing m blocks X_i (Z_i) to make a generating set of X and Z stabilizers, respectively. We require the length of X_i and Z_i to be the same but they may differ for indices $i \neq j$. In particular, the pairs X_i, Z_i and X_j, Z_j need not have any relationship to each other. Since X must commute with Z, $X_i \subset Z_i^{\perp}$ for

all $1 \leq i \leq m$.

Glue vectors are of the form $(g_1, \ldots, g_m)$, where each glue component, g_i, is a vector the length of block i. If only one g_i is nonzero, the vector may be expurgated from the glue and augmented to the appropriate block as a stabilizer, if it is not already one. So we may assume without loss of generality that at least two g_i are nonzero for every glue vector. Codes with decomposable parts are not physically interesting, so we require that g_i is nonzero for at least one glue vector for all i.

We analyze the X glue vectors, the Z discussion is similar. The only constraint on the g_i is that it also commutes with Z_i. There are three ways to select them:

Type I: glue vectors may consist entirely of stabilizers ($g_i \in X_i$) by removing generators from a block and concatenating them in the glue

Type II: glue vectors may consist entirely of logical operators of each block ($g_i \in Z_i^{\perp} \backslash X_i$)

Type III: glue vectors may consist of a combination of logicals and stabilizers removed from blocks.

Not all three types of gluing are available in every scenario. For example, taking the direct sum of two $[[n_i, 1, d_i]]$ CSS codes gives $n_1 + n_2 - 2$ stabilizers. Assuming both X and Z need at least one glue vector, a Type II gluing produces a $k = 0$ code.

Example 16. The $[[7, 1, 3]]$ Steane code has X and Z stabilizers

$$f_1 = (1\,0\,1\,0\,1\,0\,1),\ f_2 = (0\,1\,1\,0\,0\,1\,1),\ f_3 = (0\,0\,0\,1\,1\,1\,1),$$

and the all-ones vector is a logical representative L. One possible Type I gluing for two copies of this code would be

$$S_{X/Z} = \{(f_1 \mid 0), (f_2 \mid 0), (0 \mid f_1), (0 \mid f_2), (f_3 \mid f_3)\},$$

where $(\cdot \mid \cdot)$ denotes vector concatenation. One possible Type III gluing would be

$$\begin{aligned} S_X &= \{(f_1 \mid 0), (f_2 \mid 0), (0 \mid f_1), (0 \mid f_2), (f_3 \mid f_3)\}, \\ S_Z &= \{(f_1 \mid 0), (f_2 \mid 0), (f_3 \mid 0), \\ &\qquad (0 \mid f_1), (0 \mid f_2), (0 \mid f_3), (L \mid L). \end{aligned}$$

The first example has parameters $[[14, 4, 2]]$ and the second $[[14, 2, 2]]$. To understand this distance drop notice that for every stabilizer there is a weight-one error which anticommutes with only that stabilizer. Then any $(f_i \mid f_j)$ has a commuting weight-two error. (It may be helpful to picture the Steane code as the $d = 3$ triangular color code, in which case these errors are on the outer corners of Figure 1.6a.)

Example 17. It is well-known that the QRM code contains two copies of the Steane code plus an extra qubit. We may consider this as an inhomogeneous modular architecture with blocks of sizes seven, one, and seven and codes $[[7, 1, 3]]$, $[[1, 1, 1]]$, and $[[7, 1, 3]]$, respectively. Then the stabilizers (4.4) and (4.5) are of the form

$$G_X = \{(f_1 \mid 0 \mid f_1), (f_2 \mid 0 \mid f_2), (f_3 \mid 0 \mid f_3), (0 \mid 1 \mid L)\}$$

and

$$\begin{aligned} G_Z = \{&(f_1 \mid 0 \mid f_1), (f_2 \mid 0 \mid f_2), (f_3 \mid 0 \mid f_3), (0 \mid 1 \mid L), \\ &g_1, g_2, g_3, \\ &(0 \mid 0 \mid f_3), (0 \mid 0 \mid f_2), (0 \mid 0 \mid f_1)\}, \end{aligned}$$

respectively. The last three elements of G_Z may be used to remove the second Steane copy

from the first three elements leaving the ABDF

$$\begin{aligned} G_Z = \{&(f_1 \mid 0 \mid 0), (f_2 \mid 0 \mid 0), (f_3 \mid 0 \mid 0), \\ &(0 \mid 0 \mid f_3), (0 \mid 0 \mid f_2), (0 \mid 0 \mid f_1), \\ &(0 \mid 1 \mid L), g_1, g_2, g_3\}, \end{aligned}$$

where the last four elements are Type II glue vectors. The inclusion of the logical prevents the distance drop from the Type I gluing in the previous example, and the remaining three rows of (4.5), $\{g_1, g_2, g_3\}$, may be interpreted as glue vectors. Alternatively, we could have started with two copies of Steane and tried to target a transversal T logical gate. The Hamming weight-eight condition for T forces a gluing similar to G_X with an extra qubit because (the usual representative of) L has weight seven. We also know that we need $\mathrm{rowspace}(G_X) \subset \mathrm{rowspace}(G_Z)$, leaving a $[[15, 7, 3]]$ code. Desiring $k = 1$, we can decouple the rows of G_Z back into blocks since they are not required to be higher weight to get a $[[15, 4, 3]]$ code. Type II glue vectors may be chosen to reduce this to $[[15, 1, 3]]$.

Remark: This code had previously only been derived via puncturing and shortening and happens to come out with the desirable properties it has. The cyclic derivation of the previous section is new and demonstrates a previously unknown internal structure of the code, but was also not very illuminating. Gluing theory was able to derive it with just a few guiding principles. The approach of the previous example also shows that stabilizers of the form $(f_i \mid 0 \mid f_j)$ with $i \neq j$ are also viable, representing a permutation of the second Steane block. Most importantly, we see that from this perspective the stabilizers $\{g_1, g_2, g_3\}$ only exist for a singular reason: to lower the number of encoded qubits. These may therefore be selected arbitrarily from $G_X^{\perp}$ as long as they satisfy the weight requirements for maintaining the transversal T gate.

Converting between the Steane and QRM codes is a leading method to circumvent the Eastin-Knill theorem to obtain a transversal, universal gate set. The logical gate set of the input and output codes of the gluing are thus not only changed, but in this extreme example, complementary. An interesting question is whether or not this is generic. To answer this, we recall the fundamentals of the stabilizer formalism from Section 1.3.2.

To align with the rest of the coding theory literature, we refer to the process of removing a stabilizer generator as expurgating and adding a generator by augmenting. Fix an ordering of stabilizer generators $\mathcal{S} = \langle S_1, \ldots, S_{n-k} \rangle$ and let

$$\mathcal{Q}^{(i)} = \bigcap_{j=1}^{i} \{v \in \mathbb{C}^{q^n} \mid S_j v = v\} \tag{4.6}$$

be the quantum stabilizer code for part of $\mathcal{S}$ and $\mathcal{Q}$ be the full quantum stabilizer code. Each stabilizer S_i adds a new constraint, fixing a degree of freedom in $\mathbb{C}^{q^n}$. Consider expurgating, without loss of generality, the generator S_{n-k} to form a new stabilizer group $\mathcal{S} \backslash \langle S_{n-k} \rangle$. Then $\mathcal{Q} \subset \mathcal{Q}^{(n-k-1)}$ with co-dimension one. A basis for the new space is obtained by extending the old basis $\{u_i, v_i\}$ by one pair. In particular, none of the existing logical operators $\{\overline{X}_i, \overline{Z}_i\}$ are modified, so the stored information and operators acting as linear combinations of these matrices are preserved.

Now consider augmenting S_{n-k+1}: $\mathcal{S} \mapsto= \mathcal{S} \times \langle S_{n-k+1} \rangle$. Then $\mathcal{Q}^{(n-k+1)} \subset \mathcal{Q}$ with co-dimension one. The new stabilizer must commute with all elements $\eta^c X(a) Z(b) \in \mathcal{S}$, which only depends on $(a \mid b) \in \mathbb{F}_p^{2n}$. It therefore suffices to study this in the symplectic formulation via $\Psi : \mathcal{P}_n \to \mathbb{F}_p^{2n}$ (see Section 1.3.2). The problem is then understanding the behavior of the orthogonal complement of a vector space as the vector space grows. Recall $\Psi(\mathcal{S}) \subset \Psi(\mathcal{S}^\perp) \subset \mathbb{F}_p^{2n}$ and that the inner product is given by the symplectic form $\langle (a, b) \mid (c, d) \rangle_s = b \cdot c - d \cdot a$, where $b \cdot a$ is the standard dot (Euclidean inner) product. Since $\mathcal{S}$ is abelian, the form vanishes on $\Psi(\mathcal{S})$, making it a totally isotropic subspace of $\Psi(\mathcal{S}^\perp)$. The generators of $\mathcal{S}$ induce a basis on this space $\{s_1, \ldots, s_{n-k}\}$. The full space $\Psi(\mathcal{S}^\perp)$ is

symplectic and therefore there exists a basis $\{u_1, \ldots, u_{n+k}, v_1, \ldots, v_{n+k}\}$ satisfying

$$\langle u_i, u_j \rangle_s = \langle v_i, v_j \rangle_s = 0 \quad , \quad \langle u_i, v_j \rangle_s = \delta_{ij}.$$

Instead of using this basis, the logical operators are defined by extending the basis of $\Psi(\mathcal{S})$ to $\Psi(\mathcal{S}^\perp)$ by $2k$ operators: $\{s_1, \ldots, s_{n-k}, u_1, \ldots, u_k, v_1, \ldots, v_k\}$. To do this, solve the system of equations

$$\langle s_1, u_i \rangle_s = \ldots = \langle s_{n-k}, u_i \rangle_s = 0 \quad , \quad \langle u_j, u_i \rangle = 0 \tag{4.7}$$

for $u_i \notin \Psi(\mathcal{S})$ for $1 \leq i \leq k$ and $1 \leq j < i$ when $i \neq 1$. Then solve the system of equations

$$\langle s_1, w_i \rangle_s = \ldots = \langle s_{n-k}, w_i \rangle_s = 0 \quad , \quad \langle u_i, w_j \rangle_s = \delta_{ij} \tag{4.8}$$

for w_j for $1 \leq j \leq k$. Finally, set c_{ij} in $v_i = w_i + c_{ij} u_j$ such that $\langle v_i, v_j \rangle_s = 0$.

Codes are created by fixing the degrees of freedom corresponding to $\{s_1, \ldots, s_{n-k}\}$. Information is stored in the remaining degrees of freedom as $v = \sum_{i=1}^{k} a_i u_i + b_i v_i$ with coefficients $a_i, b_i \in \mathbb{F}_p$. Suppose a new stabilizer s_{n-k+1} is added to the basis of $\Psi(\mathcal{S})$. If $s_{n-k+1} \in \Psi(\mathcal{S})$ this has no affect, so assume this is not the case. Without loss of generality, we may consider $s_{n-k+1} \in \Psi(\mathcal{S}^\perp) \backslash \Psi(\mathcal{S})$ such that $s_{n-k+1} = \sum_{i=1}^{k} d_i u_i + e_i v_i$ with $d_i, e_i \in \mathbb{F}_p$.

Consider the special case $s_{n-k+1} = u_1$, v_1, or $u_1 + v_1$. Adding a generator to $\Psi(\mathcal{S})$ drops the dimension of $\Psi(\mathcal{S}^\perp)$ by two. The set $\{s_1, \ldots, s_{n-k}, s_{n-k+1}, u_2, \ldots, u_k, v_2, \ldots, v_k\}$ satisfies Equations (4.7) and (4.8), so $\Psi(\mathcal{S}^\perp)$ only updates by removing u_1 and v_1. If either a_1 or b_1 are nonzero, the information in v is destroyed. In the more general case, any nonzero d_i or e_i causes a violation of the system of equations and must be removed from the basis for $\Psi(\mathcal{S}^\perp)$. If too many basis elements are removed, the system needs to be resolved for new u_i, v_i to complete the set. Sometimes a simple Gram-Schmidt-like procedure can up-

date the old basis, but it is unclear if this procedure is always available. Any information stored using the removed or updated basis elements will be destroyed.

We hope that this insight can serve as a guiding principle pulling all sections of this chapter together: QECCs for modular architectures glued together in such a way that desirable properties of individual codes are not disturbed by the others. This shares similarities with gauge fixing of subsystem codes and may open new directions in this area such as a more general and less intuitive method of construction. The above examples concentrated on CSS codes, but gluing is even easier for general stabilizer codes because only one glue vector is required instead of one for each X and Z type. Further exploration of the connection between cyclotomic cosets, Hamming weight, and logical operators may be necessary to extend the cyclic code examples above.

The number of active generators in the trellis of codes in ABDF is bounded by the size of the block plus the number of glue vectors, which is usually considerably smaller than the number of generators of the full code. One may also consider a 2-stage decoder similar to the concatenated Steane code in Example 9. Detailed modeling and the minimization of noisy qudit interactions using the ABDF may compensate for the sub-optimality of this style of decoder.

REFERENCES

[1] S. Garoufalidis, E. Sabo, and S. Scott, "Exact computation of the n-loop invariants of knots," *Experimental Mathematics*, vol. 25, no. 2, pp. 125–129, 2016.

[2] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[3] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*, 2002.

[4] D. A. Lidar and T. A. Brun, *Quantum error correction*. Cambridge university press, 2013.

[5] J. H. Van Lint, *Coding theory*. Springer, 1971, vol. 201.

[6] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. Elsevier, 1977, vol. 16.

[7] W. C. Huffman and V. Pless, *Fundamentals of error-correcting codes*. Cambridge university press, 2010.

[8] J. A. Wood, "Duality for modules over finite rings and applications to coding theory," *American journal of Mathematics*, pp. 555–575, 1999.

[9] Y. Wu, "On expanded cyclic and reed–solomon codes," *IEEE transactions on information theory*, vol. 57, no. 2, pp. 601–620, 2011.

[10] D. Jungnickel, A. J. Menezes, and S. A. Vanstone, "On the number of self-dual bases of $gf(q^m)$ over $gf(q)$," *Proceedings of the American Mathematical Society*, vol. 109, no. 1, pp. 23–29, 1990.

[11] C. T. Retter, "Orthogonality of binary codes derived from reed-solomon codes," *IEEE transactions on information theory*, vol. 37, no. 4, pp. 983–994, 1991.

[12] G. D. Forney, "Coset codes. ii. binary lattices and related codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1152–1187, 1988.

[13] K. Parthasarathy, *An Introduction To Quantum Stochastic Calculus*. Birkhäuser, 2012, vol. 85.

[14] P. Meyer, *Quantum Probability For Probabilists*. Springer, 2006.

[15] B. Coecke and E. O. Paquette, "Categories for the practising physicist," in *New structures for physics*, Springer, 2010, pp. 173–286.

[16] P. Busch, P. J. Lahti, J.-P. Pellonpää, and K. Ylinen, *Quantum measurement*. Springer, 2016, vol. 890.

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

[18] M. Choi, "Positive linear maps on c^*-algebras," *Canad. J. Math*, vol. 24, 1972.

[19] J. Watrous, *The theory of quantum information*. Cambridge University Press, 2018.

[20] W. Stinespring, "Positive functions on $c*$-algebras," *Proc. Of The AMS*, vol. 6, no. 2, 1955.

[21] M. Choi, "Completely positive linear maps on complex matrices," *Linear Algebra And Its Applications*, vol. 10, no. 3, 1975.

[22] G. Palma, K. Suominen, and A. Ekert, "Quantum computers and dissipation," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 452, 1996.

[23] L. Duan and G. Guo, "Preserving coherence in quantum computation by pairing quantum bits," *Phys. Rev. Lett.*, vol. 79, no. 10, 1997.

[24] P. Zanardi and M. Rasetti, "Noiseless quantum codes," *Phys. Rev. Lett.*, vol. 79, no. 17, 1997.

[25] D. Lidar, I. Chuang, and K. Whaley, "Decoherence-free subspaces for quantum computation," *Phys. Rev. Lett.*, vol. 81, no. 12, 1998.

[26] E. Knill, R. Laflamme, and L. Viola, "Theory of quantum error correction for general noise," *Phys. Rev. Lett.*, vol. 84, no. 11, 2000.

[27] P. Zanardi, "Stabilizing quantum information," *Phys. Rev. A*, vol. 63, no. 1, 2000.

[28] J. Kempe, D. Bacon, D. Lidar, and K. Whaley, "Theory of decoherence-free fault-tolerant universal quantum computation," *Phys. Rev. A*, vol. 63, no. 4, 2001.

[29] C. Bennett, D. DiVincenzo, J. Smolin, and W. Wootters, "Mixed-state entanglement and quantum error correction," *Phys. Rev. A*, vol. 54, no. 5, 1996.

[30] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," *Phys. Rev. A*, vol. 55, no. 2, 1997.

[31] D. Kribs, R. Laflamme, and D. Poulin, "Unified and generalized approach to quantum error correction," *Phys. Rev. Lett.*, vol. 94, no. 18, 2005.

[32] D. Kribs, R. Laflamme, D. Poulin, and M. Lesosky, "Operator quantum error correction," *arXiv preprint quant-ph/0504189*, 2005.

[33] R. Johansen, "Operator quantum error correction," PhD thesis, Københavns Universitet, 2006.

[34] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. Sloane, "Quantum error correction and orthogonal geometry," *Physical Review Letters*, vol. 78, no. 3, p. 405, 1997.

[35] E. Knill, "Non-binary unitary error bases and quantum codes," *arXiv preprint quant-ph/9608048*, 1996.

[36] ——, "Group representations, error bases and quantum codes," *arXiv preprint quant-ph/9608049*, 1996.

[37] A. Klappenecker and M. Rotteler, "Beyond stabilizer codes. i. nice error bases," *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2392–2395, 2002.

[38] A. Klappenecker and M. Rotteler, "On the monomiality of nice error bases," *IEEE transactions on information theory*, vol. 51, no. 3, pp. 1084–1089, 2005.

[39] R. Howe, "Nice error bases, mutually unbiased bases, induced representations, the heisenberg group and finite geometries," *Indagationes Mathematicae*, vol. 16, no. 3-4, pp. 553–583, 2005.

[40] R. J. Higgs, "Nice error bases and sylow subgroups," *IEEE transactions on information theory*, vol. 54, no. 9, pp. 4199–4207, 2008.

[41] R. Cleve and D. Gottesman, "Efficient computations of encodings for quantum error correction," *Physical Review A*, vol. 56, no. 1, p. 76, 1997.

[42] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, no. 2, p. 1098, 1996.

[43] A. M. Steane, "Error correcting codes in quantum theory," *Physical Review Letters*, vol. 77, no. 5, p. 793, 1996.

[44] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *arXiv preprint quant-ph/9707021*, 1997.

[45] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," *arXiv preprint quant-ph/9811052*, 1998.

[46] M. H. Freedman and D. A. Meyer, "Projective plane and planar quantum codes," *Foundations of Computational Mathematics*, vol. 1, no. 3, pp. 325–332, 2001.

[47] H. Bombín and M. Martin-Delgado, "Topological quantum error correction with optimal encoding rate," *Physical Review A*, vol. 73, no. 6, p. 062 303, 2006.

[48] C. Albuquerque, R. Palazzo Jr, and E. Silva, "Topological quantum codes on compact surfaces with genus $g \geq 2$," *Journal of Mathematical Physics*, vol. 50, no. 2, p. 023 513, 2009.

[49] H. Bombín and M. A. Martin-Delgado, "Homological error correction: Classical and quantum codes," *Journal of mathematical physics*, vol. 48, no. 5, p. 052 105, 2007.

[50] G. Zémor, "On cayley graphs, surface codes, and the limits of homological coding for quantum error correction," in *International Conference on Coding and Cryptology*, Springer, 2009, pp. 259–273.

[51] J. T. Anderson, "Homological stabilizer codes," *Annals of Physics*, vol. 330, pp. 1–22, 2013.

[52] N. Delfosse, P. Iyer, and D. Poulin, "Generalized surface codes and packing of logical qubits," *arXiv preprint arXiv:1606.07116*, 2016.

[53] P. Vrana and M. Farkas, "Homological codes and abelian anyons," *arXiv preprint arXiv:1505.01001*, 2015.

[54] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, no. 3, p. 032 324, 2012.

[55] Y. Tomita and K. M. Svore, "Low-distance surface codes under realistic quantum noise," *Physical Review A*, vol. 90, no. 6, p. 062 320, 2014.

[56] Wikipedia, *Truncated square tiling — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Truncated%20square%20tiling&oldid=1044805958, [Online; accessed 17-March-2022], 2022.

[57] ——, *Hexagonal tiling — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Hexagonal%20tiling&oldid=1075758815, [Online; accessed 17-March-2022], 2022.

[58] ——, *Truncated trihexagonal tiling — Wikipedia, the free encyclopedia*, http://en.wikipedia.org/w/index.php?title=Truncated%20trihexagonal%20tiling&oldid=1032204050, [Online; accessed 17-March-2022], 2022.

[59] C. Trout, "Methods for universal fault-tolerant quantum computation in small devices," PhD thesis, Georgia Institute of Technology, 2018.

[60] A. Kubica and M. E. Beverland, "Universal transversal gates with color codes: A simplified approach," *Physical Review A*, vol. 91, no. 3, p. 032 330, 2015.

[61] A. M. Kubica, "The abcs of the color code: A study of topological quantum codes as toy models for fault-tolerant quantum computation and quantum phases of matter," PhD thesis, California Institute of Technology, 2018.

[62] D. E. Deutsch, "Quantum computational networks," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 425, no. 1868, pp. 73–90, 1989.

[63] N. J. Ross and P. Selinger, "Optimal ancilla-free clifford+ t approximation of z-rotations," *arXiv preprint arXiv:1403.2975*, 2014.

[64] A. Kissinger and J. van de Wetering, "Reducing t-count with the zx-calculus," *arXiv preprint arXiv:1903.10477*, 2019.

[65] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical review A*, vol. 52, no. 5, p. 3457, 1995.

[66] C. Dawson and M. Nielsen, "The solovay-kitaev algorithm," *arXiv preprint quant-ph/0505030*, 2005.

[67] K. Karnas, "Universality in quantum computation," 2018.

[68] D. Gottesman, "The heisenberg representation of quantum computers," *arXiv preprint quant-ph/9807006*, 1998.

[69] D. Gottesman and I. L. Chuang, "Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations," *Nature*, vol. 402, no. 6760, pp. 390–393, 1999.

[70] B. Zeng, X. Chen, and I. L. Chuang, "Semi-clifford operations, structure of c k hierarchy, and gate complexity for fault-tolerant quantum computation," *Physical Review A*, vol. 77, no. 4, p. 042 313, 2008.

[71] S. Beigi and P. W. Shor, "C3, semi-clifford and generalized semi-clifford operations," *arXiv preprint arXiv:0810.5108*, 2008.

[72] I. Bengtsson, K. Blanchfield, E. Campbell, and M. Howard, "Order 3 symmetry in the clifford hierarchy," *Journal of Physics A: Mathematical and Theoretical*, vol. 47, no. 45, p. 455 302, 2014.

[73] S. X. Cui, D. Gottesman, and A. Krishna, "Diagonal gates in the clifford hierarchy," *Physical Review A*, vol. 95, no. 1, p. 012 329, 2017.

[74] N. Rengaswamy, R. Calderbank, and H. D. Pfister, "Unifying the clifford hierarchy via symmetric matrices over rings," *Physical Review A*, vol. 100, no. 2, p. 022 304, 2019.

[75] T. Pllaha, N. Rengaswamy, O. Tirkkonen, and R. Calderbank, "Un-weyl-ing the clifford hierarchy," *Quantum*, vol. 4, p. 370, 2020.

[76] J. Hu, Q. Liang, and R. Calderbank, "Climbing the diagonal clifford hierarchy," *arXiv preprint arXiv:2110.11923*, 2021.

[77] A. R. Calderbank, E. M. Rains, P. Shor, and N. J. Sloane, "Quantum error correction via codes over gf (4)," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1369–1387, 1998.

[78] J. Tolar, "On clifford groups in quantum computing," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1071, 2018, p. 012 022.

[79] V. Teska, "Clifford groups in quantum computing,"

[80] S. Bravyi and D. Maslov, "Hadamard-free circuits expose the structure of the clifford group," *IEEE Transactions on Information Theory*, vol. 67, no. 7, pp. 4546–4563, 2021.

[81] S. Huang and K. R. Brown, "Between shor and steane: A unifying construction for measuring error syndromes," *Physical review letters*, vol. 127, no. 9, p. 090 505, 2021.

[82] R. Chao and B. W. Reichardt, "Quantum error correction with only two extra qubits," *Physical review letters*, vol. 121, no. 5, p. 050 502, 2018.

[83] ——, "Fault-tolerant quantum computation with few qubits," *npj Quantum Information*, vol. 4, no. 1, pp. 1–8, 2018.

[84] T. Tansuwannont, C. Chamberland, and D. Leung, "Flag fault-tolerant error correction, measurement, and quantum computation for cyclic calderbank-shor-steane codes," *Physical Review A*, vol. 101, no. 1, p. 012 342, 2020.

[85] G. B. Horn and F. R. Kschischang, "On the intractability of permuting a block code to minimize trellis complexity," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 2042–2048, 1996.

[86] B. Eastin and E. Knill, "Restrictions on transversal encoded quantum gate sets," *Physical review letters*, vol. 102, no. 11, p. 110 502, 2009.

[87] A. M. Steane and B. Ibinson, "Fault-tolerant logical gate networks for calderbank-shor-steane codes," *Physical Review A*, vol. 72, no. 5, p. 052 335, 2005.

[88] N. Rengaswamy, R. Calderbank, M. Newman, and H. D. Pfister, "On optimality of css codes for transversal t," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 2, pp. 499–514, 2020.

[89] J. Hu, Q. Liang, and R. Calderbank, "Designing the quantum channels induced by diagonal gates," *arXiv preprint arXiv:2109.13481*, 2021.

[90] J. T. Anderson, G. Duclos-Cianci, and D. Poulin, "Fault-tolerant conversion between the steane and reed-muller quantum codes," *Physical review letters*, vol. 113, no. 8, p. 080 501, 2014.

[91] H. Bombín, "Gauge color codes: Optimal transversal gates and gauge fixing in topological stabilizer codes," *New Journal of Physics*, vol. 17, no. 8, p. 083 002, 2015.

[92] F. H. Watson, E. T. Campbell, H. Anwar, and D. E. Browne, "Qudit color codes and gauge color codes in all spatial dimensions," *Physical Review A*, vol. 92, no. 2, p. 022 312, 2015.

[93] H. Bombín, "Dimensional jump in quantum error correction," *New Journal of Physics*, vol. 18, no. 4, p. 043 038, 2016.

[94] ——, "Transversal gates and error propagation in $3D$ topological codes," *arXiv preprint arXiv:1810.09575*, 2018.

[95] S. Bravyi and A. Cross, "Doubled color codes," *arXiv preprint arXiv:1509.03239*, 2015.

[96] C. Jones, P. Brooks, and J. Harrington, "Gauge color codes in two dimensions," *Physical Review A*, vol. 93, no. 5, p. 052 332, 2016.

[97] T. Jochym-O'Connor and S. D. Bartlett, "Stacked codes: Universal fault-tolerant quantum computation in a two-dimensional layout," *Physical Review A*, vol. 93, no. 2, p. 022 323, 2016.

[98] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal clifford gates and noisy ancillas," *Physical Review A*, vol. 71, no. 2, p. 022 316, 2005.

[99] E. T. Campbell and D. E. Browne, "On the structure of protocols for magic state distillation," in *Workshop on Quantum Computation, Communication, and Cryptography*, Springer, 2009, pp. 20–32.

[100] S. Bravyi and J. Haah, "Magic-state distillation with low overhead," *Physical Review A*, vol. 86, no. 5, p. 052 329, 2012.

[101] E. T. Campbell, H. Anwar, and D. E. Browne, "Magic-state distillation in all prime dimensions using quantum reed-muller codes," *Physical Review X*, vol. 2, no. 4, p. 041 021, 2012.

[102] T. Jochym-O'Connor, Y. Yu, B. Helou, and R. Laflamme, "The robustness of magic state distillation against errors in clifford gates," *arXiv preprint arXiv:1205.6715*, 2012.

[103] D. Litinski, "Magic state distillation: Not as costly as you think," *Quantum*, vol. 3, p. 205, 2019.

[104] C. Chamberland and A. W. Cross, "Fault-tolerant magic state preparation with flag qubits," *Quantum*, vol. 3, p. 143, 2019.

[105] C. Chamberland and K. Noh, "Very low overhead fault-tolerant magic state preparation using redundant ancilla encoding and flag qubits," *npj Quantum Information*, vol. 6, no. 1, pp. 1–12, 2020.

[106] M. Li, M. Gutiérrez, S. E. David, A. Hernandez, and K. R. Brown, "Fault tolerance with bare ancillary qubits for a [[7, 1, 3]] code," *Physical Review A*, vol. 96, no. 3, p. 032 341, 2017.

[107] D. Gottesman and L. L. Zhang, "Fibre bundle framework for unitary quantum fault tolerance," *arXiv preprint arXiv:1309.7062*, 2013.

[108] A. J. Landahl, J. T. Anderson, and P. R. Rice, "Fault-tolerant quantum computing with color codes," *arXiv preprint arXiv:1108.5738*, 2011.

[109] C. J. Trout, M. Li, M. Gutiérrez, Y. Wu, S.-T. Wang, L. Duan, and K. R. Brown, "Simulating the performance of a distance-3 surface code in a linear ion trap," *New Journal of Physics*, vol. 20, no. 4, p. 043 038, 2018.

[110] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *Journal of Mathematical Physics*, vol. 43, no. 9, pp. 4452–4505, 2002.

[111] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Physical Review A*, vol. 90, no. 3, p. 032 326, 2014.

[112] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the inherent intractability of certain coding problems (corresp.)," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[113] M.-H. Hsieh and F. Le Gall, "Np-hardness of decoding quantum error-correction codes," *Physical Review A*, vol. 83, no. 5, p. 052 331, 2011.

[114] K.-Y. Kuo and C.-C. Lu, "On the hardness of decoding quantum stabilizer codes under the depolarizing channel," in *2012 International Symposium on Information Theory and its Applications*, IEEE, 2012, pp. 208–211.

[115] ——, "On the hardnesses of several quantum decoding problems," *Quantum Information Processing*, vol. 19, no. 4, pp. 1–17, 2020.

[116] P. Iyer and D. Poulin, "Hardness of decoding quantum stabilizer codes," *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 5209–5223, 2015.

[117] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *arXiv preprint quant-ph/0504218*, 2005.

[118] P. Aliferis, "Level reduction and the quantum threshold theorem," *arXiv preprint quant-ph/0703230*, 2007.

[119] P. Aliferis, D. Gottesman, and J. Preskill, "Accuracy threshold for postselected quantum computation," *arXiv preprint quant-ph/0703264*, 2007.

[120] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error rate," *SIAM Journal on Computing*, 2008.

[121] K. Fujii, "Noise threshold of quantum supremacy," *arXiv preprint arXiv:1610.03632*, 2016.

[122] A. G. Fowler, A. C. Whiteside, and L. C. Hollenberg, "Towards practical classical processing for the surface code," *Physical review letters*, vol. 108, no. 18, p. 180 501, 2012.

[123] K. Fujii and Y. Tokunaga, "Error and loss tolerances of surface codes with general lattice structures," *Physical Review A*, vol. 86, no. 2, p. 020 303, 2012.

[124] A. G. Fowler, "Proof of finite surface code threshold for matching," *Physical review letters*, vol. 109, no. 18, p. 180 502, 2012.

[125] ——, "Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $o(1)$ parallel time," *arXiv preprint arXiv:1307.1740*, 2013.

[126] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, pp. 449–467, 1965.

[127] ——, "Maximum matching and a polyhedron with 0, 1-vertices," *Journal of research of the National Bureau of Standards B*, vol. 69, no. 125-130, pp. 55–56, 1965.

[128] V. Kolmogorov, "Blossom v: A new implementation of a minimum cost perfect matching algorithm," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 43–67, 2009.

[129] A. G. Fowler, "Optimal complexity correction of correlated errors in the surface code," *arXiv preprint arXiv:1310.0863*, 2013.

[130] D. S. Wang, A. G. Fowler, and L. C. Hollenberg, "Surface code quantum computing with error rates over 1%," *Physical Review A*, vol. 83, no. 2, p. 020 302, 2011.

[131] H. Bombin, G. Duclos-Cianci, and D. Poulin, "Universal topological phase of two-dimensional stabilizer codes," *New Journal of Physics*, vol. 14, no. 7, p. 073 048, 2012.

[132] A. M. Stephens, "Efficient fault-tolerant decoding of topological color codes," *arXiv preprint arXiv:1402.3037*, 2014.

[133] N. Delfosse, "Decoding color codes by projection onto surface codes," *Physical Review A*, vol. 89, no. 1, p. 012 317, 2014.

[134] A. B. Aloshious, A. N. Bhagoji, and P. K. Sarvepalli, "On the local equivalence of $2d$ color codes and surface codes with applications," *arXiv preprint arXiv:1804.00866*, 2018.

[135] A. B. Aloshious and P. K. Sarvepalli, "Projecting three-dimensional color codes onto three-dimensional toric codes," *Physical Review A*, vol. 98, no. 1, p. 012 302, 2018.

[136] ——, "Local equivalence of qudit color codes and toric codes," *Physical Review A*, vol. 100, no. 1, p. 012 348, 2019.

[137] A. Kubica and N. Delfosse, "Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders," *arXiv preprint arXiv:1905.07393*, 2019.

[138] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, "Triangular color codes on trivalent graphs with flag qubits," *New Journal of Physics*, vol. 22, no. 2, p. 023 019, 2020.

[139] J. P. Barnes, C. J. Trout, D. Lucarelli, and B. Clader, "Quantum error-correction failure distributions: Comparison of coherent and stochastic error models," *Physical Review A*, vol. 95, no. 6, p. 062 338, 2017.

[140] E. M. Rains, "Nonbinary quantum codes," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1827–1832, 1999.

[141] E. Sabo, A. B. Aloshious, and K. R. Brown, "Trellis decoding for qudit stabilizer codes and its application to qubit topological codes," *arXiv preprint arXiv:2106.08251*, 2021.

[142] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *arXiv preprint arXiv:1709.06218*, 2017.

[143] S. Huang, M. Newman, and K. R. Brown, "Fault-tolerant weighted union-find decoding on the toric code," *Physical Review A*, vol. 102, no. 1, p. 012 419, 2020.

[144] N. Delfosse and M. B. Hastings, "Union-find decoders for homological product codes," *Quantum*, vol. 5, p. 406, 2021.

[145] N. Delfosse, V. Londe, and M. Beverland, "Toward a union-find decoder for quantum ldpc codes," *arXiv preprint arXiv:2103.08049*, 2021.

[146] M. Herold, E. T. Campbell, J. Eisert, and M. J. Kastoryano, "Cellular-automaton decoders for topological quantum memories," *npj Quantum information*, vol. 1, no. 1, pp. 1–8, 2015.

[147] M. Herold, M. J. Kastoryano, E. T. Campbell, and J. Eisert, "Cellular automaton decoders of topological quantum memories in the fault tolerant setting," *New Journal of Physics*, vol. 19, no. 6, p. 063 012, 2017.

[148] A. Kubica and J. Preskill, "Cellular-automaton decoders with provable thresholds for topological codes," *Physical review letters*, vol. 123, no. 2, p. 020 501, 2019.

[149] G. Duclos-Cianci and D. Poulin, "Fast decoders for topological quantum codes," *Physical review letters*, vol. 104, no. 5, p. 050 504, 2010.

[150] ——, "A renormalization group decoding algorithm for topological quantum codes," in *2010 IEEE Information Theory Workshop*, IEEE, 2010, pp. 1–5.

[151] P. Sarvepalli and R. Raussendorf, "Efficient decoding of topological color codes," *Physical Review A*, vol. 85, no. 2, p. 022 317, 2012.

[152] D. J. MacKay, G. Mitchison, and P. L. McFadden, "Sparse-graph codes for quantum error correction," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2315–2330, 2004.

[153] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *arXiv preprint arXiv:0801.1241*, 2008.

[154] A. J. Ferris and D. Poulin, "Tensor networks and quantum error correction," *Physical review letters*, vol. 113, no. 3, p. 030 501, 2014.

[155] A. S. Darmawan and D. Poulin, "Tensor-network simulations of the surface code under realistic noise," *Physical review letters*, vol. 119, no. 4, p. 040 502, 2017.

[156] ——, "Linear-time general decoding algorithm for the surface code," *Physical Review E*, vol. 97, no. 5, p. 051 302, 2018.

[157] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, "Ultrahigh error threshold for surface codes with biased noise," *Physical review letters*, vol. 120, no. 5, p. 050 505, 2018.

[158] C. T. Chubb, "General tensor network decoding of 2d pauli codes," *arXiv preprint arXiv:2101.04125*, 2021.

[159] G. Torlai and R. G. Melko, "Neural decoder for topological codes," *Physical review letters*, vol. 119, no. 3, p. 030 501, 2017.

[160] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. Beenakker, "Machine-learning-assisted correction of correlated qubit errors in a topological code," *Quantum*, vol. 2, p. 48, 2018.

[161] C. Chamberland and P. Ronagh, "Deep neural decoders for near term fault-tolerant experiments," *Quantum Science and Technology*, vol. 3, no. 4, p. 044 002, 2018.

[162] P. Baireuther, M. D. Caio, B. Criger, C. W. Beenakker, and T. E. O'Brien, "Neural network decoder for topological color codes with circuit level noise," *New Journal of Physics*, vol. 21, no. 1, p. 013 003, 2019.

[163] C. Chinni, A. Kulkami, D. MPai, K. Mitra, and P. K. Sarvepalli, "Neural decoder for topological codes using pseudo-inverse of parity check matrix," in *2019 IEEE Information Theory Workshop (ITW)*, IEEE, 2019, pp. 1–5.

[164] N. Maskara, A. Kubica, and T. Jochym-O'Connor, "Advantages of versatile neural-network decoding for topological codes," *Physical Review A*, vol. 99, no. 5, p. 052 351, 2019.

[165] P. Panteleev and G. Kalachev, "Degenerate quantum ldpc codes with good finite length performance," *arXiv preprint arXiv:1904.02703*, 2019.

[166] J. Roffe, D. R. White, S. Burton, and E. T. Campbell, "Decoding across the quantum ldpc code landscape," *arXiv preprint arXiv:2005.07016*, 2020.

[167] H. Ollivier and J.-P. Tillich, "Trellises for stabilizer codes: Definition and uses," *Physical Review A*, vol. 74, no. 3, p. 032 304, 2006.

[168] D. Poulin, J.-P. Tillich, and H. Ollivier, "Quantum serial turbo codes," *IEEE Transactions on Information Theory*, vol. 55, no. 6, pp. 2776–2798, 2009.

[169] E. Pelchat and D. Poulin, "Degenerate viterbi decoding," *IEEE transactions on information theory*, vol. 59, no. 6, pp. 3915–3921, 2013.

[170] F. Xiao and H. Chen, "Construction of minimal trellises for quantum stabilizer codes," *Science China Information Sciences*, vol. 56, no. 1, pp. 1–11, 2013.

[171] M. Grassl, *Bounds on the minimum distance of linear codes and quantum codes*, Online available at http://www.codetables.de, 2007.

[172] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Transactions on information theory*, vol. 20, no. 2, pp. 284–287, 1974.

[173] J. Wolf, "Efficient maximum likelihood decoding of linear block codes using a trellis," *IEEE Transactions on Information Theory*, vol. 24, no. 1, pp. 76–80, 1978.

[174] D. J. Muder, "Minimal trellises for block codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1049–1053, 1988.

[175] R. J. McEliece, "On the bcjr trellis for linear block codes," *IEEE Transactions on Information Theory*, vol. 42, no. 4, pp. 1072–1092, 1996.

[176] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[177] G. D. Forney and M. D. Trott, "The dynamics of group codes: State spaces, trellis diagrams, and canonical encoders," *IEEE Transactions on Information Theory*, vol. 39, no. 5, pp. 1491–1513, 1993.

[178] G. D. Forney, "Dimension/length profiles and trellis complexity of linear block codes," *IEEE Transactions on information theory*, vol. 40, no. 6, pp. 1741–1752, 1994.

[179] F. R. Kschischang and V. Sorokine, "On the trellis structure of block codes," *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 1924–1937, 1995.

[180] R. J. McEliece, "The viterbi decoding complexity of linear block codes," in *Proceedings of 1994 IEEE International Symposium on Information Theory*, IEEE, 1994, p. 341.

[181] A. Vardy and F. R. Kschischang, "Proof of a conjecture of mceliece regarding the expansion index of the minimal trellis," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 2027–2033, 1996.

[182] A. B. Kiely, S. J. Dolinar, R. J. McEliece, L. L. Ekroot, and W. Lin, "Trellis decoding complexity of linear block codes," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1687–1697, 1996.

[183] R. Li, Z. Xu, and X. Li, "Standard forms of stabilizer and normalizer matrices for additive quantum codes," *IEEE transactions on information theory*, vol. 54, no. 8, pp. 3775–3778, 2008.

[184] M. M. Wilde, "Logical operators of quantum codes," *Physical Review A*, vol. 79, no. 6, p. 062 322, 2009.

[185] M. Grassl, "Variations on encoding circuits for stabilizer quantum codes," in *International Conference on Coding and Cryptology*, Springer, 2011, pp. 142–158.

[186] V. Sidorenko, G. Markarian, and B. Honary, "Minimal trellis design for linear codes based on the shannon product," *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 2048–2053, 1996.

[187] H. Kan and H. Shen, "Trellis properties of product codes," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 88, no. 1, pp. 353–358, 2005.

[188] J. Cannon, W. Bosma, C. Fieker, and A. Steel, "Handbook of mamga functions: Volume 13 - coding theory and cryptography," vol. 13, 2008.

[189] S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and trellis-based decoding algorithms for linear block codes*. Springer Science & Business Media, 2012, vol. 443.

[190] C. Wang, J. Harrington, and J. Preskill, "Confinement-higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory," *Annals of Physics*, vol. 303, no. 1, pp. 31–58, 2003.

[191] H. Bombín, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, "Strong resilience of topological codes to depolarization," *Physical Review X*, vol. 2, no. 2, p. 021 004, 2012.

[192] H. G. Katzgraber, H. Bombín, and M. Martin-Delgado, "Error threshold for color codes and random three-body ising models," *Physical review letters*, vol. 103, no. 9, p. 090 501, 2009.

[193] A. M. Stephens, "Fault-tolerant thresholds for quantum error correction with the surface code," *Physical Review A*, vol. 89, no. 2, p. 022 321, 2014.

[194] B. Rahn, A. C. Doherty, and H. Mabuchi, "Exact performance of concatenated quantum codes," *Physical Review A*, vol. 66, no. 3, p. 032 304, Sep. 2002.

[195] D. Poulin, "Optimal and efficient decoding of concatenated quantum block codes," *Phys. Rev. A*, vol. 74, p. 052 333, 5 Nov. 2006.

[196] A. Lafourcade and A. Vardy, "Optimal sectionalization of a trellis," *IEEE Transactions on Information Theory*, vol. 42, no. 3, pp. 689–703, 1996.

[197] D. Conti and N. Boston, "Matrix representations of trellises and enumerating trellis pseudocodewords," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2011, pp. 1438–1445.

[198] R. H. Hammack, W. Imrich, S. Klavžar, W. Imrich, and S. Klavžar, *Handbook of product graphs*. CRC press Boca Raton, 2011, vol. 2.

[199] R. Koetter and A. Vardy, "On the theory of linear trellises," in *Information, Coding and Mathematics*, Springer, 2002, pp. 323–354.

[200] C. Vuillot, L. Lao, B. Criger, C. G. Almudéver, K. Bertels, and B. M. Terhal, "Code deformation and lattice surgery are gauge fixing," *New Journal of Physics*, vol. 21, no. 3, p. 033 028, 2019.

[201] L. P. Pryadko, V. A. Shabashov, and V. K. Kozin, "Qdistrnd: A gap package for computing the distance of quantum error-correcting codes," *Journal of Open Source Software*, vol. 7, no. 71, p. 4120, 2022.

[202] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1757–1766, 1997.

[203] J. S. Leon, "A probabilistic algorithm for computing minimum weights of large error-correcting codes," *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1354–1359, 1988.

[204] M. Grassl, "Searching for linear codes with large minimum distance," in *Discovering mathematics with magma*, Springer, 2006, pp. 287–313.

[205] J. Cramwinckel, E. Roijackers, R. Baart, E. Minkes, L. Ruscio, R. L. Miller, T. Boothby, D. Joyner, and J. Fields, "A gap4 package for computing with error-correcting codes," 2008.

[206] T. S. D. Team, *Sage 9.5 reference manual: Coding theory - sagemath*, Online available at https://doc.sagemath.org/pdf/en/reference/coding/coding.pdf, 2022.

[207] G. White, "Enumeration-based algorithms in linear coding theory," 2006.

[208] K.-H. Zimmermann, "Integral hecke modules, integral generalized reed-muller codes, and linear codes," Techn. Univ. Hamburg-Harburg, Tech. Rep., 1996.

[209] A. Betten, H. Fripertinger, A. Kerber, A. Wassermann, and K.-H. Zimmermann, "Zyklische codes," in *Codierungstheorie*, Springer, 1998, pp. 77–174.

[210] P. Lisoněk and L. Trummer, "Algorithms for the minimum weight of linear codes," *Advances in Mathematics of Communications*, vol. 10, no. 1, p. 195, 2016.

[211] A. Canteaut, V. Lallemand, and M. Naya-Plasencia, "Related-key attack on full-round picaro," in *International Conference on Selected Areas in Cryptography*, Springer, 2015, pp. 86–101.

[212] S. Bouyuklieva and I. Bouyukliev, "An extension of the brouwer–zimmermann algorithm for calculating the minimum weight of a linear code," *Mathematics*, vol. 9, no. 19, p. 2354, 2021.

[213] F. Hernando, F. D. Igual, and G. Quintana-Ortí, "Algorithm 994: Fast implementations of the brouwer-zimmermann algorithm for the computation of the minimum distance of a random linear code," *ACM Transactions on Mathematical Software (TOMS)*, vol. 45, no. 2, pp. 1–28, 2019.

[214] G. Quintana-Ortí, F. Hernando, and F. D. Igual, "Parallel implementations for computing the minimum distance of a random linear code on multicomputers," *arXiv preprint arXiv:1911.08963*, 2019.

[215] G. White and M. Grassl, "A new minimum weight algorithm for additive codes," in *2006 IEEE International Symposium on Information Theory*, IEEE, 2006, pp. 1119–1123.

[216] J. Hu, Q. Liang, N. Rengaswamy, and R. Calderbank, "Mitigating coherent noise by balancing weight-2 z-stabilizers," *IEEE Transactions on Information Theory*, 2021.

[217] Y. Desaki, T. Fujiwara, and T. Kasami, "A method for computing the weight distribution of a block code by using its trellis diagram," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 77, no. 8, pp. 1230–1237, 1994.

[218] P. Shor and R. Laflamme, "Quantum analog of the macwilliams identities for classical coding theory," *Physical review letters*, vol. 78, no. 8, p. 1600, 1997.

[219] E. M. Rains, "Quantum weight enumerators," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1388–1394, 1998.

[220] E. Knill and R. Laflamme, "Quantum computing and quadratically signed weight enumerators," *Information Processing Letters*, vol. 79, no. 4, pp. 173–179, 2001.

[221] P. Rall, "Signed quantum weight enumerators characterize qubit magic state distillation," *arXiv preprint arXiv:1702.06990*, 2017.

[222] J. Hu, Q. Liang, and R. Calderbank, "Divisible codes for quantum computation," *arXiv preprint arXiv:2204.13176*, 2022.

[223] I. Bouyukliev and V. Bakoev, "A method for efficiently computing the number of codewords of fixed weights in linear codes," *Discrete applied mathematics*, vol. 156, no. 15, pp. 2986–3004, 2008.

[224] V. Senk, P. Radivojac, and I. Stanojevic, "An overview of decoding procedures for trellis codes," in *Proc. of TELFOR*, vol. 99, pp. 257–264.

[225] A. d. iOlius, J. E. Martinez, P. Fuentes, P. M. Crespo, and J. Garcia-Frias, "Performance of surface codes in realistic quantum hardware," *arXiv preprint arXiv:2203.15695*, 2022.

[226] A. Holmes, S. Johri, G. G. Guerreschi, J. S. Clarke, and A. Y. Matsuura, "Impact of qubit connectivity on quantum algorithm performance," *Quantum Science and Technology*, vol. 5, no. 2, p. 025 009, 2020.

[227] A. Vardy and Y. Be'Ery, "Bit-level soft-decision decoding of reed-solomon codes," *IEEE Transactions on Communications*, vol. 39, no. 3, pp. 440–444, 1991.

[228] T. R. Halford, V. Ponnampalam, A. J. Grant, and K. M. Chugg, "Soft-in soft-out decoding of reed-solomon codes based on vardy and be'ery's decomposition," *IEEE transactions on information theory*, vol. 51, no. 12, pp. 4363–4368, 2005.

[229] A. Vardy and Y. Be'ery, "Maximum-likelihood soft decision decoding of bch codes," *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 546–554, 1994.

[230] T. Kasami, S. Lin, and W. Peterson, "New generalizations of the reed-muller codes–i: Primitive codes," *IEEE Transactions on Information Theory*, vol. 14, no. 2, pp. 189–199, 1968.

[231] C. Chamberland and T. Jochym-O'Connor, "Error suppression via complementary gauge choices in reed-muller codes," *Quantum Science and Technology*, vol. 2, no. 3, p. 035 008, 2017.