Heuristics in quantum error correction

Alex Rigby

Bachelor of Science and Engineering (Honours), University of Tasmania, 2016

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy (Electronic Engineering)

December 2020

Declaration of originality

This thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis, and to the best of my knowledge and belief no material previously published or written by another person except where due acknowledgement is made in the text of the thesis, nor does the thesis contain any material that infringes copyright.

Authority of access

This thesis may be made available for loan and limited copying and communication in accordance with the Copyright Act 1968.

Statement regarding published work contained in the thesis

The publishers of the papers comprising Chapters 2 to 4 hold the copyright for that content, and access to the material should be sought from the respective journals. The remaining non published content of the thesis may be made available for loan and limited copying and communication in accordance with the Copyright Act 1968.

Signed: _____

Date: 18/12/2020

Statement of co-authorship

The following people and institutions contributed to the publication of work undertaken as part of this thesis:

Alex Rigby, School of Engineering = **Candidate**
JC Olivier, University of Tasmania = **Author 1**
Peter Jarvis, University of Tasmania = **Author 2**

Author details and their roles:

Paper 1, Modified belief propagation decoders for quantum low-density parity-check codes
Located in Chapter 2
Contributors: Candidate (85%), Author 1 (10%), and Author 2 (5%)
Candidate developed methods, gathered and interpreted data, and drafted the paper. Author 1 contributed to ideas, presentation and the development of methods. Author 2 contributed to ideas and presentation.

Paper 2, Optimizing short stabilizer codes for asymmetric channels
Located in Chapter 3
Contributors: Candidate (90%), Author 1 (5%), and Author 2 (5%)
Candidate developed methods, gathered and interpreted data, and drafted the paper. Authors 1 and 2 contributed to ideas and presentation.

Paper 3, Heuristic construction of codeword stabilized codes
Located in Chapter 4
Contributors: Candidate (90%), Author 1 (5%), and Author 2 (5%)
Candidate developed methods, gathered and interpreted data, and drafted the paper. Authors 1 and 2 contributed to ideas and presentation.

We the undersigned agree with the above stated "proportion of work undertaken" for each of the above published (or submitted) peer-reviewed manuscripts contributing to this thesis:

Signed: _____	Signed: _____	Signed: _____
JC Olivier	Andrew Chan	Alex Rigby
Supervisor	Head of School	Candidate
School of Engineering	School of Engineering	School of Engineering
University of Tasmania	University of Tasmania	University of Tasmania
Date: 18/12/2020	Date: 19th December 2020	Date: 18/12/2020

Acknowledgements

I would like to express my thanks to my primary supervisor JC Olivier for giving me the opportunity to study a topic that I am passionate about and for steering me in the right direction when required. I am also grateful to my secondary supervisor Peter Jarvis for being generous with his time and ensuring that my writing was always clear and concise.

I would like to thank my family, particularly Mum and Grandma, for their support and motivation over the last few years. Lastly, a special mention goes to Cassandra for giving me something to look forward to outside of work; it would have been a lonely time without you.

Table of contents

List of figures

List of tables

Abstract

Noise is a major obstacle in the development of practical schemes for quantum computation and communication. Similar to the case of classical communication, this noise can be protected against by employing a code, which provides a means for encoding quantum states prior to transmission and allows for errors to be inferred, and hopefully corrected, by a decoder at the receiver. Unfortunately, designing good codes and decoders is typically a difficult problem. This thesis focuses on developing low-complexity heuristic approaches to three such problems: the design of modified belief propagation decoders for quantum low-density parity-check codes, the design of stabilizer codes for asymmetric channels, and the design of codeword stabilized codes.

Quantum low-density parity-check codes are stabilizer codes with low-weight generators. Such codes permit low-complexity decoding via the use of belief propagation, which is an iterative message passing algorithm that takes place on a factor graph defined by the code. However, the performance of such a decoder is limited both by code structure and the degenerate nature of quantum errors. To overcome these limitations, at least in part, a number of modifications to belief propagation are developed. Central among these is the augmented decoder, which in the case of a decoding error, iteratively reattempts decoding using modified factor graph. This heuristic modification simply involves the duplication of a randomly selected subset of the graph's check nodes, which are in one-to-one correspondence with the code's stabilizer generators. Across a range of codes, it is shown that the decoders developed perform as well as or better than other modified decoders presented in literature.

For a number of channels of physical interest, phase-flip errors occur far more frequently than bit-flip errors. When transmitting across these so-called asymmetric channels, the decoding error rate can be minimized by tailoring the code used to the channel. However, assessing the performance of codes on a given channel is made difficult by the #P-completeness of optimal decoding. To address this complexity, it is shown that the decoding error rate can be accurately approximated using only a small fraction of the possible errors caused by the channel. This approximation is then used to identify a number of cyclic stabilizer codes that perform well on two different asymmetric channels. To further build on this, a heuristic is demonstrated for assessing code performance based on the decoding error rate of an associated classical code. The complexity of calculating this classical error rate is relatively low, and it is shown that it can be used as the basis for a hill-climbing search algorithm. Such searches have yielded a large number of highly performant codes satisfying various structure constraints.

The family of codeword stabilized codes encompasses both the stabilizer codes as well as many of the best known nonadditive codes. Constructing a standard form codeword stabilized code is a matter of selecting a simple undirected graph and a binary classical code. This makes designing optimal codes difficult as the number of possible graphs grows exponentially with code length, and the clique search required to construct the classical code is NP-hard. To address the exponential growth of the search space, a heuristic is developed for assessing graphs. This heuristic is then employed by a genetic algorithm that also makes use of a novel crossover operation based on spectral bisection, which is show to be superior to more standard crossover operations. With a graph selected, it is demonstrated that the complexity of the clique search required to construct the associated classical code can be mitigated through the use of a heuristic clique finding algorithm. A number of best known codes are presented that have been found using this approach.

Chapter 1

Introduction

1.1 Motivation

In quantum computation and communication, the fundamental unit of information is the qubit. At the physical level, a qubit is a two-level quantum system; simple examples include the spin of an electron or the polarization of a photon. In a number of ways, the nature of a qubit is quite different to that of a classical bit. For example, while the state of a classical bit is either 0 or 1, a qubit can be in a superposition (linear combination) of its two basis states $|0\rangle$ and $|1\rangle$. Furthermore, while the state of a pair of bits is one of 00, 01, 10, or 11, a system of two qubits can be in a superposition of the four basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. More generally, a system of n qubits can be in a superposition of 2^n basis states, and acting on such states lends an inherent parallelism to quantum computation. A number of quantum algorithms, such as those proposed by Shor for prime factorization and computing discrete logarithms [1, 2], exploit this parallelism to provide an exponential speedup compared to their classical analogs. Other algorithms, such as Grover's search algorithm [3], provide more modest, yet still useful, polynomial speedups.

Another fundamental difference between bits and qubits is the ability for two (or more) qubits to interact, potentially across a large physical distance, in such a way that it is not possible to describe the state of either individually. This is a phenomenon called entanglement, and it can be utilized in interesting and often surprising ways. One such example is superdense coding [4], which makes use of a preshared pair of entangled qubits to transmit the state of two bits in that of a single qubit. Conversely, quantum teleportation utilizes a preshared pair of entangled qubits to convey the arbitrary state of a qubit by sending only two classical bits [5]. Unfortunately, beyond interacting with each other, qubits tend to also interact with their environment, which causes the combined qubit-environment system to become entangled. This serves to corrupt the qubit's state and can be viewed as the action of a noisy channel across which the qubit is transmitted [6].

Being able to control the errors caused by noisy channels is key to achieving reliable quantum communication and computation. In the classical case, this can be done by employing a code,

which is a set of codewords to which data can be mapped, or encoded, prior to transmission. This adds redundancy that can be utilized to infer the transmitted codeword from the corrupted signal that is received. A particularly basic example is the three-bit repetition code, which as the name suggests, encodes 0 as 000 and 1 as 111. If any one of the three bits is flipped in transmission, it can be corrected at the receiver through a simple majority vote; for example, if the received signal is 010, which contains more zeros than ones, then it can be inferred that the transmitted codeword was 000. In the quantum case, such an encoding cannot be realized as it is physically impossible to duplicate arbitrary quantum states [7, 8, 9]. Instead, the state of k qubits is encoded in the highly entangled state of $n > k$ qubits, which serves to spread the information, protecting it from errors on a small subset of qubits [10]. In essence, employing a quantum code can be viewed as "fighting[ing] entanglement with entanglement" [11].

The focus of this thesis is on developing new methods for the design of quantum codes and their associated decoders. To begin, Sec. 1.2 gives a brief review of the mathematical foundations of quantum mechanics. This is built upon in Sec. 1.3 to give an introduction to noisy quantum channels. Section 1.4 then shows, largely through example, how quantum codes can be used to protect against such noise. For those interested, more exhaustive treatments of much of the content of Secs. 1.2 to 1.4 can be found in a number of well-known texts [6, 12, 13]. With the basics in hand, Sec. 1.5 outlines the structure of the remaining chapters.

1.2 Closed systems

1.2.1 States

A closed, or isolated, physical system is one that does not interact with its environment. Associated with each such system is a complex Hilbert space $\mathcal{H}$, and the system's state is described by a unit vector in $\mathcal{H}$. These state vectors are typically written in Dirac notation as "kets" $|\phi\rangle \in \mathcal{H}$ and are equivalent up to a global phase (that is, the vectors $|\phi\rangle$ and $e^{i\theta}|\phi\rangle$ correspond to the same state). All systems considered in this thesis are finite dimensional, in which case $\mathcal{H}$ is simply a complex inner product space.

In Dirac notation, the inner product $(|\phi\rangle, |\psi\rangle)$ of $|\phi\rangle, |\psi\rangle \in \mathcal{H}$ is typically written as $\langle\phi|\psi\rangle \equiv \langle\phi|(|\psi\rangle)$, where $\langle\phi|$ is a linear functional called a "bra." In adherence with standard physics convention, inner products in this thesis are taken to be linear in the second argument rather than the first. The inner product on $\mathcal{H}$ defines an outer product, which maps two states to a linear operator on $\mathcal{H}$. In particular, the outer product of $|\phi\rangle$ and $|\psi\rangle$ is the unique linear operator $|\phi\rangle\langle\psi|$ satisfying $(|\phi\rangle\langle\psi|)|\xi\rangle = |\phi\rangle\langle\psi|\xi\rangle = \langle\psi|\xi\rangle|\phi\rangle$ for all $|\xi\rangle \in \mathcal{H}$.

If $\mathcal{H}$ has dimension $\dim(\mathcal{H}) = q$, then an orthonormal basis is often written as $\mathcal{B} = \{|0\rangle, \ldots, |q-1\rangle\}$. If $|\phi\rangle \in \mathcal{H}$ is expressed in this basis as $|\phi\rangle = \sum_{i=0}^{q-1} \alpha_i |i\rangle$, then the normalization requirement $\langle\phi|\phi\rangle = 1$ becomes $\langle\phi|\phi\rangle = \sum_{i=0}^{q-1} |\alpha_i|^2 = 1$. Furthermore, if $|\psi\rangle = \sum_{i=0}^{q-1} \beta_i |i\rangle \in \mathcal{H}$, then $\langle\phi|\psi\rangle = \sum_{i=0}^{q-1} \alpha_i^* \beta_i$, which can be identified as the standard inner product on $\mathbb{C}^q$. Therefore,

if $|\phi\rangle$ is represented as a column vector in the basis $\mathcal{B}$ as $(\alpha_0, \ldots, \alpha_{q-1})^T$, then the bra $\langle\phi|$ is its Hermitian conjugate; that is, $\langle\phi| = |\phi\rangle^\dagger = (\alpha_0^*, \ldots, \alpha_{q-1}^*)$.

Of particular interest in this thesis, and in quantum computation and communication more generally, are qubits. As discussed in Sec. 1.1, a qubit is a two-level quantum system, meaning that it is described by a two-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^2$. It follows that a qubit's state can be written as

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1.1}$$

where $|\alpha|^2 + |\beta|^2 = 1$. This linear combination, or superposition, is in stark contrast to a classical bit, whose state is either 0 or 1. It can be convenient to use a basis other than $\{|0\rangle, |1\rangle\}$, which is often called the computational basis, to express the state of a qubit; for example, another commonly used orthonormal basis, called the Hadamard basis, consists of the states $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$.

1.2.2 Evolution

The time evolution of a closed quantum system is unitary. This means that if the state of a system at time t_1 is $|\phi\rangle \in \mathcal{H}$, then its state at time t_2 is $|\phi'\rangle = U|\phi\rangle$, where U is a unitary operator on $\mathcal{H}$ that depends only on t_1 and t_2. Such evolution is deterministic and preserves the normalization of states since

$$\langle\phi'|\phi'\rangle = (U|\phi\rangle)^\dagger U|\phi\rangle = (\langle\phi|U^\dagger)U|\phi\rangle = \langle\phi|U^\dagger U|\phi\rangle = \langle\phi|\phi\rangle = 1.$$

Furthermore, the guaranteed existence of the inverse $U^{-1} = U^\dagger$, which is itself unitary, means that time evolution is also reversible.

In the computational basis, the Pauli matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.2}$$

define an important set of unitary operators on qubits. Acting on the basis states, $X|0\rangle = |1\rangle$, $X|1\rangle = |0\rangle$, $Z|0\rangle = |0\rangle$, and $Z|1\rangle = -|1\rangle$; it therefore follows that X can be viewed as a bit-flip operator, Z as a phase-flip operator, and $Y = iXZ$ as a combination of the two. In the Hadamard basis, these interpretations are effectively reversed as $X|+\rangle = |+\rangle$, $X|-\rangle = -|-\rangle$, $Z|+\rangle = |-\rangle$, and $Z|-\rangle = |+\rangle$. This role reversal can also be seen using the Hadamard gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{1.3}$$

which is a unitary (and also Hermitian) operator mapping $|0\rangle \leftrightarrow |+\rangle$ and $|1\rangle \leftrightarrow |-\rangle$; the matrix representations of the bit-flip and phase-flip operators in the in the Hadamard basis are therefore $H^{-1}XH = HXH = Z$ and $HZH = X$, respectively.

1.2.3 Observables and measurement

An observable, which is a property of a system that can be measured, corresponds to Hermitian (self-adjoint) operator A that acts on the system's space $\mathcal{H}$. As A is Hermitian, it has real eigenvalues $\lambda_1, \ldots, \lambda_r$ corresponding to the mutually orthogonal eigenspaces $\mathcal{E}_1, \ldots, \mathcal{E}_r$, where $r \leq \dim(\mathcal{H})$. Furthermore, A can be written as $A = \sum_{i=1}^{r} \lambda_i P_i$, where P_i is a projector onto $\mathcal{E}_i$. If the system is initially in the state $|\phi\rangle$, then the outcome of measuring A will be the eigenvalue λ_i with probability $P(\lambda_i) = \langle\phi|P_i|\phi\rangle$. Furthermore, given an outcome λ_i, the state of the system after measurement will be

$$|\phi'\rangle = \frac{P_i|\phi\rangle}{\sqrt{P(\lambda_i)}} = \frac{P_i|\phi\rangle}{\sqrt{\langle\phi|P_i|\phi\rangle}}. \tag{1.4}$$

Unlike time evolution, this projection, or collapse, of states into an eigenspace of A is, in general, both nondeterministic and irreversible.

Returning to qubits, the Pauli matrices define valid observables as they are Hermitian. Furthermore, they each have two one-dimensional eigenspaces with associated eigenvalues $\lambda_1 = +1$ and $\lambda_2 = -1$. Taking Z as an example, suitable projectors onto these eigenspaces are $P_1 = |0\rangle\langle 0|$ and $P_2 = |1\rangle\langle 1|$, which gives $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$. It follows that making a measurement of Z on the state $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ will yield either the outcome $+1$ and post-measurement state $|\phi'\rangle = |0\rangle$ with probability $|\alpha|^2$ or the outcome -1 and post-measurement state $|1\rangle$ with probability $|\beta|^2$.

As the eigenspaces of A are mutually orthogonal, the projectors satisfy $P_i P_j = \delta_{ij} P_i$, where δ_{ij} is the Kronecker delta. It follows that remeasuring A is guaranteed to yield the same outcome λ_i as the initial measurement since

$$\langle\phi'|P_j|\phi'\rangle = \frac{\langle\phi|P_i P_j P_i|\phi\rangle}{\langle\phi|P_i|\phi\rangle} = \frac{\langle\phi|P_i \delta_{ij} P_i|\phi\rangle}{\langle\phi|P_i|\phi\rangle} = \frac{\langle\phi|P_i|\phi\rangle}{\langle\phi|P_i|\phi\rangle}\delta_{ij} = \delta_{ij}. \tag{1.5}$$

Suppose that after measuring A with outcome λ_i, a measurement of a different observable $B = \sum_j \tilde{\lambda}_j \tilde{P}_j$ is made. This can potentially project $|\phi'\rangle$ outside of the eigenspace $\mathcal{E}_i$, which means that subsequent measurements of A may yield an outcome $\lambda_j \neq \lambda_i$. This will not be the case if and only if A and B share a common eigenbasis, which is equivalent to requiring that they commute. Such observables are called compatible, and if A and B are compatible, then the order in which they are measured is unimportant: the probability of measuring A with outcome λ_i and then B with outcome $\tilde{\lambda}_j$ is the same as the probability of measuring B with outcome $\tilde{\lambda}_j$ and then A with outcome λ_i (the resulting states are also identical). Note that the Pauli matrices are not compatible observables as they anticommute with one another.

1.2.4 Multipartite states

Suppose that two individual systems, while potentially interacting with each other, do not interact with their greater environment. If these systems have associated spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, then the Hilbert space $\mathcal{H}$ for the combined system is given by their tensor product; that is, $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$. If the two systems are, in fact, not interacting and are in states $|\phi\rangle \in \mathcal{H}_1$

and $|\psi\rangle \in \mathcal{H}_2$, respectively, then the combined system will be in a so-called product state $|\phi\rangle_1 \otimes |\psi\rangle_2 \in \mathcal{H}$. Note that the tensor product and subscripts are often implied if there is no danger of ambiguity; that is, $|\phi\rangle_1 \otimes |\psi\rangle_2$ may be written as $|\phi\rangle_1|\psi\rangle_2$, $|\phi\rangle \otimes |\psi\rangle$, $|\phi\rangle|\psi\rangle$, or even $|\phi\psi\rangle$. In general, the state of the combined system will be a superposition $\sum_i \alpha_i|\phi_i\psi_i\rangle$, where $|\phi_i\rangle \in \mathcal{H}_1$ and $|\psi_i\rangle \in \mathcal{H}_2$. If this superposition is not itself a product state, then the state of each subsystem cannot be described individually, and the system is said to be entangled. A simple example of an entangled state is the two-qubit Bell state

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_1|0\rangle_2 + |1\rangle_1|1\rangle_2) \equiv \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{1.6}$$

An inner product on $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ is induced by those on $\mathcal{H}_1$ and $\mathcal{H}_2$. In particular,

$$(|\phi_i\psi_i\rangle, |\phi_j\psi_j\rangle) \equiv \langle\phi_i\psi_i|\phi_j\psi_j\rangle = \langle\phi_i|\phi_j\rangle\langle\psi_i|\psi_j\rangle, \tag{1.7}$$

where $|\phi_i\rangle, |\phi_j\rangle \in \mathcal{H}_1$ and $|\psi_i\rangle, |\psi_j\rangle \in \mathcal{H}_2$. It therefore follows that if $\mathcal{H}_1$ and $\mathcal{H}_2$ have orthonormal bases $\mathcal{B}_1 = \{|\phi_1\rangle, \ldots, |\phi_p\rangle\}$ and $\mathcal{B}_2 = \{|\psi_1\rangle, \ldots, |\psi_q\rangle\}$, respectively, then $\mathcal{B} = \{|\phi_1\psi_1\rangle, \ldots, |\phi_1\psi_q\rangle, \ldots\ldots, |\phi_p\psi_1\rangle, \ldots, |\phi_p\psi_q\rangle\}$ is an orthonormal basis for $\mathcal{H}$, meaning that $\dim(\mathcal{H}) = \dim(\mathcal{H}_1)\dim(\mathcal{H}_2)$. Furthermore, the matrix (column vector) representation in the basis $\mathcal{B}$ of some product state $|\phi\psi\rangle \in \mathcal{H}$ can be found by taking the Kronecker product of the representations of $|\phi\rangle$ and $|\psi\rangle$ in $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively. Note that the Kronecker product of two matrices $A = (a_{ij})$ and $B = (b_{ij})$ is

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}. \tag{1.8}$$

As was the case for the inner product, operators on $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ are defined in terms of those on $\mathcal{H}_1$ and $\mathcal{H}_2$. In particular, if A and B are operators on $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, then $A_1 \otimes B_2 \equiv A \otimes B$ is an operator on $\mathcal{H}$ satisfying

$$(A \otimes B)(|\phi\rangle \otimes |\psi\rangle) = (A|\phi\rangle) \otimes (B|\psi\rangle) \tag{1.9}$$

for all $|\phi\rangle \in \mathcal{H}_1$ and $|\psi\rangle \in \mathcal{H}_2$. It follows directly from Eq. (1.9) that composition of operators is component-wise [that is, $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$] and that the Hermitian conjugate distributes across the tensor product [that is, $(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger$]. As a result, if A and B are Hermitian and/or unitary, then so is $A \otimes B$. In the context of qubits, this means that tensor products of Pauli and identity operators define both time evolution operators and observables. These identity components and the tensor products themselves are often implied; for example, $A_1 \otimes I_2 \equiv A \otimes I$ may also be written as $A_1 I_2$ or A_1 if the meaning is clear. The number of nonidentity components from which an operator is comprised is called its weight. Usefully, with the basis selection outlined in the previous paragraph, the tensor product of operators also corresponds to the Kronecker product.

In general, an operator on $\mathcal{H}$ can be a linear combination $\sum_i A^{(i)} \otimes B^{(i)}$ of tensor products, where $A^{(i)}$ and $B^{(i)}$ are operators on $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively. An example of such an operator

on two qubits is the controlled NOT (CNOT) gate

$$\mathrm{CNOT}_{12} = |0\rangle_1\langle 0|_1 \otimes I_2 + |1\rangle_1\langle 1|_1 \otimes X_2 \equiv |0\rangle_1\langle 0|_1 + |1\rangle_1\langle 1|_1 X_2, \tag{1.10}$$

which maps $|00\rangle \to |00\rangle$, $|01\rangle \to |01\rangle$, $|10\rangle \to |11\rangle$, and $|11\rangle \to |10\rangle$; that is, it applies a bit flip to the second qubit if the first is in the state $|1\rangle$.

While the discussion here has been in the context of collections of two systems, it extends inductively to any number of systems. For example, if there are three systems with spaces $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$, then $\mathcal{H} = (\mathcal{H}_1 \otimes \mathcal{H}_2) \otimes \mathcal{H}_3 = \mathcal{H}_1 \otimes (\mathcal{H}_2 \otimes \mathcal{H}_3) = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3$ is the Hilbert space of the combined system, which has dimension $\dim(\mathcal{H}_1)\dim(\mathcal{H}_2)\dim(\mathcal{H}_3)$. Furthermore, operators on $\mathcal{H}$ will be of the form $\sum_i A^{(i)} \otimes B^{(i)} \otimes C^{(i)}$, where $A^{(i)}$, $B^{(i)}$, and $C^{(i)}$ are operators on $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$, respectively. More generally, if there are n systems with spaces $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_n$, then $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n$, $\dim(\mathcal{H}) = \prod_i \dim(\mathcal{H}_i)$, and operators will be linear combinations of n-fold tensor products. If the constituent systems are identical with $\mathcal{H}_1 = \mathcal{H}_2 = \cdots = \mathcal{H}_n$, then this space can be expressed more compactly as $\mathcal{H} = \mathcal{H}_1^{\otimes n}$. Using this notation, the 2^n-dimensional space of a system of n qubits is $\mathcal{H} \cong (\mathbb{C}^2)^{\otimes n} \cong \mathbb{C}^{2^n}$.

1.3 Open systems

1.3.1 Density operators

As the name suggests, an open system is one that interacts with its environment. The behavior of such systems can be considered using the density operator formalism, which is a framework for describing statistical ensembles of states. Suppose that a closed system with space $\mathcal{H}$ is in the state $|\phi_i\rangle$ with probability p_i; this ensemble of states gives the density operator

$$\rho = \sum_i p_i |\phi_i\rangle\langle\phi_i|. \tag{1.11}$$

It follows from this definition that ρ is a positive operator on $\mathcal{H}$ with trace one. If the system's state is known with certainty to be $|\phi\rangle \in \mathcal{H}$, then the density operator is a so-called pure state $\rho = |\phi\rangle\langle\phi|$ (note that state vectors themselves are often also referred to as pure states). If ρ is not of this form, then it is called a mixed state; it describes a system about which there is incomplete information. Given a density operator, a simple test for purity is to calculate $\mathrm{tr}(\rho^2)$, with a state being pure if and only if $\mathrm{tr}(\rho^2) = 1$. The density operator formalism extends naturally to ensembles of mixed states; if a system is in state $\rho^{(i)} = \sum_j p_{ij}|\phi_{ij}\rangle\langle\phi_{ij}|$ with probability p_i, then the density operator is

$$\rho = \sum_{ij} p_i p_{ij} |\phi_{ij}\rangle\langle\phi_{ij}| = \sum_i p_i \sum_j p_{ij} |\phi_{ij}\rangle\langle\phi_{ij}| = \sum_i p_i \rho^{(i)}. \tag{1.12}$$

The unitary evolution of a system in state ρ follows directly from the evolution of the constituent pure states, with

$$\rho = \sum_i p_i |\phi_i\rangle\langle\phi_i| \to \sum_i p_i (U|\phi_i\rangle)(U|\phi_i\rangle)^\dagger = \sum_i p_i U|\phi_i\rangle\langle\phi_i|U^\dagger = U\rho U^\dagger. \tag{1.13}$$

A similar approach shows that measuring the observable $A = \sum_i \lambda_i P_i$ on a system in state ρ will yield the outcome λ_i with probability

$$P(\lambda_i) = \mathrm{tr}(P_i \rho), \tag{1.14}$$

and the state of the system after measurement will be $\rho' = P_i \rho P_i / P(\lambda_i)$. While it is possible for multiple ensembles to yield the same density operator, it follows from Eq. (1.14) depending only on ρ that there is no measurement capable of distinguishing between them; they are, in this sense, equivalent ensembles.

1.3.2 Multipartite states

If two noninteracting systems with spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ are in the states $\rho_1 = \sum_i p_i |\phi_i\rangle\langle\phi_i|$ and $\rho_2 = \sum_j q_j |\psi_j\rangle\langle\psi_j|$, respectively, then it follows that the density operator for the combined system is simply

$$\rho = \sum_{ij} p_i q_j (|\phi_i \psi_j\rangle)(|\phi_i \psi_j\rangle)^\dagger = \sum_{ij} p_i q_j |\phi_i\rangle\langle\phi_i| \otimes |\psi_j\rangle\langle\psi_j| = \rho_1 \otimes \rho_2. \tag{1.15}$$

Again, this is referred to as a product state. If the state of the combined system can be written as a convex combination $\rho = \sum_i p_i \rho_1^{(i)} \otimes \rho_2^{(i)}$ of product states, then it is called separable. Alternatively, if the system's state cannot be represented in such a way, then it is entangled.

A useful tool in considering subsystems of larger systems is the reduced density operator, which completely defines the statistics of local measurements made on the subsystem. If a system with space $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ is in state ρ, then the reduced density operator for the first system is $\rho_1 = \mathrm{tr}_2(\rho)$, where tr_2 is the partial trace operator over the second system; this is the unique linear operator satisfying

$$\mathrm{tr}_2(|\phi_i\rangle\langle\phi_j| \otimes |\psi_i\rangle\langle\psi_j|) = |\phi_i\rangle\langle\phi_j| \mathrm{tr}(|\psi_i\rangle\langle\psi_j|) = |\phi_i\rangle\langle\phi_j| \langle\psi_j|\psi_i\rangle, \tag{1.16}$$

where $|\phi_i\rangle, |\phi_j\rangle \in \mathcal{H}_1$ and $|\psi_i\rangle, |\psi_j\rangle \in \mathcal{H}_2$. If $\{|0\rangle, \ldots, |q-1\rangle\}$ is a basis for $\mathcal{H}_2$, then a useful way to express the partial trace is as

$$\mathrm{tr}_2(\rho) = \sum_i (I \otimes \langle i|) \rho (I \otimes |i\rangle). \tag{1.17}$$

If the combined system is in the separable state $\rho = \sum_i p_i \rho_1^{(i)} \otimes \rho_2^{(i)}$, then $\rho_1 = \mathrm{tr}_2(\rho) = \sum_i p_i \rho_1^{(i)}$. Things are somewhat more interesting when the combined system is entangled. For example, the reduced density operator for the first qubit in the Bell state of Eq. (1.6) is

$$\begin{aligned}
\rho_1 &= \mathrm{tr}_2(|\Psi^+\rangle\langle\Psi^+|) \\
&= \frac{1}{2} \sum_i (I \otimes \langle i|)(|00\rangle + |11\rangle)(\langle 00| + \langle 11|)(I \otimes |i\rangle) \\
&= \frac{1}{2} \sum_i (|0\rangle\langle i|0\rangle + |1\rangle\langle i|1\rangle)(\langle 0|\langle 0|i\rangle + \langle 1|\langle 1|i\rangle) \\
&= \frac{1}{2} (|0\rangle\langle 0| + |1\rangle\langle 1|) \\
&= \frac{1}{2} I.
\end{aligned}$$

As $\mathrm{tr}(\rho_1^2) = 1/2 < 1$, this means that while the combined two-qubit system is in a pure state, the first qubit is in a mixed state (by symmetry, the second qubit is also in the same mixed state). This reflects the fact that for an entangled system, the state of a subsystem cannot be known exactly in isolation. This logic also runs in the other direction: any system in a mixed state can be viewed as part of a larger system that is in a pure entangled state.

1.3.3 Channels

As noted in Sec. 1.1, noise in quantum systems results from becoming entangled with the environment. Suppose that a system is initially in state ρ and not entangled with its environment, which is in state ρ_{env}. Without loss of generality, by considering a large enough environmental system, ρ_{env} can be taken to be a pure state $\rho_{\mathrm{env}} = |e_0\rangle\langle e_0|$. If the combined system evolves according to the unitary U, then its state becomes

$$\rho \otimes |e_0\rangle\langle e_0| \to U(\rho \otimes |e_0\rangle\langle e_0|)U^\dagger. \tag{1.18}$$

In general, this is an entangled state, and the reduced density operator for the system of interest is

$$\Phi(\rho) = \mathrm{tr}_{\mathrm{env}}[U(\rho \otimes |e_0\rangle\langle e_0|)U^\dagger] = \sum_k (I \otimes \langle e_k|)U(\rho \otimes |e_0\rangle\langle e_0|)U^\dagger(I \otimes |e_k\rangle), \tag{1.19}$$

where the states $|e_k\rangle$ form a basis for the environment. With a bit of manipulation, it can be shown that $\rho \otimes |e_0\rangle\langle e_0| = (I \otimes |e_0\rangle)\rho(I \otimes \langle e_0|)$, which reduces Eq. (1.19) to

$$\Phi(\rho) = \sum_k A_k \rho A_k^\dagger, \tag{1.20}$$

where the $A_k = (I \otimes \langle e_k|)U(I \otimes |e_0\rangle)$ are operators on the system of interest. To preserve the trace of ρ, these so-called Kraus operators must satisfy $\sum_k A_k^\dagger A_k = I$ [14]. Unlike that of the system-environment combination, the evolution described by (1.20) is generally nonunitary and hence irreversible; the state of the system can be viewed as having been corrupted by noise.

A useful stochastic interpretation of Eq. (1.20) is of Φ as a noisy channel that maps its input ρ to an output $A_k \rho A_k^\dagger / \mathrm{tr}(A_k \rho A_k^\dagger)$ with probability $\mathrm{tr}(A_k \rho A_k^\dagger)$ [6]. All channels considered in this thesis act on qubits; furthermore, the majority of them are of the form

$$\Phi(\rho) = p_I \rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z. \tag{1.21}$$

Due to the Kraus operators being $\sqrt{p_\sigma}\sigma$ for $\sigma \in \{I, X, Y, Z\}$, these are called Pauli channels. It follows from the cyclicity of the trace operator that

$$\mathrm{tr}(\sqrt{p_\sigma}\sigma\rho\sqrt{p_\sigma}\sigma^\dagger) = \mathrm{tr}(p_\sigma \sigma^\dagger\sigma\rho) = \mathrm{tr}(p_\sigma\rho) = p_\sigma. \tag{1.22}$$

This means that irrespective of the input state, a Pauli channel can be viewed as applying the error operator σ with probability p_σ, which serves to map $\rho \to \sigma\rho\sigma^\dagger = \sigma\rho\sigma$. If consideration is restricted to pure inputs $\rho = |\phi\rangle\langle\phi|$, then this interpretation reduces to a mapping of $|\phi\rangle \to \sigma|\phi\rangle$ with probability p_σ. A particularly simple Pauli channel is the bit-flip channel, which applies either a bit-flip error with probability p or leaves the input unchanged with probability $1 - p$

(that is, $p_X = p$, $p_Y = p_Z = 0$, and $p_I = 1-p$). Perhaps the most commonly considered channel, Pauli or otherwise, is the depolarizing channel, which applies an X, Y, or Z error with equal probability (that is, $p_X = p_Y = p_Z = p/3$ and $p_I = 1-p$). Also considered in this thesis are non-Pauli channels related to the amplitude damping channel, which has the Kraus operators

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, A_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}. \tag{1.23}$$

1.4 Quantum codes

1.4.1 A three-qubit code

Suppose that the arbitrary state of a qubit is to be transmitted across a quantum bit-flip channel with $p < 1/2$. As alluded to in Sec. 1.1, the errors caused by the channel cannot be protected against by simply mimicking the approach of the classical three-bit repetition code. This is a result of the no-cloning theorem, which asserts that there is no physically realizable operation that duplicates an arbitrary quantum state [7, 8, 9]. This is straightforward to prove: Duplicating an arbitrary state $|\psi\rangle \in \mathcal{H}$ would require a unitary operator U that maps $|\psi\xi\rangle \to |\psi\psi\rangle$ for some ancilla state $|\xi\rangle \in \mathcal{H}$. Supposing that such a unitary exists, it must be able to duplicate the distinct nonorthogonal states $|\psi_1\rangle, |\psi_2\rangle \in \mathcal{H}$; that is, it must be the case that $U|\psi_1\xi\rangle = |\psi_1\psi_1\rangle$ and $U|\psi_2\xi\rangle = |\psi_2\psi_2\rangle$. However, this gives

$$|\langle\psi_1|\psi_2\rangle| = |\langle\psi_1|\psi_2\rangle\langle\xi|\xi\rangle| = |\langle\psi_1\xi|\psi_2\xi\rangle| = |\langle\psi_1\xi|U^\dagger U|\psi_2\xi\rangle| = |\langle\psi_1\psi_1|\psi_2\psi_2\rangle| = |\langle\psi_1|\psi_2\rangle|^2, \tag{1.24}$$

which is contradiction as $0 < |\langle\psi_1|\psi_2\rangle| < 1$.

While the no-cloning theorem prohibits encoding of the arbitrary qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \in \mathbb{C}^2$ as $|\psi\psi\psi\rangle \in (\mathbb{C}^2)^{\otimes 3}$, it is possible to prepare the state $|\psi 00\rangle \in (\mathbb{C}^2)^{\otimes 3}$ and map it to

$$|\phi\rangle = \mathrm{CNOT}_{12}\mathrm{CNOT}_{13}|\psi 00\rangle = \alpha|000\rangle + \beta|111\rangle = \alpha|0_L\rangle + \beta|1_L\rangle. \tag{1.25}$$

As in the classical case, this encoded state is called a codeword. The set of all codewords forms a subspace $\mathcal{Q} \subset (\mathbb{C}^2)^{\otimes 3}$, called a code, with the basis $\{|0_L\rangle = |000\rangle, |1_L\rangle = |111\rangle\}$; $\mathcal{Q}$ can therefore be viewed as encoding the state of a single "logical" qubit in that of three "physical" qubits. Codewords are transmitted across a combination of three independent bit-flip channels, one for each physical qubit. This combined channel serves to map $|\phi\rangle \to E|\phi\rangle$, where the error E is a three-fold tensor product of identity and/or bit-flip operators. Furthermore, the probability of E occurring is simply the product of the component probabilities; for example, the error $E = X \otimes I \otimes I \equiv X_1$ occurs with probability $p_X p_I^2 = p(1-p)^2$. The eight possible errors and their associated probabilities can be found in Table 1.1.

As for encoding, care needs to be taken in designing a scheme for decoding, which is the process of inferring the error E caused by the channel. For the classical three-bit repetition code, the location of a bit-flip error can be determined indirectly by inferring the transmitted codeword and comparing it to the channel output. For example, as was detailed in Sec. 1.1, if the

Table 1.1: Possible errors caused by the three-qubit bit-flip channel. Also given are their associated probabilities and measurement outcomes for $M_1 = Z_1Z_2$ and $M_2 = Z_2Z_3$.

E	$P(E)$	M_1 outcome	M_2 outcome
I	$(1-p)^3$	$+1$	$+1$
X_1	$p(1-p)^2$	-1	$+1$
X_2	$p(1-p)^2$	-1	-1
X_3	$p(1-p)^2$	$+1$	-1
X_1X_2	$p^2(1-p)$	$+1$	-1
X_1X_3	$p^2(1-p)$	-1	-1
X_2X_3	$p^2(1-p)$	-1	$+1$
$X_1X_2X_3$	p^3	$+1$	$+1$

channel output is 010, then it follows that the transmitted codeword was 000, meaning that there was an error on the second bit. Such an approach is unsuitable in the quantum case as performing a measurement of the channel output $E|\phi\rangle$ that yields information about the transmitted codeword $|\phi\rangle$ will, in general, irreversibly alter it. Measurements must therefore be made that reveal something about E but not about $|\phi\rangle$. This can be achieved by selecting a set of compatible Pauli observables $\{M_1, \ldots, M_m\}$ for which the codewords are $+1$ eigenstates. Making a measurement of M_i on the channel output $E|\phi\rangle$ will then reveal only whether M_i and E commute or anticommute: If they commute, then $M_iE|\phi\rangle = EM_i|\phi\rangle = E|\phi\rangle$, meaning the outcome will be $+1$. Otherwise, if they anticommute, then $M_iE|\phi\rangle = -EM_i|\phi\rangle = -E|\phi\rangle$, meaning the outcome will be -1.

In this instance, $M_1 = Z_1Z_2$ and $M_2 = Z_2Z_3$ are appropriate observables as they commute with each other and $M_1|\phi\rangle = M_2|\phi\rangle = |\phi\rangle$ for all $|\phi\rangle \in \mathcal{Q}$. The outcomes of measuring these observables for the eight possible errors are given in Table 1.1 (note that M_i and E will commute if their nonidentity components differ in an even number of positions and will anticommute otherwise). It can be seen that there are two errors that cause each pair of measurement outcomes; for example, the errors X_1 and X_2X_3 both result in outcomes of -1 and $+1$ for the measurements of M_1 and M_2, respectively. However, it is reasonable to infer that the highest-probability error $\hat{E}$ associated with a given pair of outcomes has occurred. For $p < 1/2$, this means that $\hat{E}$ will be one of I, X_1, X_2, or X_3.

With the most likely error inferred, correction can be attempted by applying $\hat{E}$ to the channel output to give $\hat{E}E|\phi\rangle$. As all Pauli matrices square to the identity (they are Hermitian and unitary), this will return the codeword $|\phi\rangle$ if $\hat{E} = E$; however, if $\hat{E} \neq E$, then $\hat{E}E|\phi\rangle \neq |\phi\rangle$ in general, in which case a decoding error is said to have occurred. As this procedure will correct any error on one or fewer qubits, the probability of an error occurring that cannot be corrected, called the decoding error rate, is $1-(1-p)^3-3p(1-p)^2 = 3p^2-2p^3$. For $p < 1/2$, $3p^2-2p^3 < p$, which means that employing this code allows the qubit's state to be transmitted more reliably than if it were sent unencoded.

The procedure described here can, in fact, correct more than just the single-qubit bit-flip errors X_1, X_2, and X_3. For example, suppose the error $E = \alpha_0 I + \alpha_1 X_1 + \alpha_2 X_2 + \alpha_3 X_3$ occurs, where $\sum_i |\alpha_i|^2 = 1$. Depending on the outcomes, measuring M_1 and M_2 will project $E|\phi\rangle$

into one of the four intersections of an eigenspace of M_1 with another of M_2. As $I|\phi\rangle$, $X_1|\phi\rangle$, $X_2|\phi\rangle$, and $X_3|\phi\rangle$ each belong to different intersections (see Table 1.1), this effectively discretizes E, mapping $E|\phi\rangle \to E'|\phi\rangle$, where $E' = I$ with probability $|\alpha_0|^2$ or X_i with probability $|\alpha_i|^2$. Furthermore, the value of E' can be inferred based on the measurement outcomes in exactly the same way as previously outlined. In general, the discretization, or digitization, of errors afforded by measurement means that any superposition of correctable errors is itself a correctable error. Conversely, it means that a continuum of errors can be protected against by designing a scheme that corrects a basis for such errors.

1.4.2 The Shor code

By using the fact that Z errors act as a bit flip in the Hadamard basis, the three-qubit code of Sec. 1.4.1 can be modified to instead protect against phase-flip errors. In particular, altering the encoding of Eq. (1.25) to

$$|\phi\rangle = H_1H_2H_3\text{CNOT}_{12}\text{CNOT}_{13}|\psi 00\rangle = \alpha|{+}{+}{+}\rangle + \beta|{-}{-}{-}\rangle = \alpha|0_L\rangle + \beta|1_L\rangle, \tag{1.26}$$

and selecting the observables $M_1 = X_1X_2$ and $M_2 = X_2X_3$ allows any single-qubit phase-flip error to be corrected.

The Shor code [10] encodes the state $|\psi\rangle$ of one qubit in that of nine by combining the three-qubit bit-flip and phase-flip codes. First, $|\psi\rangle$ is encoded using the phase-flip code according to Eq. (1.26). Each of the three physical qubits are then further encoded using the bit-flip code according to Eq. (1.25). The combination of these two encodings maps $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \to \alpha|0_L\rangle + \beta|1_L\rangle$, where the basis codewords are

$$|0_L\rangle = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle), \tag{1.27}$$

$$|1_L\rangle = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle). \tag{1.28}$$

As for the three-qubit bit-flip code, measuring $M_1 = Z_1Z_2$ and $M_2 = Z_2Z_3$ will allow a bit flip in the first block of three qubits to be located. Similarly, if a bit flip occurs in the second block, then measuring $M_3 = Z_4Z_5$ and $M_4 = Z_5Z_6$ will locate it, and if it occurs in the third block, then measuring $M_5 = Z_7Z_8$ and $M_6 = Z_8Z_9$ will locate it. Again, with a bit flip located, it can be corrected by applying another bit flip to the appropriate qubit. Correcting phase flips is somewhat more interesting as the Shor code exhibits degeneracy. This is a property of quantum codes whereby distinct correctable errors have the same effect on the code space. In this instance, it can be seen that any two phase-flip errors on qubits within the same three-qubit block will act identically on the code. For example, the errors Z_1, Z_2, and Z_3 all map the basis codewords to

$$Z_1|0_L\rangle = Z_2|0_L\rangle = Z_3|0_L\rangle = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle), \tag{1.29}$$

$$Z_1|1_L\rangle = Z_2|0_L\rangle = Z_3|1_L\rangle = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle). \tag{1.30}$$

As a result of this degeneracy, it is sufficient to infer only which block a phase-flip error has occurred in, which can be achieved by making the measurements $M_7 = X_1X_2X_3X_4X_5X_6$ and $M_8 = X_4X_5X_6X_7X_8X_9$. Once the appropriate block has been located, applying a phase flip to any of the qubits in it will correct the error.

As the observables selected are compatible, and both M_7 and M_8 commute with any X_i, the process of correcting a bit-flip error has no effect on the correction of a phase-flip error. This is the case even if a bit flip and phase flip occur on the same qubit, meaning that Shor code also allows for any single-qubit $Y \propto XZ$ error to be corrected. As the Pauli matrices along with the identity form a basis for the complex vector space of 2×2 matrices, it therefore follows from the discussion of Sec. 1.4.1 that the Shor code can, in fact, correct any arbitrary error on a single qubit. Beyond this, it can also detect any error on two or fewer qubits, which means that any such error will either act trivially on the code or yield at least one measurement outcome of -1. However, there are three-qubit errors, such as $X_1X_2X_3$, that have a nontrivial effect on the code but commute with every observable and hence cannot be detected. As a result of this, the Shor code is said to have distance $d = 3$.

1.4.3 General codes

The codes presented in Secs. 1.4.1 and 1.4.2 are examples of stabilizer codes [15]. The stabilizer itself is the Abelian group generated by the observables, which must be n-fold tensor products of Pauli and identity operators. The code, which is said to be of length n, is simply the space of n-qubit states that are $+1$ eigenstates of every element of the stabilizer. As the codes presented each had $m = n - 1$ stabilizer generators, they encoded the space of a single qubit; more generally, a code whose stabilizer has m generators will encode the space of $k = n - m$ qubits. As was discussed for the Shor code in Sec. 1.4.2, a code is said to have distance d if it can detect any error on $d - 1$ or fewer qubits but not some error on d qubits. A length-n stabilizer code that encodes the state of k qubits and has distance d is called an $[[n, k]]$ or $[[n, k, d]]$ code; for example, the Shor code is a $[[9, 1, 3]]$ code.

Stabilizer codes can be designed with a number of different objectives in mind. At its simplest, at least conceptually, the aim when designing an $[[n, k]]$ code for a given channel is to minimize the decoding error rate. However, such an approach is limited by the #P-completeness of optimal stabilizer code decoding [16], which makes determining a code's error rate computationally impractical for even moderate code lengths. For the depolarizing channel, this complexity can be avoided in part by instead designing for large distance, which serves as something of a proxy for the decoding error rate. Alternatively, rather than fixing the number of encoded qubits k, it is also possible to construct codes of fixed length n and distance d with the aim of maximizing k. More generally, this distance criterion can be extended to requiring that a set $\mathcal{E}$ of errors is detected.

The majority of known code families fall within the stabilizer framework that has been the focus of this section. However, it is possible for codes outside of this framework, called nonadditive

codes, to encode a larger subspace while still having the same distance or detecting the same set of errors [17, 18, 19, 20, 21, 22]. Unlike stabilizer codes, this subspace is not necessarily that of k qubits; that is, the dimension K of a nonadditive code need not be a power of two. This leads to a slight difference in the notation used to describe them: a potentially nonadditive code of length n, dimension K, and distance d is called an $((n, K))$ or $((n, K, d))$ code.

1.5 Thesis outline

The body of this thesis is comprised of three papers on the design of quantum codes and their decoders. These papers are largely self-contained, providing reviews of relevant theory and literature, which does come at the cost of some amount of unavoidable repetition; however, efforts have been made to keep this to a minimum. There are a number of common themes that run through the papers. Chief among them, as suggested by the title of the thesis, is the heuristic nature of the methods employed. These heuristic methods (or simply heuristics) are approaches that can quickly yield good, if not optimal, solutions to hard problems.

Quantum low-density parity-check (QLDPC) codes are stabilizer codes that have low-weight generators [23]. Similar to classical low-density parity-check (LDPC) codes, decoding for QLDPC codes can be performed using belief propagation, which is a heuristic message passing algorithm that takes place on a bipartite graph, called a factor graph, defined by the code's generators. Unfortunately, the commuting nature of these generators results in unavoidable cycles of length four in the factor graph, which are detrimental to the performance (decoding error rate) of belief propagation [24]. This performance is further degraded by the degenerate nature of quantum errors, which is not accounted for in the component-wise inference of a belief propagation decoder [25]. Chapter 2, which has been published as Ref. [26], develops heuristic modifications to belief propagation that overcome these obstacles to allow for improved QLDPC decoding performance.

By ensuring large distance, the vast majority of stabilizer codes have been designed implicitly for good performance on the depolarizing channel, for which X, Y, and Z errors occur with equal probability. However, for a number of quantum channels of interest in the context of quantum computation and communication, phase-flip errors occur far more frequently than bit-flip errors [27, 28]. Such channels are called asymmetric, and when communicating across them, the decoding error rate can be minimized by using a code tailored to the channel [29, 30]. However, distance is a less useful metric in this instance, and directly evaluating a code's performance is limited by the previously mentioned #P-completeness of optimal stabilizer code decoding. Chapter 3, which has been published as Ref. [31], addresses this complexity by developing heuristic methods of constructing of highly performant codes for asymmetric channels.

As noted in Sec. 1.4.3, nonadditive codes can potentially encode a higher dimensional subspace than an optimal (maximum k) stabilizer code detecting the same error set. A particularly promising class of codes are the codeword stabilized (CWS) codes, which encompasses both the

stabilizer codes as well as many of the best known nonadditive codes [32, 33]. A standard form $((n, K))$ CWS code detecting an error set $\mathcal{E}$ is defined by an n-node simple undirected graph G and a classical code of size K that must detect an error set induced from $\mathcal{E}$ by G. This leads to two main obstacles in the construction of optimal CWS codes. The first of these is the exponential increase with n in the number of inequivalent graphs from which a code can be constructed [34, 35, 36]. The second is the NP-hardness of the clique search required to find a maximum-size classical code detecting the error set induced by a given graph [37]. Chapter 4, which has been published as Ref. [38], develops new heuristic methods for constructing CWS codes that address these two issues.

Chapter 5 concludes the thesis, providing a summary of the main results and detailing future research directions.

Chapter 2

Modified belief propagation decoders for quantum low-density parity-check codes[1]

Abstract

Quantum low-density parity-check codes can be decoded using a syndrome-based GF(4) belief propagation decoder (where GF denotes Galois field). However, the performance of this decoder is limited both by unavoidable 4-cycles in the code's factor graph and the degenerate nature of quantum errors. For the subclass of CSS codes, the number of 4-cycles can be reduced by breaking an error into an X and Z component and decoding each with an individual GF(2)-based decoder. However, this comes at the expense of ignoring potential correlations between these two error components. We present a number of modified belief propagation decoders that address these issues. We propose a GF(2)-based decoder for CSS codes that reintroduces error correlations by reattempting decoding with adjusted error probabilities. We also propose the use of an augmented decoder, which has previously been suggested for classical binary low-density parity-check codes. This decoder iteratively reattempts decoding on factor graphs that have a subset of their check nodes duplicated. The augmented decoder can be based on a GF(4) decoder for any code, a GF(2) decoder for CSS code, or even a supernode decoder for a dual-containing CSS code. For CSS codes, we further propose a GF(2)-based decoder that combines the augmented decoder with error probability adjustment. We demonstrate the performance of these new decoders on a range of different codes, showing that they perform favorably compared to other decoders presented in literature.

[1]This chapter has been published as Ref. [26]: A. Rigby, J. C. Olivier, and P. D. Jarvis, "Modified belief propagation decoders for quantum low-density parity-check codes," *Physical Review A*, vol. 100, no. 1, p. 012330, Jul. 2019, doi.org/10.1103/PhysRevA.100.012330. Only minor typographical and formatting changes have been made.

2.1 Introduction

In the classical setting, low-density parity-check (LDPC) codes are effective at protecting information against noise. LDPC codes are particularly useful as their sparse structure permits the use of an iterative belief propagation decoder that is of relatively low complexity [39, 40]. Belief propagation is a message passing algorithm that takes place on a code's factor graph. This is a bipartite graph defined by a parity-check matrix for the code, with each row corresponding to a check node and each column to an error node. Quantum LDPC (QLDPC) codes, which are stabilizer codes with sparse generators, can be used to protect against the effects of a noisy quantum channel. The generators of an n-qubit stabilizer code can be represented as elements of $\mathrm{GF}(4)^n$ [41, 15]. This representation can be used to define a GF(4) parity-check matrix, which allows for slightly altered GF(4) belief propagation decoding of QLDPC codes [42]. The requirement that all stabilizer generators must commute results in unavoidable 4-cycles in the factor graph associated with the GF(4) parity-check matrix [25], which can be detrimental to decoding performance [24]. Belief propagation performance is also limited by the fact that it attempts to converge to the single most likely error (in a symbol-wise fashion), rather than accounting for the degenerate nature of quantum errors [25]. For the subclass of Calderbank-Shor-Steane (CSS) codes, the number of 4-cycles can be reduced by instead representing generators as elements of $\mathrm{GF}(2)^{2n}$ [43, 15]. This allows an error to be broken into an X and Z component, which can then be decoded individually using two GF(2) belief propagation decoders [23]. However, for many channels, including the depolarizing channel, this has the effect of ignoring correlations between the two components [42].

Modified belief propagation decoders have been proposed that aim to improve QLDPC decoding performance. Several decoders are presented in Ref. [25] that aim to alleviate so-called symmetric degeneracy errors, which occur as a result of symbol-wise decoding in the face of error degeneracy. The best performing of these is the random perturbation decoder, which attempts to break decoding symmetries by iteratively reattempting decoding with randomly modified channel error probabilities. The enhanced feedback (EFB) decoder of Ref. [44] behaves similarly in that it also iteratively reattempts decoding with modified error probabilities. However, unlike the random perturbation decoder, this modification is informed by the decoder's output. The supernode decoder of Ref. [42] is a modification to the standard GF(4) decoder for the subclass of dual-containing CSS codes. For this decoder, pairs of check nodes in the factor graph are combined to form supernodes. This both reduces decoding complexity and lowers the number of 4-cycles in the factor graph, which can lead to improved decoding performance.

The augmented decoder that we investigate has been previously proposed for classical binary LDPC codes in Ref. [45]. Like the random perturbation and EFB decoders, it also iteratively reattempts decoding. Each of these attempts employs a version of the standard factor graph with a randomly selected subset of check nodes duplicated. In the classical case, this simple approach gives performance that compares favorably with other, typically more complicated, decoders presented in literature. In this paper, we show that augmented decoders can be applied to QLDPC codes, whether the underlying decoder is GF(2), GF(4), or supernode based. For CSS

codes, we propose the GF(2)-based adjusted decoder, which attempts to reintroduce correlations between the X and Z components of an error that are lost when using a standard GF(2) decoder. If one of the two constituent GF(2) decoders fails, then the adjusted decoder reattempts decoding of this component using error probabilities that are modified according to the output of the other constituent decoder (this is a slight generalization of the decoder presented in Ref. [46]). We also present a GF(2)-based decoder for CSS codes that combines the augmented and adjusted decoders. We simulate the performance of our decoders on six different codes: two dual-containing CSS codes, two nondual-containing CSS codes, and two non-CSS codes. We show that for dual-containing CSS codes, our augmented GF(4), augmented supernode, and combined decoders all outperform random perturbation and EFB decoders. For the four other codes, we demonstrate that augmented GF(4) and supernode decoders perform similarly to to the random perturbation and EFB decoders.

The paper is organized as follows. Section 2.2 gives an overview of belief propagation decoding for classical LDPC codes and extends this to the quantum case. Section 2.3 details the operation of existing modified decoders (random perturbation, EFB, and supernode) and describes the adjusted, augmented, and combined decoders that we propose. Section 2.4 presents simulation results for our decoders on six different codes, comparing them to existing decoders. The paper is concluded in Sec. 2.5.

2.2 Background

2.2.1 Classical codes

A classical channel is a map $\Phi : \mathcal{A}_x \to \mathcal{A}_y$, where $\mathcal{A}_x$ is the set of possible inputs and $\mathcal{A}_y$ is the set of possible outputs. We are concerned with channels where the input and output sets are finite fields with q elements; that is, $\mathcal{A}_x = \mathcal{A}_y = \mathrm{GF}(q)$. In this case, the action of the channel can be expressed as

$$\Phi(x) = x + e = y, \tag{2.1}$$

where $x \in \mathrm{GF}(q)$ is the channel input, $y \in \mathrm{GF}(q)$ is the channel output, and $e \in \mathrm{GF}(q)$ is an error (or noise) symbol that occurs with probability $P(e)$. A channel Φ is called symmetric if $P(0) = 1 - p$ and $P(e_i) = p/(q-1)$ for $e_i \neq 0$. A code $\mathcal{C} \subseteq \mathrm{GF}(q)^n$ can be used to protect against the noise introduced by the channel. Elements $\boldsymbol{x} \in \mathcal{C}$, called codewords, are transmitted as n sequential uses of Φ or, equivalently, as a single use of the combined channel Φ^n, which is comprised of n copies of Φ. The action of Φ^n on some input $\boldsymbol{x} \in \mathcal{C}$ is

$$\Phi^n(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{e} = \boldsymbol{y}, \tag{2.2}$$

where $\boldsymbol{y} \in \mathrm{GF}(q)^n$ is the channel output and $\boldsymbol{e} \in \mathrm{GF}(q)^n$ is an error "vector." Assuming the error components are independent, the probability of an error $\boldsymbol{e} = (e_1, \ldots, e_n)$ occurring is

$$P(\boldsymbol{e}) = \prod_{i=1}^{n} P(e_i), \tag{2.3}$$

where $P(e_i)$ is the probability of the error symbol e_i occurring on Φ. The weight of a codeword $\boldsymbol{x} \in \mathcal{C}$ or an error $\boldsymbol{e} \in \mathrm{GF}(q)^n$ is the number of nonzero components it contains. It follows from Eq. (2.3) that if Φ is symmetric, then the probability of $\boldsymbol{e} \in \mathrm{GF}(q)^n$ occurring depends only on its weight. The distance between two codewords $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}$, denoted $\Delta(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is the number of components in which they differ. The distance of $\mathcal{C}$ is

$$d = \min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}} \Delta(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{2.4}$$

Equivalently, the distance of $\mathcal{C}$ is equal to the weight of the lowest-weight error that maps one codeword to another.

If a code $\mathcal{C} \subseteq \mathrm{GF}(q)^n$ forms an (additive) group, then it is called additive; if it forms a vector space, then it is called linear (note that there is no distinction between additive and linear codes in the binary case). Suppose a linear code $\mathcal{C}$ has a basis $\mathcal{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\}$. This defines a generator matrix

$$G^T = \begin{pmatrix} \boldsymbol{b}_1 & \cdots & \boldsymbol{b}_k \end{pmatrix}, \tag{2.5}$$

where the basis elements are considered as column vectors. A generator matrix can be defined in the same way for an additive code; however, in this case, $\mathcal{B}$ is a generating set. For a linear code, the generator matrix defines a bijective encoding operation that maps some $\boldsymbol{d} \in \mathrm{GF}(q)^k$ to a codeword $\boldsymbol{x} = G^T\boldsymbol{d} \in \mathcal{C}$ ($\boldsymbol{d}$ is also considered as a column vector). A linear code can also be defined as the kernel of a $\mathrm{GF}(q)$ parity-check matrix H; that is,

$$\mathcal{C} = \{\boldsymbol{x} \in \mathrm{GF}(q)^n : H\boldsymbol{x} = \boldsymbol{0}\}. \tag{2.6}$$

Note that for a given code, neither the generator or parity-check matrix is unique. If H has m rows, then $\dim(\mathcal{C}) = k \geq n - m$, with equality when H is full rank. If $\mathcal{C}$ is linear with dimension k and distance d, then it is called an $[n, k]_q$ or $[n, k, d]_q$ code (the q is typically omitted for binary codes, where $q = 2$). For a linear code, this distance is equal to weight of the minimum-weight nonzero codeword (as the errors that map one codeword to another are the nontrivial $\boldsymbol{e} \in \mathcal{C}$). The rate of a code is given by $R = k/n$.

The dual code of some code $\mathcal{C} \subseteq \mathrm{GF}(q)^n$ with respect to the inner product $\langle \cdot, \cdot \rangle : \mathrm{GF}(q)^n \times \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ is

$$\mathcal{C}^\perp = \{\boldsymbol{c} \in \mathrm{GF}(q)^n : \langle \boldsymbol{c}, \boldsymbol{x} \rangle = 0 \ \forall \ \boldsymbol{x} \in \mathcal{C}\}. \tag{2.7}$$

$\mathcal{C}^\perp$ is the annihilator of $\mathcal{C}$ and is therefore a linear code. If $\mathcal{C}^\perp \subseteq \mathcal{C}$, then $\mathcal{C}$ is called dual containing; if $\mathcal{C} \subseteq \mathcal{C}^\perp$, then $\mathcal{C}$ is called self-orthogonal; and if $\mathcal{C}^\perp = \mathcal{C}$, then $\mathcal{C}$ is called self-dual. Unless otherwise specified, the dual code is with respect to the Euclidean inner product

$$\langle \boldsymbol{c}, \boldsymbol{x} \rangle = \boldsymbol{c} \cdot \boldsymbol{x} = \sum_{i=1}^{n} c_i x_i. \tag{2.8}$$

In this case, if $\mathcal{C}$ is linear with generator matrix G, then a necessary and sufficient condition for $\boldsymbol{c} \in \mathcal{C}^\perp$ is $G\boldsymbol{c} = \boldsymbol{0}$; that is, a generator matrix for $\mathcal{C}$ is a parity-check matrix for $\mathcal{C}^\perp$. Conversely, if H is a parity-check matrix for $\mathcal{C}$, then it is a generator matrix for $\mathcal{C}^\perp$.

The aim of a decoder is to determine the channel's input given its output. For a linear code $\mathcal{C}$,

this decoder can make use of the error syndrome. If $\mathcal{C}$ has an $m \times n$ parity-check matrix H and the channel output is $\boldsymbol{y}$, then the syndrome is

$$\boldsymbol{z} = H\boldsymbol{y} = H(\boldsymbol{x} + \boldsymbol{e}) = H\boldsymbol{e} \in \mathrm{GF}(q)^m. \tag{2.9}$$

An optimal decoder returns the most probable error given the syndrome measurement

$$\hat{\boldsymbol{e}} = \underset{\boldsymbol{e} \in \mathrm{GF}(q)^n}{\operatorname{argmax}} P(\boldsymbol{e}|\boldsymbol{z}) = \underset{\boldsymbol{e} \in \mathrm{GF}(q)^n}{\operatorname{argmax}} P(\boldsymbol{e})\delta(H\boldsymbol{e} = \boldsymbol{z}), \tag{2.10}$$

where $\delta(H\boldsymbol{e} = \boldsymbol{z}) = 1$ if $H\boldsymbol{e} = \boldsymbol{z}$ and 0 otherwise. The channel input can then be estimated as $\hat{\boldsymbol{x}} = \boldsymbol{y} - \hat{\boldsymbol{e}}$. If $\hat{\boldsymbol{e}} = \boldsymbol{e}$ (and hence $\hat{\boldsymbol{x}} = \boldsymbol{x}$), then decoding is successful; otherwise, a decoding error has occurred. Unfortunately, even in the simple case of a binary code operating on the binary symmetric channel (a symmetric channel with $q = 2$), this decoding problem can be shown to be NP-complete [47].

It follows from Eq. (2.9) that the syndrome resulting from some error $\boldsymbol{e} \in \mathrm{GF}(q)^n$ depends only on which coset of $\mathrm{GF}(q)^n/\mathcal{C}$ it belongs to. If $\hat{\boldsymbol{e}}$ is the most probable error in the coset $\boldsymbol{e} + \mathcal{C}$, then the probability of a decoding failure given the syndrome $\boldsymbol{z} = H\boldsymbol{e}$ is

$$P(\boldsymbol{e} \neq \hat{\boldsymbol{e}}|\boldsymbol{z}) = \frac{P(\boldsymbol{e} + \mathcal{C}) - P(\hat{\boldsymbol{e}})}{P(\boldsymbol{e} + \mathcal{C})}, \tag{2.11}$$

where $P(\boldsymbol{e}+\mathcal{C})$ is the probability of any error in $\boldsymbol{e}+\mathcal{C}$ occurring. Therefore, the probability of a decoding error is high if the error probability distribution over $\boldsymbol{e}+\mathcal{C}$ is not sharply peaked [that is, if $P(\hat{\boldsymbol{e}})$ is small]. If the channel is symmetric, then this corresponds to $\boldsymbol{e}+\mathcal{C}$ containing errors with similar weight to $\hat{\boldsymbol{e}}$, which will be the case if $\mathcal{C}$ contains low-weight codewords. It therefore follows that the distance of $\mathcal{C}$ gives some indication of the fraction of transmissions that will not be decoded correctly, which is called call the frame error rate (FER).

2.2.2 Factor graphs and belief propagation

The factor graph of a linear code is a bipartite graph $G = (V, C, E)$. The error nodes $V = \{v_1, \ldots, v_n\}$ correspond to the n error components, and the check nodes $C = \{c_1, \ldots, c_m\}$ correspond to the m constraints imposed by the rows of a parity-check matrix H. An edge $\{c_i, v_j\} \in E$ connects check node c_i to error node v_j if $H_{ij} \neq 0$. For example, the $[7, 4, 3]$ Hamming code of Ref. [48] can be defined by the parity-check matrix

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{2.12}$$

which gives the factor graph shown in Fig. 2.1. In general, a given code does not have a unique factor graph as the parity-check matrix from which it is defined is not unique. Furthermore, except in the case of a binary code, the mapping from a parity-check matrix to its corresponding factor graph is not one-to-one as an edge only indicates that $H_{ij} \neq 0$; it does not give the value of H_{ij} (although this information can be included by decorating the edges). A walk is a sequence whose elements alternate between connected nodes and the edges that connect them. The length

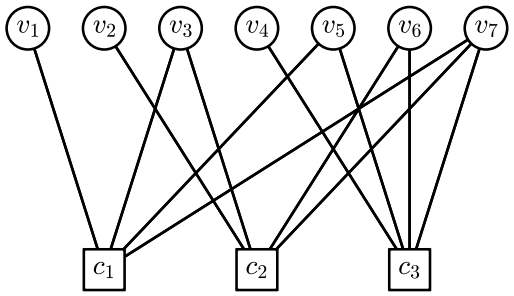

Figure 2.1: The factor graph of the $[7, 4, 3]$ Hamming code corresponding to the parity-check matrix given in Eq. (2.12). Error nodes are represented as circles and check nodes as squares.

of a walk is the number of edges it contains. A path is a walk containing no repeated nodes or edges, with the exception that the first and last node can be the same, in which case the path is called a cycle. The bipartite nature of a code's factor graph ensures that the size of all cycles is even and greater than or equal to four. As an example, the walk $c_1, \{c_1, v_5\}, v_5, \{c_3, v_5\}, c_3, \{c_3, v_7\}, v_7, \{c_1, v_7\}, c_1$ in the graph of Fig. 2.1 is a 4-cycle (that is, a cycle of length four). Typically a code's factor graph will not be cycle-free (that is, it will not be a tree) as if a code has such a representation, then its distance is bounded by [49]

$$d \leq \left\lfloor \frac{n}{k+1} \right\rfloor + \left\lfloor \frac{n+1}{k+1} \right\rfloor. \tag{2.13}$$

For $R \geq 1/2$, this reduces to $d \leq 2$, and for $R > 1/2$, it reduces to $d \lesssim 2\lfloor 1/R \rfloor$.

The factor graph representation of a linear code serves as the foundation for a belief propagation decoder. Instead of determining the most likely error as given in Eq. (2.10), a belief propagation decoder approximates it in a symbol-wise fashion. This gives an estimate $\hat{\boldsymbol{e}} = (\hat{e}_1, \ldots, \hat{e}_n)$, where

$$\hat{e}_j = \underset{e_j \in \mathrm{GF}(q)}{\operatorname{argmax}} P(e_j | \boldsymbol{z}). \tag{2.14}$$

An expression for $P(e_j|\boldsymbol{z})$ can be obtained by marginalizing $P(\boldsymbol{e}|\boldsymbol{z})$. Assuming that the error components are independent,

$$P(\boldsymbol{e}|\boldsymbol{z}) \propto \prod_{l=1}^{n} P(e_l)\delta(H\boldsymbol{e} = \boldsymbol{z}) = \prod_{l=1}^{n} P(e_l) \prod_{i=1}^{m} \delta\left(\sum_{j=1}^{n} H_{ij} e_j = z_i \right). \tag{2.15}$$

Fixing $e_j = a$ and summing over all other components gives

$$P(e_j = a|\boldsymbol{z}) \propto \sum_{\boldsymbol{e}: e_j = a} \prod_{l=1}^{n} P(e_l) \prod_{i=1}^{m} \delta\left(\sum_{j=1}^{n} H_{ij} e_j = z_i \right). \tag{2.16}$$

Belief propagation efficiently approximates these marginals by passing messages on the code's factor graph. For a code over $\mathrm{GF}(q)$, these messages will be vectors of length q. Initially, a message is sent from every error node v_j to the check nodes in the neighborhood $\mathcal{N}(v_j) = \{c_i \in C : \{c_i, v_j\} \in E\}$. In particular, the message sent to check node $c_i \in \mathcal{N}(v_j)$ is $\mu_{j \to i}$, where the element corresponding to $a \in \mathrm{GF}(q)$ is

$$\mu^a_{j \to i} = P(e_j = a). \tag{2.17}$$

Note that this message simply gives the channel error probabilities. Every check node c_i then

sends a message back to the error nodes in the neighborhood $\mathcal{M}(c_i) = \{v_j \in V : \{c_i, v_j\} \in E\}$. In particular, the message sent to error node $v_j \in \mathcal{M}(c_i)$ is $\lambda_{i\to j}$, with

$$\lambda^a_{i\to j} = K \sum_{\boldsymbol{e}:e_j=a} \delta\left(\sum_{j'\in\mathcal{M}(i)} H_{ij'}e_{j'} = z_i\right) \prod_{j'\in\mathcal{M}(i)\setminus j} \mu^{e_{j'}}_{j'\to i}, \tag{2.18}$$

where, through slight abuse of notation, $\mathcal{M}(i) = \{j \in \{1,\dots,n\} : v_j \in \mathcal{M}(c_i)\}$ and K is a normalization factor chosen such that $\sum_a \lambda^a_{i\to j} = 1$. An estimate of the marginal probability $P(e_j|\boldsymbol{z})$ can then be made, with

$$\hat{P}(e_j = a|\boldsymbol{z}) = KP(e_j = a) \prod_{i\in\mathcal{N}(j)} \lambda^a_{i\to j}, \tag{2.19}$$

where $\mathcal{N}(j) = \{i \in \{1,\dots,m\} : c_i \in \mathcal{N}(v_j)\}$ and K is a normalization factor. From this, $\hat{\boldsymbol{e}}$ can be estimated in a symbol-wise fashion as in Eq. (2.14). If $\hat{\boldsymbol{z}} = H\hat{\boldsymbol{e}} = \boldsymbol{z}$, then decoding is complete; otherwise, another message is sent from each error node to its connected check nodes. The elements of this message are

$$\mu^a_{j\to i} = KP(e_j = a) \prod_{i'\in\mathcal{N}(j)\setminus i} \lambda^a_{i'\to j}, \tag{2.20}$$

where K is again a normalization factor. There is then another round of check to error node messages as in Eq. (2.18), followed by an approximation of marginals as in Eq. (2.19). This process of sending error to check messages followed by check to error messages and a computation of marginals proceeds iteratively until either $\hat{\boldsymbol{z}} = \boldsymbol{z}$ or a maximum number of iterations $I_{\max}$ is reached. The most computationally complex component of belief propagation is the check to error node message calculation of Eq. (2.18). However, it can be performed efficiently using a Fourier transform as outlined in Appendix 2.A.1.

There are two types of decoding error exhibited by a belief propagation decoder. The first type is the detected error, where decoding ends with $\hat{\boldsymbol{z}} \neq \boldsymbol{z}$ (and hence $\hat{\boldsymbol{e}} \neq \boldsymbol{e}$). Such errors do not occur when using an optimal decoder and, as such, are fundamentally a failing of the belief propagation decoder itself. The second type of error is the undetected error, where decoding ends with $\hat{\boldsymbol{z}} = \boldsymbol{z}$ but $\hat{\boldsymbol{e}} \neq \boldsymbol{e}$. These are the same type of error exhibited by the optimal decoder and, as such, can be attributed to a failing of the code. It therefore follows that for a symmetric channel, using a code with a lower distance will tend to result in a higher rate of undetected errors.

Belief propagation decoding is an approximation on two levels. Firstly, it assumes that the most likely error is equal to the symbol-wise most likely error. Secondly, the estimate of the symbol-wise most likely error is based on the approximate marginal probabilities $\hat{P}(e_j|\boldsymbol{z})$ that are only exact when the code's factor graph is a tree [50], which as previously outlined, is unlikely. However, good decoding performance can still be achieved when the factor graph is sparsely connected [50]. Linear codes with such a representation are called low-density parity-check (LDPC) codes (most codes do not have such a representation [51, 50]). Decoding performance is further improved when the factor graph contains few short cycles [24].

2.2.3 Stabilizer codes

The action of a quantum channel Φ on a quantum state described by the density operator ρ is

$$\Phi(\rho) = \sum_k A_k \rho A_k^\dagger, \tag{2.21}$$

where the A_k, called Kraus operators, satisfy $\sum_k A_k^\dagger A_k = I$ (the identity operator) [14]. In this paper, we are interested in qubit systems; that is, systems where states $|\phi\rangle$ belong to a two-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^2$. Furthermore, we are concerned with Pauli channels. These are channels of the form

$$\Phi(\rho) = p_I \rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z, \tag{2.22}$$

where in the computational $\{|0\rangle, |1\rangle\}$ basis,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.23}$$

The action of this channel can be interpreted as mapping a pure state $|\phi\rangle$ to $E|\phi\rangle$, where the error E is I with probability p_I, X with probability p_X, Y with probability p_Y, or Z with probability p_Z [6]. X can be viewed as a bit flip operator as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Z can be viewed as a phase flip as $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. $Y = iXZ$ can be viewed as a combined bit and phase flip. Of particular interest is the depolarizing channel, where $p_I = 1-p$ and $p_X = p_Y = p_Z = p/3$. We are also interested in the XZ channel, for which the X and Z components of an error $E \propto X^u Z^v$, where $u, v \in \mathrm{GF}(2)$, occur independently with equal probability q. It follows from the independence of the error components that $p_X = p_Z = q(1-q)$ and $p_Y = q^2$. These values can be expressed in terms of the total error probability $p = p_X + p_Y + p_Z$ as $q = 1 - \sqrt{1-p}$, $p_X = p_Z = \sqrt{1-p}(1-\sqrt{1-p})$, and $p_Y = (1-\sqrt{1-p})^2$.

The Pauli matrices are Hermitian, unitary, and anticommute with each other. Furthermore, they form a group called the Pauli group

$$\mathcal{P}_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} = \langle X, Y, Z\rangle. \tag{2.24}$$

The n-qubit Pauli group $\mathcal{P}_n$ is defined as all n-fold tensor product combinations of elements of $\mathcal{P}_1$. For example, $\mathcal{P}_8$ contains the element $I \otimes I \otimes X \otimes I \otimes Y \otimes Z \otimes I \otimes I$, which is often written more compactly as $IIXIYZII$ or $X_3Y_5Z_6$. The weight of some $g \in \mathcal{P}_n$ is the number of elements in the tensor product that are not equal to the identity up to phase. The commutation relations of the Pauli matrices mean that elements of $\mathcal{P}_n$ must either commute or anticommute, with two elements anticommuting if their nonidentity components differ in an odd number of places.

Similar to the classical case, the noise introduced by a quantum channel can be protected against by employing a code. A quantum (qubit) code is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$. Codewords $|\phi\rangle \in \mathcal{Q}$ are transmitted across the combined n-qubit channel $\Phi^{\otimes n}$. If Φ is a Pauli channel, then $\Phi^{\otimes n}$ maps codewords $|\phi\rangle$ to $E|\phi\rangle$, where $E \in \mathcal{P}_n$. Assuming the channel acts on each qubit independently,

the probability of an error E occurring (up to phase) is

$$P(E) = \prod_{i=1}^{n} P(E_i), \tag{2.25}$$

where $P(E_i)$ is the probability of the error E_i occurring (up to phase) on the single-qubit channel Φ. Note that errors are considered up to phase as the resulting state is equivalent up to such a phase factor. A convenient way of handling this is to group errors in $\mathcal{P}_n$ up to phase, with $\tilde{E} = \{E, -E, iE, -iE\} \in \mathcal{P}_n/\{\pm I, \pm iI\} = \tilde{\mathcal{P}}_n$.

Stabilizer codes are defined by an abelian subgroup $\mathcal{S} < \mathcal{P}_n$, called the stabilizer, that does not contain $-I$ [15]. The code $\mathcal{Q}$ is the space of states that are fixed by every element $s_i \in \mathcal{S}$; that is,

$$\mathcal{Q} = \{|\phi\rangle \in (\mathbb{C}^2)^{\otimes n} : s_i|\phi\rangle = |\phi\rangle \; \forall \, s_i \in \mathcal{S}\}. \tag{2.26}$$

The requirement that $-I \notin \mathcal{S}$ both means that no $s \in \mathcal{S}$ can have a phase factor of $\pm i$, and that if $s \in \mathcal{S}$, then $-s \notin \mathcal{S}$. If $\mathcal{S}$ is generated by $M = \{M_1, \ldots, M_m\} \subset \mathcal{P}_n$, then it is sufficient (and obviously necessary) for $\mathcal{Q}$ to be stabilized by every M_i. Assuming that the set of generators is minimal, it can be shown that $\dim(\mathcal{Q}) = 2^{n-m} = 2^k$ [6]; that is, $\mathcal{Q}$ encodes the state of a k-qubit system. If the generators of $\mathcal{S}$ are sparse, then $\mathcal{Q}$ is called a quantum LDPC (QLDPC) code.

Suppose an error E occurs, mapping some codeword $|\phi\rangle \in \mathcal{Q}$ to $E|\phi\rangle$. A projective measurement of a generator M_i will give the result $+1$ if $[E, M_i] = EM_i - M_iE = 0$ or -1 if $\{E, M_i\} = EM_i + M_iE = 0$. These measurement values define the syndrome $\boldsymbol{z} \in \mathrm{GF}(2)^m$, with

$$z_i = \begin{cases} 0 & \text{if } [E, M_i] = 0, \\ 1 & \text{if } \{E, M_i\} = 0. \end{cases} \tag{2.27}$$

There are three classes of error that can occur (note that the following paragraph will give greater context to this classification of errors). The first class are those errors $\tilde{E} = \{E, -E, iE, -iE\} \in \tilde{\mathcal{S}}$, where $\tilde{\mathcal{S}}$ is the group

$$\tilde{\mathcal{S}} = \{\tilde{s} = \{s, -s, is, -is\} : s \in \mathcal{S}\}. \tag{2.28}$$

Such errors have no effect on the code and result in the trivial syndrome $\boldsymbol{z} = \mathbf{0}$ (as the stabilizer is abelian). The second class of errors are those $\tilde{E} \in \widetilde{C(\mathcal{S})} \backslash \tilde{\mathcal{S}}$, where $C(\mathcal{S})$ is the centralizer of $\mathcal{S}$ in $\mathcal{P}_n$ and $\widetilde{C(\mathcal{S})} \subseteq \tilde{\mathcal{P}}_n$ is defined in the same way as $\tilde{\mathcal{S}}$ in Eq. (2.28). In this case, $C(\mathcal{S})$ is actually equal to the normalizer $N(\mathcal{S})$ [15]. These are errors that commute with every stabilizer and therefore also yield $\boldsymbol{z} = \mathbf{0}$; however, the effect of such errors on the code is nontrivial. The final class of errors are those $\tilde{E} \in \tilde{\mathcal{P}}_n \backslash \widetilde{N(\mathcal{S})}$, which yield nontrivial syndromes $\boldsymbol{z} \neq \mathbf{0}$ and also act nontrivially on the code. In general, the syndrome resulting from some error $\tilde{E} \in \tilde{\mathcal{P}}_n$ depends only on which coset of $\tilde{\mathcal{P}}_n / \widetilde{N(\mathcal{S})}$ it belongs to, while its effect on the code depends only on which coset of $\tilde{\mathcal{P}}_n/\tilde{\mathcal{S}}$ it belongs to [note that $\tilde{\mathcal{S}} \lhd \widetilde{N(\mathcal{S})} \lhd \tilde{\mathcal{P}}_n$ as $\tilde{\mathcal{P}}_n$ is abelian]. This phenomena of distinct errors having an identical effect on a code is called degeneracy and has no classical analog. In the classical case, the distance d of a linear code is equal to the weight of the lowest-weight error yielding a trivial syndrome while having a nontrivial effect on the code. This extends to the quantum case, with the distance d of a stabilizer code being the weight of

the lowest-weight element in $\widetilde{N(\mathcal{S})}\backslash\tilde{\mathcal{S}}$ [15]. An n-qubit code of dimension 2^k with distance d is called an $[[n,k]]$ or $[[n,k,d]]$ code (the double brackets differentiate it from a classical code).

From a decoding point of view, the syndrome measurement determines which coset of $\tilde{\mathcal{P}}_n/\widetilde{N(\mathcal{S})}$ an error $\tilde{E}$ belongs to. If this coset has the representative $\tilde{g} \in \tilde{\mathcal{P}}_n$, then an ideal decoder determines the coset $\hat{A}$ in $(\tilde{g}\widetilde{N(\mathcal{S})})/\tilde{\mathcal{S}}$ that $\tilde{E}$ is most likely to belong to. Importantly, $\hat{A}$ does not necessarily contain the individually most likely error in $\tilde{g}\widetilde{N(\mathcal{S})}$. If $\hat{A}$ has the representative $\hat{E} = \{\hat{E}, -\hat{E}, i\hat{E}, -i\hat{E}\}$, then the decoder attempts to correct the channel error by applying $\hat{E}$ to the channel output. If $\tilde{E} \in \hat{A}$, then $\hat{E}\tilde{E} \in \tilde{\mathcal{S}}$, and as such, this process corrects the error; otherwise, if $\tilde{E} \notin \hat{A}$, then a decoding error has occurred. Similar to the classical case, the probability of a decoding failure given some syndrome measurement $\boldsymbol{z}$ is

$$P(\tilde{E} \notin \hat{A}|\boldsymbol{z}) = \frac{P(\tilde{g}\widetilde{N(\mathcal{S})}) - P(\hat{A})}{P(\tilde{g}\widetilde{N(\mathcal{S})})}, \tag{2.29}$$

where $P(\tilde{g}\widetilde{N(\mathcal{S})})$ and $P(\hat{A})$ are the probabilities of an error being in $\tilde{g}\widetilde{N(\mathcal{S})}$ or $\hat{A}$, respectively. From this, it follows that the probability of a decoding error is high if the probability distribution over $(\tilde{g}\widetilde{N(\mathcal{S})})/\tilde{\mathcal{S}}$ is not sharply peaked, which will occur if $\widetilde{N(\mathcal{S})}\backslash\tilde{\mathcal{S}}$ contains high-probability errors. For the depolarizing channel, this corresponds to $\widetilde{N(\mathcal{S})}\backslash\tilde{\mathcal{S}}$ containing low-weight elements, meaning that the distance d gives some indication of decoder performance.

2.2.4 Stabilizer code representations

It is possible to represent elements of $\tilde{\mathcal{P}}_1$ as elements of $\mathrm{GF}(2)^2$ according to the isomorphism [43, 15]

$$I \leftrightarrow (0,0),\ X \leftrightarrow (1,0),\ Y \leftrightarrow (1,1),\ Z \leftrightarrow (0,1). \tag{2.30}$$

This can be extended to elements of $\tilde{\mathcal{P}}_n$ according to

$$X^{u_1}Z^{v_1} \otimes \cdots \otimes X^{u_n}Z^{v_n} \leftrightarrow (u_1, \ldots, u_n|v_1, \ldots, v_n). \tag{2.31}$$

This can be written more compactly as $X^{\boldsymbol{u}}Z^{\boldsymbol{v}} \leftrightarrow (\boldsymbol{u}|\boldsymbol{v}) \in \mathrm{GF}(2)^{2n}$, where $\boldsymbol{u} = (u_1, \ldots, u_n)$, $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathrm{GF}(2)^n$. The product of elements in $\tilde{\mathcal{P}}_n$ corresponds to addition in $\mathrm{GF}(2)^{2n}$. Representatives of elements in $\tilde{\mathcal{P}}_n$ commute if the symplectic inner product of the binary representations is zero; otherwise, they anticommute. Note that the symplectic inner product of $\boldsymbol{a} = (\boldsymbol{u}|\boldsymbol{v}) \in \mathrm{GF}(2)^{2n}$ and $\boldsymbol{b} = (\boldsymbol{u}'|\boldsymbol{v}') \in \mathrm{GF}(2)^{2n}$ is

$$\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{u} \cdot \boldsymbol{v}' + \boldsymbol{u}' \cdot \boldsymbol{v} = \sum_{i=1}^{n} (u_i v_i' + u_i' v_i). \tag{2.32}$$

Considering $\boldsymbol{a}$ and $\boldsymbol{b}$ as row vectors, this simplifies to $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{a}P\boldsymbol{b}^T$, where P is the $2n \times 2n$ matrix

$$P = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix}. \tag{2.33}$$

The binary representations of the m generators of some stabilizer $\mathcal{S}$ define the rows of an $m \times 2n$

binary matrix H. This matrix has the form

$$H = (H_X|H_Z), \tag{2.34}$$

where H_X and H_Z are each $m \times n$ matrices. Note that while H only defines a stabilizer up to phase $\tilde{\mathcal{S}}$, the codes defined by different stabilizers corresponding to $\tilde{\mathcal{S}}$ will all have the same error correction properties. Considering H as the parity-check matrix of a classical binary code $\mathcal{C}$, the stabilizer elements correspond to elements of the dual code $\mathcal{C}^{\perp}$. The requirement that all stabilizer generators commute becomes

$$H_X H_Z^T + H_Z H_X^T = 0. \tag{2.35}$$

Any classical linear code with a parity-check matrix H that satisfies this constraint can be used to define a stabilizer code. Furthermore, if H is sparse, then this stabilizer code is a QLDPC code. Errors can also be considered within the binary framework. Suppose that some error $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z}$ occurs. This error has the binary representation $\boldsymbol{e} = (\boldsymbol{e}_X^T|\boldsymbol{e}_Z^T)^T$, and the corresponding syndrome is simply $\boldsymbol{z} = HP\boldsymbol{e}$ (where $\boldsymbol{e}_X$, $\boldsymbol{e}_Z$, and $\boldsymbol{e}$ are column vectors for consistency with the classical case).

A subclass of stabilizer codes are the Calderbank-Shor-Steane (CSS) codes [52, 53], which have a binary representation of the form

$$H = \left(\begin{array}{c|c} \tilde{H}_X & 0 \\ 0 & \tilde{H}_Z \end{array}\right). \tag{2.36}$$

The commutation condition of Eq. (2.35) becomes $\tilde{H}_Z \tilde{H}_X^T = 0$ (or equivalently $\tilde{H}_X \tilde{H}_Z^T = 0$). Considering $\tilde{H}_X$ and $\tilde{H}_Z$ as parity-check matrices for classical codes $\mathcal{C}_X$ and $\mathcal{C}_Z$, respectively, this commutation condition requires that $\mathcal{C}_X^{\perp} \subseteq \mathcal{C}_Z$ (or equivalently $\mathcal{C}_Z^{\perp} \subseteq \mathcal{C}_X$). If $\tilde{H}_Z = \tilde{H}_X$, then $\mathcal{C}_Z = \mathcal{C}_X$, which gives $\mathcal{C}_X^{\perp} \subseteq \mathcal{C}_X$. Such codes are called dual-containing CSS codes.

Elements of $\tilde{\mathcal{P}}_1$ can also be represented as elements of $\mathrm{GF}(4) = \{0, 1, \omega, \omega^2 = \bar{\omega}\}$ according to the isomorphism [41, 15]

$$I \leftrightarrow 0,\ X \leftrightarrow 1,\ Y \leftrightarrow \bar{\omega},\ Z \leftrightarrow \omega. \tag{2.37}$$

Elements of $\tilde{\mathcal{P}}_n$ then map to elements of $\mathrm{GF}(4)^n$, with the product of elements in $\tilde{\mathcal{P}}_n$ corresponding to addition in $\mathrm{GF}(4)^n$ [GF(4) addition and multiplication are defined in Tables 2.1 and 2.2, respectively]. Representatives of elements in $\tilde{\mathcal{P}}_n$ commute if the trace inner product of the corresponding elements of $\mathrm{GF}(4)^n$ is zero. Note that the trace inner product of $\boldsymbol{a}, \boldsymbol{b} \in \mathrm{GF}(4)^n$ is

$$\boldsymbol{a} * \boldsymbol{b} = \mathrm{tr}(\boldsymbol{a} \cdot \bar{\boldsymbol{b}}) = \mathrm{tr}\left(\sum_{i=1}^{n} a_i \bar{b_i}\right), \tag{2.38}$$

where $\bar{0} = 0$, $\bar{1} = 1$, $\bar{\omega} = \omega^2$, and $\bar{\omega^2} = \omega$; and $\mathrm{tr}(x) = x + \bar{x}$ [that is, $\mathrm{tr}(0) = \mathrm{tr}(1) = 0$ and $\mathrm{tr}(\omega) = \mathrm{tr}(\bar{\omega}) = 1$].

The $\mathrm{GF}(4)^n$ representations of the m generators of some stabilizer $\mathcal{S}$ define an $m \times n$ GF(4) matrix H in much the same way as the binary case. A stabilizer with the GF(2) representation

Table 2.1: GF(4) addition.

$+$	0	1	ω	$\bar{\omega}$
0	0	1	ω	$\bar{\omega}$
1	1	0	$\bar{\omega}$	ω
ω	ω	$\bar{\omega}$	0	1
$\bar{\omega}$	$\bar{\omega}$	ω	1	0

Table 2.2: GF(4) multiplication.

$\times$	0	1	ω	$\bar{\omega}$
0	0	0	0	0
1	0	1	ω	$\bar{\omega}$
ω	0	ω	$\bar{\omega}$	1
$\bar{\omega}$	0	$\bar{\omega}$	1	ω

of Eq. (2.34) has the GF(4) representation

$$H = H_X + \omega H_Z. \tag{2.39}$$

For a CSS code, this becomes

$$H = \begin{pmatrix} \tilde{H}_X \\ \omega \tilde{H}_Z \end{pmatrix}, \tag{2.40}$$

with $\tilde{H}_X$ and $\tilde{H}_Z$ as defined in Eq. (2.36). The stabilizer corresponds to the additive group generated by the rows of H. This group can be considered as an additive classical code $\mathcal{C}$ over GF(4). The rows of H must be orthogonal with respect to the trace inner product. Therefore, if $\mathcal{C}^\perp$ is the dual code of $\mathcal{C}$ with respect to the trace inner product, then $\mathcal{C} \subseteq \mathcal{C}^\perp$. Any such self-orthogonal additive GF(4) code can be used to define a stabilizer code. Errors can also be considered in the GF(4) framework. An error E with GF(4) representation $\boldsymbol{e}$ (again, taken to be a column vector) will yield a syndrome $\boldsymbol{z} = \mathrm{tr}(H\bar{\boldsymbol{e}})$. Note that while H is a generator matrix for $\mathcal{C}$, we essentially consider it as a parity-check matrix because of the role it plays in syndrome calculation and hence in belief propagation decoding.

2.2.5 Belief propagation decoding for stabilizer codes

Belief propagation decoding can be applied to stabilizer codes using the GF(2) and GF(4) representations of the previous section. Such a belief propagation decoder aims to estimate the symbol-wise most likely error (up to phase) $\hat{E} = \hat{E}_1 \otimes \cdots \otimes \hat{E}_n$, where

$$\hat{E}_j = \underset{E_j}{\mathrm{argmax}}\, P(E_j|\boldsymbol{z}). \tag{2.41}$$

A GF(4)-based belief propagation decoder can be used for any QLDPC code. This decoder attempts to make a symbol-wise estimate $\hat{\boldsymbol{e}} \in \mathrm{GF}(4)^n$ that maps to $\hat{E}$ according to the isomorphism outlined in Sec. 2.2.4. The GF(4) decoder behaves very similarly to the belief propagation decoder presented for classical linear codes in Sec. 2.2.2. The only change is to account for the difference in syndrome calculation. In particular, the check to error node message is modified to

$$\lambda^a_{i\to j} = K \sum_{\boldsymbol{e}:e_j=a} \delta\left[\mathrm{tr}\left(\sum_{j'\in\mathcal{M}(i)} H_{ij'}\bar{e_{j'}}\right) = z_i\right] \prod_{j'\in\mathcal{M}(i)\setminus j} \mu^{e_{j'}}_{j'\to i}. \tag{2.42}$$

This calculation can also be performed efficiently using a Fourier transform as outlined in Appendix 2.A.2. The channel error probabilities used in error to check node messages [Eqs. (2.17) and (2.20)] and in marginal calculation [Eq. (2.19)] are $P(e_j = 0) = 1 - p$, $P(e_j = 1) = p_X$, $P(e_j = \bar{\omega}) = p_Y$, and $P(e_j = \omega) = p_Z$.

For the subclass of CSS codes, it is also possible to use two separate GF(2)-based belief propagation decoders. For some error $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z}$, the corresponding binary error is $\boldsymbol{e} = (\boldsymbol{e}_X^T | \boldsymbol{e}_Z^T)^T$, which yields the syndrome

$$\boldsymbol{z} = HP\boldsymbol{e} = \begin{pmatrix} \tilde{H}_X \boldsymbol{e}_Z \\ \tilde{H}_Z \boldsymbol{e}_X \end{pmatrix} = \begin{pmatrix} \boldsymbol{z}_Z \\ \boldsymbol{z}_X \end{pmatrix}. \tag{2.43}$$

Using $\boldsymbol{z}_Z$ and $\tilde{H}_X$, an estimate $\hat{\boldsymbol{e}}_Z$ of $\boldsymbol{e}_Z$ can be made using a classical binary belief propagation decoder. The same can be done with $\boldsymbol{z}_X$ and $\tilde{H}_Z$ to make an estimate $\hat{\boldsymbol{e}}_X$ of $\boldsymbol{e}_X$. The jth component of $\boldsymbol{e}_X$, denoted $e_X^{(j)}$, is equal to one if $E_j \propto X$ or $E_j \propto Y$. Therefore, $P(e_X^{(j)} = 1) = p_X + p_Y$ and similarly $P(e_Z^{(j)} = 1) = p_Y + p_Z$. These values are used as the channel error probabilities for the two decoders, which amounts to considering the quantum channel as two binary symmetric channels. Note that for depolarizing channel, $P(e_X^{(j)} = 1) = P(e_Z^{(j)} = 1) = 2p/3$, while for the XZ channel, $P(e_X^{(j)} = 1) = P(e_Z^{(j)} = 1) = 1 - \sqrt{1-p}$.

As in the classical case, belief propagation decoding can result in both detected and undetected errors. If $\hat{\boldsymbol{z}} \neq \boldsymbol{z}$, where $\hat{\boldsymbol{z}}$ is the syndrome associated with the error estimate $\hat{E}$, then a detected error has occurred. Again, these detected errors are a failing of the decoder. If $\hat{\boldsymbol{z}} = \boldsymbol{z}$ but $\tilde{\hat{E}}\tilde{E} \notin \tilde{\mathcal{S}}$, then an undetected error has occurred, which is fundamentally a failing of the code itself. It therefore follows that for the depolarizing channel, using a code with a lower distance will tend to result in a higher rate of undetected errors.

Using belief propagation in the quantum case is an even greater approximation than in the classical case. As outlined in Sec. 2.2.3, an optimal decoder for a stabilizer code will determine the most likely coset of errors rather than the single most likely error. By definition, QLDPC codes have many low-weight stabilizers, which means there will be a large number of elements of the most likely coset with similar weight and hence similar probability. This spreading of probability increases the chance that the single most likely error will not belong to the most likely coset of errors. Approximating the ideal decoder with one that determines the single most likely error will therefore lead to an increased error rate. Belief propagation goes one step further away from the optimal decoder by estimating the single most likely error in a symbol-wise fashion, which can lead to so-called symmetric degeneracy errors. Such errors are well explained by the example of Ref. [25], which is as follows. Consider a two-qubit stabilizer code with generators $M_1 = XX$ and $M_2 = ZZ$, and assume that the error $E = IX$ occurs, leading to a syndrome $\boldsymbol{z} = (0,1)^T$. The coset of errors that give this syndrome is $\{XI, IX, YZ, ZY\}$ (grouping errors up to phase). As a result, the error probabilities on both qubits are $P(E_i = I|z) = P(E_i = X|z) = Kp_I p_X$ and $P(E_i = Y|z) = P(E_i = Z|z) = Kp_Y p_Z$, where $K = 1/(2p_I p_X + 2p_Y p_Z)$. This symmetry of error probabilities results in the decoder estimating the same error on each qubit. This is not a symmetry exhibited by any of the errors that yield $\boldsymbol{z}$, and as such, even an ideal symbol-wise decoder will yield a detected error.

The requirement that all stabilizer generators must commute also degrades belief propagation performance as it results in 4-cycles. Consider some qubit j; there must be (at least) two stabilizer generators, say M_i and $M_{i'}$, that act nontrivially on j with different Pauli matrices. If this is not the case, then there will be a weight-one element of $\widetilde{N(\mathcal{S})}\backslash\tilde{\mathcal{S}}$, meaning that the code will have distance $d = 1$ (making it of little to no interest). As M_i and $M_{i'}$ contain different Pauli matrices in position j, they must also contain different Pauli matrices at some other position j' to ensure that they commute with each other. This results in a 4-cycle in the GF(4) factor graph as check nodes c_i and $c_{i'}$ both connect to error nodes v_j and $v_{j'}$. In the case of a CSS code, any 4-cycles resulting from an overlap between one row from $\tilde{H}_X$ and one row from $\tilde{H}_Z$ can be removed by decoding with a pair of GF(2) decoders rather than a GF(4) decoder. If it is a dual-containing CSS code, then there must still be 4-cycles in the GF(2) factor graph as the rows of $\tilde{H} = \tilde{H}_X = \tilde{H}_Z$ must overlap in an even number of positions to ensure that $\tilde{H}\tilde{H}^T = 0$. If the code is not dual containing, then it is possible for $\tilde{H}_X$ and $\tilde{H}_Z$ to have corresponding GF(2) factor graphs with no 4-cycles.

The reduction in 4-cycles, along with the reduced inherent complexity, makes GF(2) decoding attractive for CSS codes. However, treating a Pauli channel as a pair of binary symmetric channels ignores potential correlations between the X and Z components of an error $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z}$. These correlations are described by the conditional probabilities

$$P(e_Z^{(j)} = 1 | e_X^{(j)} = 1) = \frac{p_Y}{p_X + p_Y}, \tag{2.44}$$

$$P(e_Z^{(j)} = 1 | e_X^{(j)} = 0) = \frac{p_Z}{1 - (p_X + p_Y)}, \tag{2.45}$$

$$P(e_X^{(j)} = 1 | e_Z^{(j)} = 1) = \frac{p_Y}{p_Y + p_Z}, \tag{2.46}$$

$$P(e_X^{(j)} = 1 | e_Z^{(j)} = 0) = \frac{p_X}{1 - (p_Y + p_Z)}. \tag{2.47}$$

The X and Z components are uncorrelated if they occur independently, which requires $P(e_Z^{(j)} = 1|e_X^{(j)}) = P(e_Z^{(j)} = 1) = p_Y + p_Z$ and $P(e_X^{(j)} = 1|e_Z^{(j)}) = P(e_X^{(j)} = 1) = p_X + p_Y$. This is equivalent to the requirement that $p_Y = (p_X + p_Y)(p_Y + p_Z)$, which is satisfied by the XZ channel but not by the depolarizing channel.

2.3 Modified decoders

2.3.1 Existing decoders

2.3.1.1 Random perturbation

A number of modified decoders have been presented in Ref. [25] to address symmetric degeneracy errors. The best performing of these is the random perturbation decoder, which attempts to break decoding symmetries by randomizing the channel error probabilities. Initially, decoding is attempted using a standard GF(4) decoder. If this results in $\hat{\boldsymbol{z}} = \boldsymbol{z}$, then decoding is complete.

Otherwise, if $\hat{\boldsymbol{z}} \neq \boldsymbol{z}$, then decoding is iteratively reattempted with modified error probabilities until either decoding is successful or a maximum number of attempts N is reached. In each decoding attempt, a frustrated check is selected. This is a check node c_i such that $\hat{z}_i \neq z_i$. The channel probabilities of all qubits $j \in \mathcal{M}(i)$ involved in this check are then perturbed (up to normalization) as follows:

$$P(E_j = I) \to P(E_j = I), \tag{2.48}$$

$$P(E_j = X) \to (1 + \delta_X)P(E_j = X), \tag{2.49}$$

$$P(E_j = Y) \to (1 + \delta_Y)P(E_j = Y), \tag{2.50}$$

$$P(E_j = Z) \to (1 + \delta_Z)P(E_j = Z). \tag{2.51}$$

Here, δ_X, δ_Y, and δ_Z are realizations of a random variable that is uniformly distributed over $[0, \delta]$, where δ is called the perturbation strength. The increasing of nonidentity error probabilities is motivated by the empirical observation that the decoder is naturally too biased towards the trivial error [25].

2.3.1.2 Enhanced feedback

The enhanced feedback (EFB) decoder of Ref. [44], which is specifically tailored for the depolarizing channel, behaves somewhat similarly to the random perturbation decoder in that it also iteratively reattempts decoding with modified channel probabilities. Again, decoding is first attempted using a standard GF(4) decoder. If this results in $\hat{\boldsymbol{z}} = \boldsymbol{z}$, then decoding is complete. If instead $\hat{\boldsymbol{z}} \neq \boldsymbol{z}$, then a frustrated check c_i is selected along with an involved qubit $j \in \mathcal{M}(i)$. If $z_i = 1$ but $\hat{z}_i = 0$, then the estimated error $\hat{E}$ commutes with the stabilizer generator M_i while the error E anticommutes with M_i. To address this, the channel probabilities for E_j are adjusted such that an anticommuting error is more likely than the commuting trivial error that the decoder is naturally too biased towards. This adjustment is

$$P(E_j = \sigma) \to \begin{cases} \frac{p}{2} & \text{if } \sigma = I, \text{ or } M_i^{(j)}, \\ \frac{1-p}{2} & \text{otherwise,} \end{cases} \tag{2.52}$$

where $M_i^{(j)}$ is the jth component of the generator M_i. Conversely, if $z_i = 0$ but $\hat{z}_i = 1$, then the adjustment is

$$P(E_j = \sigma) \to \begin{cases} \frac{1-p}{2} & \text{if } \sigma = I, \text{ or } M_i^{(j)}, \\ \frac{p}{2} & \text{otherwise.} \end{cases} \tag{2.53}$$

Decoding is then reattempted with these adjusted probabilities. If this fails, then a different qubit $j \in \mathcal{M}(i)$ is selected and the process is repeated. If all qubits involved in check c_i have been exhausted and decoding is still unsuccessful, then a different check is selected and the process continues. Again, decoding is halted if a maximum number of attempts N is reached.

2.3.1.3 Supernodes

The supernode decoder of Ref. [42] is a modification of the GF(4) decoder for dual-containing CSS codes. Decoding is performed on the factor graph corresponding to $\tilde{H} = \tilde{H}_X = \tilde{H}_Z$ with checks c_i and $c_{i+m/2}$ grouped to form a single supernode. The check node calculation is modified to

$$\lambda^a_{i\to j} = K \sum_{\boldsymbol{e}:e_j=a} \delta\left[\mathrm{tr}\left(\sum_{j'\in\mathcal{M}(i)} \bar{e_{j'}}\right) = z_Z^{(i)}\right] \delta\left[\mathrm{tr}\left(\sum_{j'\in\mathcal{M}(i)} \omega\bar{e_{j'}}\right) = z_X^{(i)}\right] \prod_{j'\in\mathcal{M}(i)\setminus j} \mu^{e_{j'}}_{j'\to i}. \quad (2.54)$$

Here, $\boldsymbol{z}_Z$ contains the first $m/2$ values of $\boldsymbol{z}$ and $\boldsymbol{z}_X$ contains the last $m/2$ values; $z_Z^{(i)}$ and $z_X^{(i)}$ are the ith values of $\boldsymbol{z}_Z$ and $\boldsymbol{z}_X$, respectively. Defining $\tilde{z}_i = \omega z_Z^{(i)} + z_X^{(i)} \in \mathrm{GF}(4)$, the two constraints of Eq. (2.54) can be combined to give

$$\lambda^a_{i\to j} = K \sum_{\boldsymbol{e}:e_j=a} \delta\left(\sum_{j'\in\mathcal{M}(i)} e_{j'} = \tilde{z}_i\right) \prod_{j'\in\mathcal{M}(i)\setminus j} \mu^{e_{j'}}_{j'\to i}. \quad (2.55)$$

Note that this is of the same form as the classical check to error message given in Eq. (2.18), and it can therefore be computed using the same Fourier transform approach. The effect of combining nodes into supernodes is twofold. Firstly, it reduces decoding complexity by halving the number of check node calculations. Secondly, it can improve decoder performance as it reduces the number of 4-cycles present in the factor graph. Note that random perturbation and EFB can also be implemented using an underlying supernode decoder rather than a standard GF(4) decoder.

2.3.2 New decoders

2.3.2.1 Adjusted

The first decoder we propose is the adjusted decoder for CSS codes. This is a GF(2)-based decoder that aims to reintroduce the correlations between X and Z errors that are lost when using a standard GF(2) decoder. Initially, decoding is attempted using a standard GF(2) decoder. If this is successful, then decoding is complete. If both $H_Z\hat{\boldsymbol{e}}_X = \hat{\boldsymbol{z}}_X \neq \boldsymbol{z}_X$ and $H_X\hat{\boldsymbol{e}}_Z = \hat{\boldsymbol{z}}_Z \neq \boldsymbol{z}_Z$, then the adjusted decoder also halts. However, if one of $\hat{\boldsymbol{z}}_X = \boldsymbol{z}_X$ or $\hat{\boldsymbol{z}}_Z = \boldsymbol{z}_Z$, then we reattempt decoding for the incorrect component using channel probabilities that are adjusted according to Eqs. (2.44) to (2.47). In particular, if $\hat{\boldsymbol{z}}_X = \boldsymbol{z}_X$ but $\hat{\boldsymbol{z}}_Z \neq \boldsymbol{z}_Z$, then the adjustment is

$$P(e_Z^{(j)} = 1) \to \begin{cases} \frac{p_Y}{p_X+p_Y} & \text{if } \hat{e}_X^{(j)} = 1, \\ \frac{p_Z}{1-(p_X+p_Y)} & \text{if } \hat{e}_X^{(j)} = 0. \end{cases} \quad (2.56)$$

Alternatively, if $\hat{\boldsymbol{z}}_Z = \boldsymbol{z}_Z$ but $\hat{\boldsymbol{z}}_X \neq \boldsymbol{z}_X$, then the adjustment is

$$P(e_X^{(j)} = 1) \to \begin{cases} \frac{p_Y}{p_Y+p_Z} & \text{if } \hat{e}_Z^{(j)} = 1, \\ \frac{p_X}{1-(p_Y+p_Z)} & \text{if } \hat{e}_Z^{(j)} = 0. \end{cases} \quad (2.57)$$

We note that the adjusted decoder presented here is similar to the decoder presented for the depolarizing channel in Ref. [46]. The decoder of Ref. [46] first attempts decoding of the X component using standard channel probabilities. If this is successful, then decoding is attempted for the Z components using the modified probabilities of Eq. (2.56).

2.3.2.2 Augmented

The second decoder we propose is the augmented decoder, which was first presented in Ref. [45] for classical binary codes. An augmented decoder for QLDPC codes can be based on a GF(4) decoder for any code, a GF(2) decoder for a CSS code, or a supernode decoder for a dual-containing CSS code. The simplest of these cases is when the underlying decoder is a GF(4) decoder. In this case, decoding is initially attempted using a standard GF(4) decoder with a standard GF(4) parity-check matrix H. If this is unsuccessful, then decoding is reattempted using a randomly generated augmented parity-check matrix

$$H_A = \begin{pmatrix} H \\ H_\delta \end{pmatrix}. \tag{2.58}$$

H_δ is comprised of a subset of rows selected at random from H. The fraction of rows selected is dictated by the augmentation density δ. The syndrome used for decoding is

$$\boldsymbol{z}_A = \begin{pmatrix} \boldsymbol{z} \\ \boldsymbol{z}_\delta \end{pmatrix}, \tag{2.59}$$

where $\boldsymbol{z}$ is the measured syndrome and $\boldsymbol{z}_\delta$ contains the syndrome values corresponding to the rows selected to form H_δ. Decoding is iteratively reattempted using different augmented matrices until either decoding is successful or a maximum number of attempts N is reached. Note that duplicating rows results in a duplication of the corresponding check nodes in the factor graph.

The behavior of a supernode-based augmented decoder is very similar. In this case, the augmented parity-check matrices are of the form

$$H_A = \begin{pmatrix} \tilde{H} \\ \tilde{H}_\delta \end{pmatrix}, \tag{2.60}$$

where $\tilde{H}_\delta$ consists of the rows selected from $\tilde{H} = \tilde{H}_X = \tilde{H}_Z$. The augmented syndrome is

$$\boldsymbol{z}_A = \begin{pmatrix} \boldsymbol{z}_Z \\ \boldsymbol{z}_{Z\delta} \\ \boldsymbol{z}_X \\ \boldsymbol{z}_{X\delta} \end{pmatrix}, \tag{2.61}$$

where the values of $\boldsymbol{z}_{Z\delta}$ and $\boldsymbol{z}_{X\delta}$ are taken from $\boldsymbol{z}_Z$ and $\boldsymbol{z}_X$, respectively, according to the rows selected for repetition.

In the GF(2) case, two augmented decoders are used, one for the X component and one for the

Z component. The augmented parity-check matrices used by the X decoder are of the form

$$H_A = \begin{pmatrix} \tilde{H}_Z \\ \tilde{H}_{Z\delta} \end{pmatrix}, \tag{2.62}$$

and the augmented syndrome is

$$\boldsymbol{z}_A = \begin{pmatrix} \boldsymbol{z}_X \\ \boldsymbol{z}_{X\delta} \end{pmatrix}. \tag{2.63}$$

The syndrome and augmented parity-check matrices used by the Z decoder are of the same form.

In all three cases [GF(2), GF(4), and supernode], decoding with an augmented parity-check matrix H_A is equivalent to running a slightly altered belief propagation algorithm using the standard parity-check matrix H. We define the function r such that

$$r(i) = \begin{cases} 1 & \text{if } c_i \text{ duplicated in } H_A, \\ 0 & \text{otherwise.} \end{cases} \tag{2.64}$$

Decoding with H_A is then equivalent to decoding using H with the marginal probability approximation of Eq. (2.19) changed to

$$\hat{P}(e_j = a|\boldsymbol{z}) = KP(e_j = a) \prod_{i \in \mathcal{N}(j)} (\lambda^a_{i\to j})^{1+r(i)}, \tag{2.65}$$

and the error to check message of Eq. (2.20) changed to

$$\mu^a_{j\to i} = KP(e_j = a)(\lambda^a_{i\to j})^{r(i)} \prod_{i' \in \mathcal{N}(j)\setminus i} (\lambda^a_{i'\to j})^{1+r(i')}. \tag{2.66}$$

As a result of this equivalence, we can consider one iteration of an augmented decoder to be of the same complexity as one iteration of the underlying decoder. This formulation also gives some insight into the effect of decoding with an augmented parity-check matrix. It can be seen that duplicating a check has the effect of increasing its influence in estimating the error. Furthermore, the message $\mu_{j\to i}$ is now no longer independent of the message $\lambda_{i\to j}$ if c_i is duplicated. This amplification and feedback will alter the convergence of the marginal probability estimates. This altered convergence can help the decoder to give a different (and hopefully correct) error estimate.

2.3.2.3 Combined

The third decoder we propose combines the augmented GF(2) and adjusted decoders for CSS codes. Initially, standard GF(2) decoding is attempted. If this is successful, then decoding is complete. If both $\hat{\boldsymbol{z}}_X \neq \boldsymbol{z}_X$ and $\hat{\boldsymbol{z}}_Z \neq \boldsymbol{z}_Z$, then we reattempt decoding for the X component using augmented parity-check matrices up to N times. If this is unsuccessful, then we repeat this procedure for the Z component. If we still have $\hat{\boldsymbol{z}}_X \neq \boldsymbol{z}_X$ and $\hat{\boldsymbol{z}}_Z \neq \boldsymbol{z}_Z$, then decoding halts. However, if one of $\hat{\boldsymbol{z}}_X = \boldsymbol{z}_X$ or $\hat{\boldsymbol{z}}_Z = \boldsymbol{z}_Z$ (either from the initial decoding or after attempting decoding with augmented parity-check matrices if required), then we reattempt decoding for

the unsatisfied component with adjusted channel error probabilities as outlined in Sec. 2.3.2.1. If this is unsuccessful, then decoding for this component will be reattempted with augmented parity-check matrices up to N times using the same adjusted probabilities.

2.4 Simulation results

2.4.1 Bicycle

The first code we have considered is a $[[400, 200]]$ bicycle code of Ref. [23]. Bicycle codes are dual-containing CSS codes that are constructed by first generating an $n/2 \times n/2$ binary circulant matrix A with row weight $w/2$. A is used to define the $n/2 \times n$ matrix $H_0 = [\ A \quad A^T\ ]$ from which $(n-m)/2$ rows are removed to give $\tilde{H}$ (following the heuristic that column weight should be kept as uniform as possible). Taking $\tilde{H}_X = \tilde{H}_Z = \tilde{H}$ defines the GF(2) and GF(4) parity-check matrices according to Eqs. (2.36) and (2.39), respectively. The associated stabilizer code will have $k \geq n-m$, with equality when the parity-check matrix is full rank (this is the case for our code). Removing rows from H_0 corresponds to removing stabilizer generators of weight w. Unless a removed row belongs to the span of the remaining rows, which is unlikely, the removed generator will be in $\widetilde{N(\mathcal{S})} \backslash \tilde{\mathcal{S}}$. A bicycle code's distance is therefore upper bounded by w (we have chosen $w = 20$ for our code).

2.4.1.1 Depolarizing channel

We first consider the depolarizing channel. Both the augmented and random perturbation decoders have a tunable parameter δ, which controls the augmentation density and perturbation strength, respectively. As shown for classical codes in Ref. [45], this δ value can have a significant impact on the performance of an augmented decoder. We observe the same behavior for both augmented and random perturbation decoders in the quantum case as shown in Fig. 2.2. Here, decoders with $N = 10$ maximum decoding attempts and varying δ have been tested at four different depolarizing probabilities (we use a maximum of $I_{\max} = 100$ iterations per attempt for every decoder in this paper). The vertical axis gives normalized FER (frame error rate), which is the modified decoder's FER divided by the underlying (standard) decoder's FER. Note that each data point in this figure, as well as all other figures presented in this paper, corresponds to at least 100 decoding errors. Based on these results, we have selected values of $\delta = 0.1$ for the augmented GF(2) decoder, $\delta = 0.15$ for the augmented GF(4) and supernode decoders, $\delta = 100$ for the random perturbation GF(4) decoder, and $\delta = 200$ for the random perturbation supernode decoder. Note that the δ value we use for the combined decoder is always the same as the value used for the augmented GF(2) decoder.

We have tested all of the decoders outlined in Sec. 2.3 on this code. The random perturbation, EFB, augmented, and combined decoders all use $N = 100$ attempts. The FER performance of these decoders is shown in Fig. 2.3, and the average number of iterations required by each of

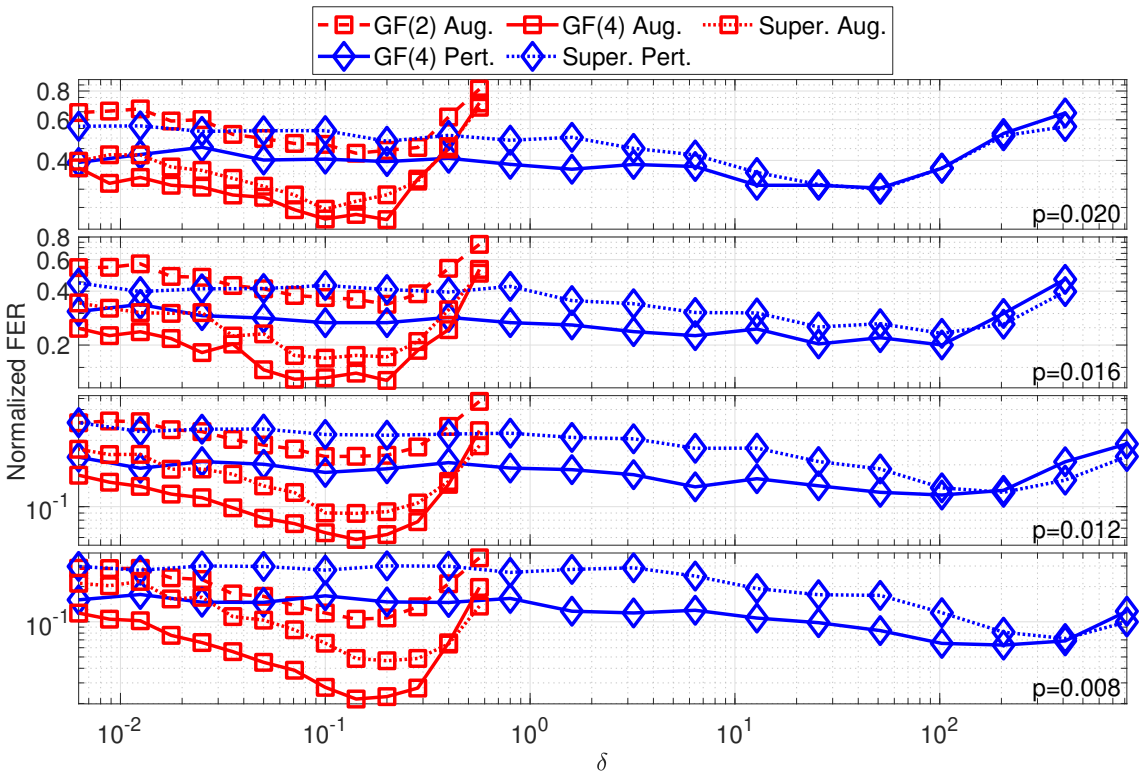

Figure 2.2: The effect of augmentation density and random perturbation strength on decoder performance (frame error rate) for the $[[400, 200]]$ bicycle code on the depolarizing channel. Each decoder uses $N = 10$ maximum attempts.

them is shown in Fig. 2.4. It can be seen that the standard supernode decoder outperforms the standard GF(4) decoder, which in turn, outperforms the standard GF(2) decoder. Furthermore, the supernode decoder requires fewer iterations on average than the standard GF(4) or GF(2) decoders [the number of iterations used by a GF(2)-based decoder is taken to be the number used by one of the two constituent decoders]. However, note that comparing the number of iterations used by these different decoders, or indeed modified decoders based on different underlying decoders, is not particularly meaningful as their iterations are of differing complexity. The adjusted decoder can be seen to give a FER similar to the standard supernode decoder at the cost of a negligible increase in required iterations compared to the standard GF(2) decoder. This FER performance suggests that the adjusted decoder is successful in reintroducing the correlation between the X and Z error components. The random perturbation and EFB decoders based on either GF(4) or supernode decoders have similar FER performance and require a near-identical number of iterations on average. The augmented GF(4) and supernode decoders outperform both the random perturbation and EFB decoders while requiring a lower number of iterations on average. The augmented GF(2) decoder does give a reasonable FER reduction compared to the standard GF(2) decoder, but it is outperformed by all modified GF(4) and supernode decoders. However, the combined decoder gives a FER lower than the random perturbation and EFB decoders. Furthermore, it also requires fewer iterations on average than the augmented GF(2) decoder. All of the decoding errors we have observed for this code are detected errors; that is, they are due to a failing of the decoder rather than the code's distance.

Figure 2.5 shows the effect of the maximum number of decoding attempts on the performance of

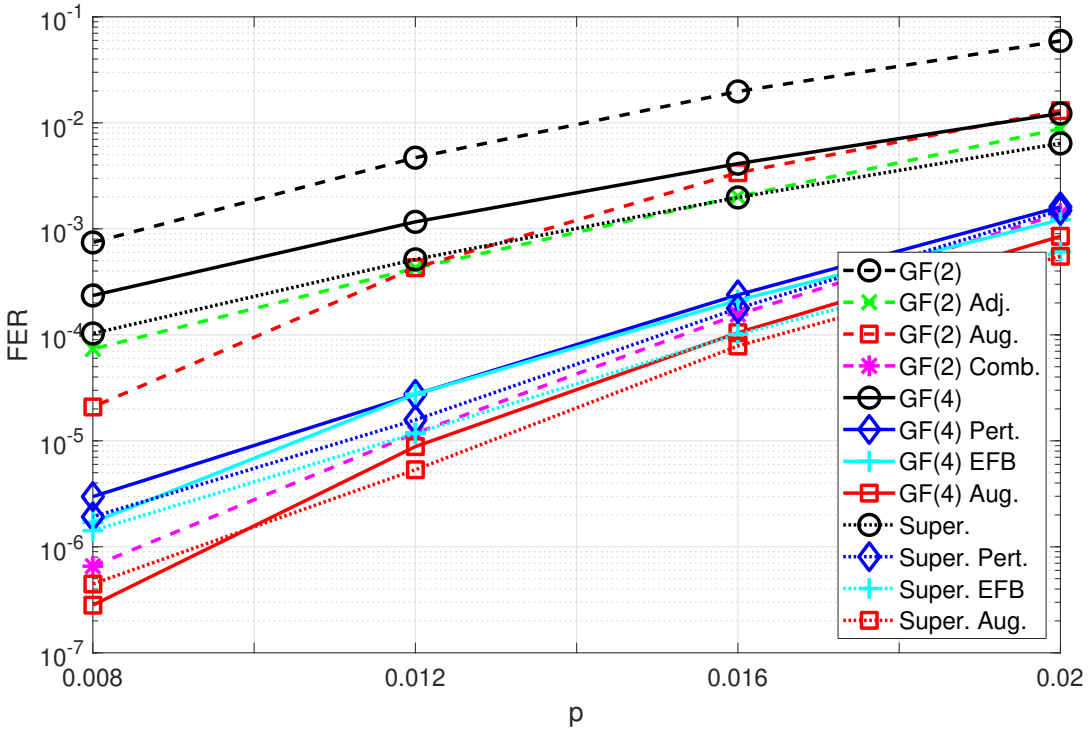

Figure 2.3: FER performance of decoders with $N = 100$ attempts (where applicable) for the $[[400, 200]]$ bicycle code on the depolarizing channel.

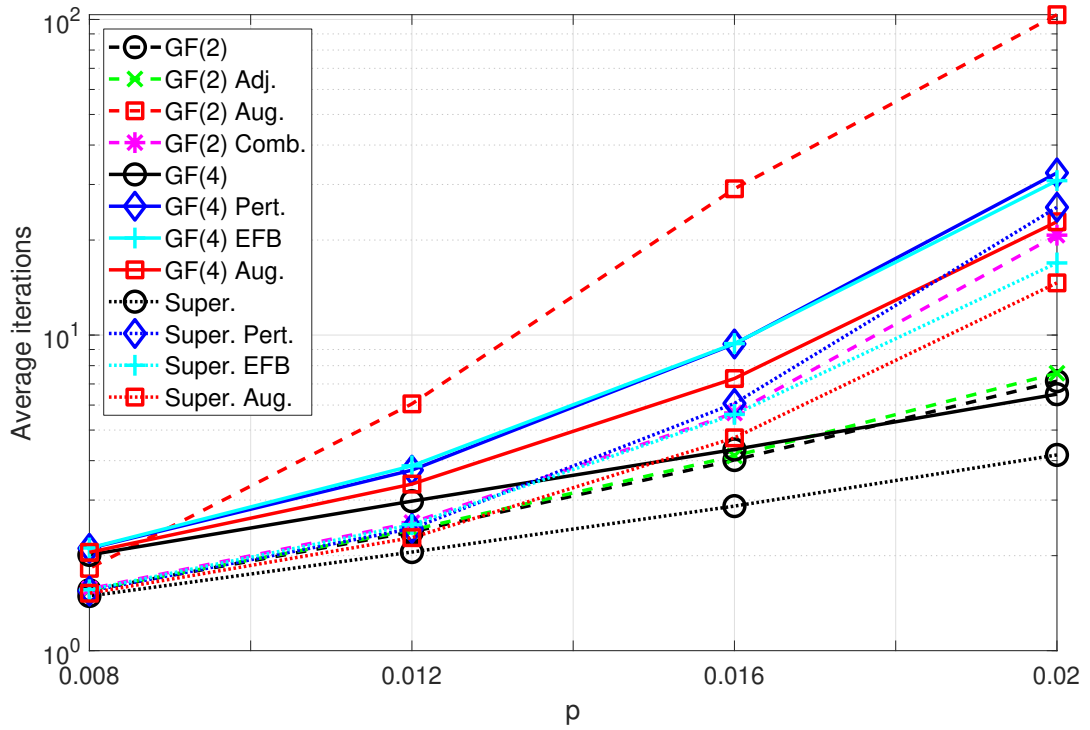

Figure 2.4: Average number of iterations required by decoders with $N = 100$ attempts (where applicable) for the $[[400, 200]]$ bicycle code on the depolarizing channel.

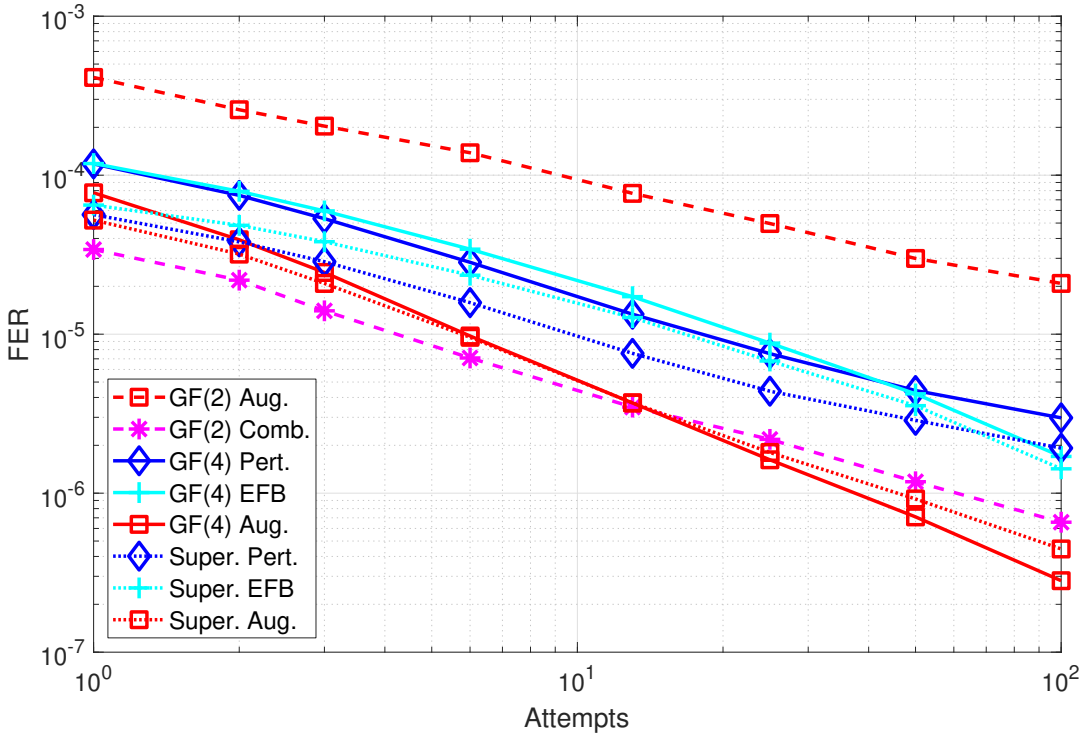

Figure 2.5: FER performance of decoders at $p = 0.008$ with a varying number of decoding attempts for the $[[400, 200]]$ bicycle code on the depolarizing channel.

the augmented, combined, random perturbation, and EFB decoders at a depolarizing probability of $p = 0.008$. For all decoders, the FER reduction with an increasing maximum number of attempts is approximately linear on a log-log plot. This suggests that we could continue to reduce the FER by increasing the maximum number of attempts beyond $N = 100$. It can be seen that the augmented and combined decoders only require approximately $N = 25$ maximum attempts to match the performance of random perturbation and EFB decoders with $N = 100$.

2.4.1.2 XZ channel

To isolate the effect of augmentation in the GF(2) case, we have repeated the analysis of the previous section for the XZ channel. As previously noted, the X and Z error components occur independently for this channel; therefore, there are no correlations to be ignored when using a GF(2)-based decoder. As a result, the adjusted and combined decoders will give no performance increase over the standard GF(2) and augmented GF(2) decoders, respectively. While we have still employed the random perturbation decoder for comparison on this channel, we have not used the EFB decoder as it is specifically tailored to the depolarizing channel.

Again, we first tune the augmentation density and random perturbation strength using decoders with $N = 10$ as shown in Fig. 2.6. It can be seen that the optimal value of δ is essentially independent of the underlying decoder. As such, we have selected a value of $\delta = 0.15$ for all augmented decoders and $\delta = 100$ for both of the random perturbation decoders. Note that these are the same values we have used for the GF(4)-based decoders in the depolarizing case.

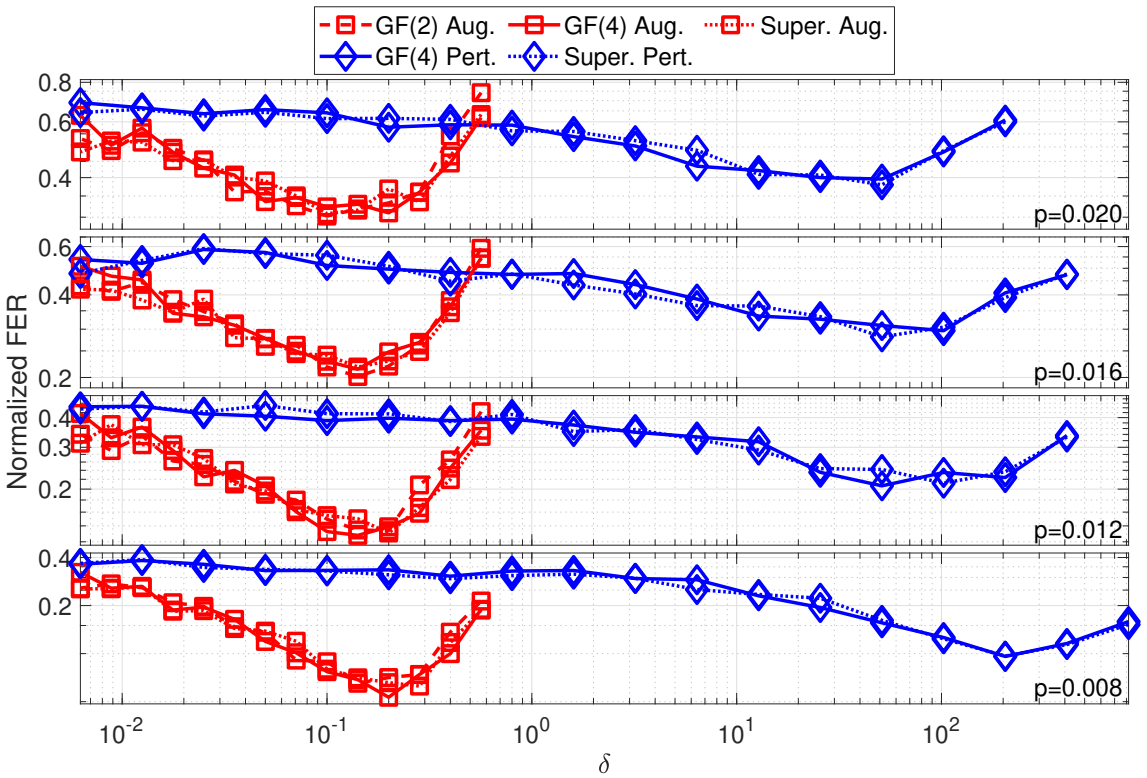

Figure 2.6: The effect of augmentation density and random perturbation strength on decoder performance for the $[[400, 200]]$ bicycle code on the XZ channel. Each decoder uses $N = 10$ maximum attempts.

The FER performance and average required iterations for decoders with $N = 100$ maximum attempts are shown in Figs. 2.7 and 2.8, respectively. It can be seen that the standard GF(2), GF(4), and supernode decoders all exhibit near-identical performance on the XZ channel. That the GF(2) and supernode decoders yield the same FER is unsurprising and is consistent with the similar performance of the adjusted and supernode decoders on the depolarizing channel. The performance of the GF(4) decoder suggests that the 4-cycles involving one row from $\tilde{H}_X$ and one row from $\tilde{H}_Z$ have no effect on decoding performance when the error components are independent. The performance of the augmented and random perturbation decoders is also largely independent of the underlying decoder. Furthermore, the relative performance of the decoders is very similar to that observed for the GF(4)-based decoders in the depolarizing case, with the augmented decoders outperforming the random perturbation decoders.

The effect of the maximum number of decoding attempts on decoder performance is shown for $p = 0.008$ in Fig. 2.9. Unsurprisingly, the performance of the augmented and random perturbation decoders remains largely independent of the underlying decoder over the range of N values tested. Furthermore, the relative performance is very similar to that exhibited by the GF(4)-based decoders in the depolarizing case, with the augmented decoders only requiring approximately $N = 25$ maximum attempts to match the performance of the random perturbation decoders with $N = 100$.

We have tested the performance of decoders on the XZ channel for all four CSS codes considered in this paper. However, we omit the results for the other three codes as they all follow the same

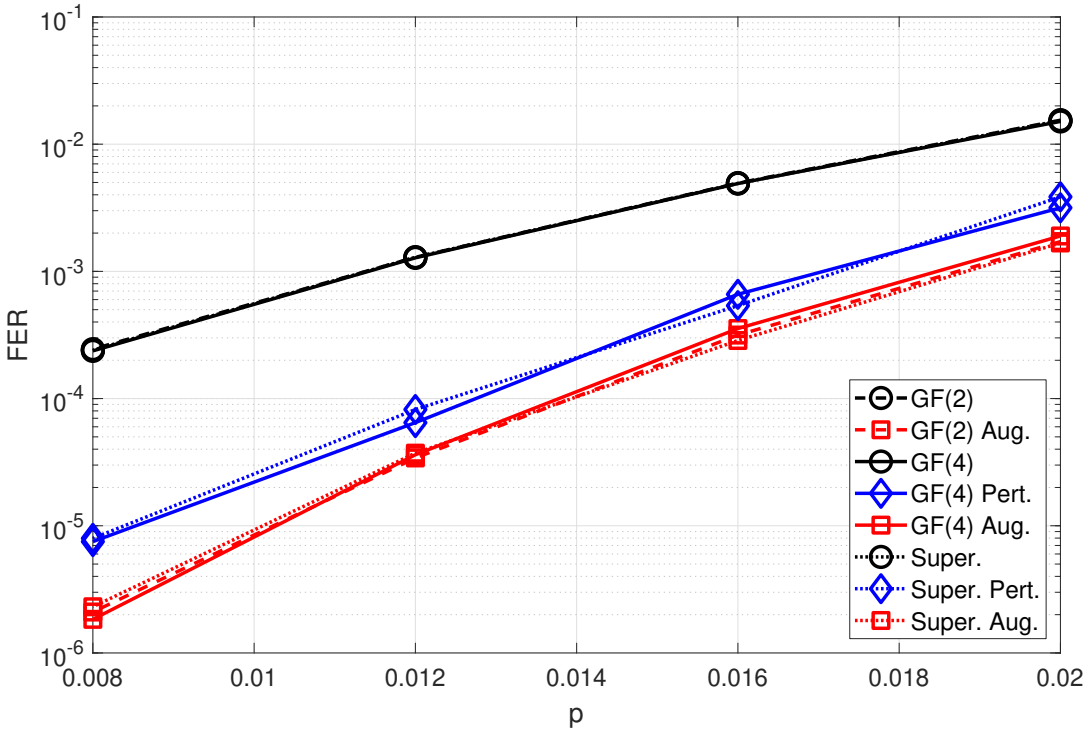

Figure 2.7: FER performance of decoders with $N = 100$ attempts (where applicable) for the $[[400, 200]]$ bicycle code on the XZ channel.

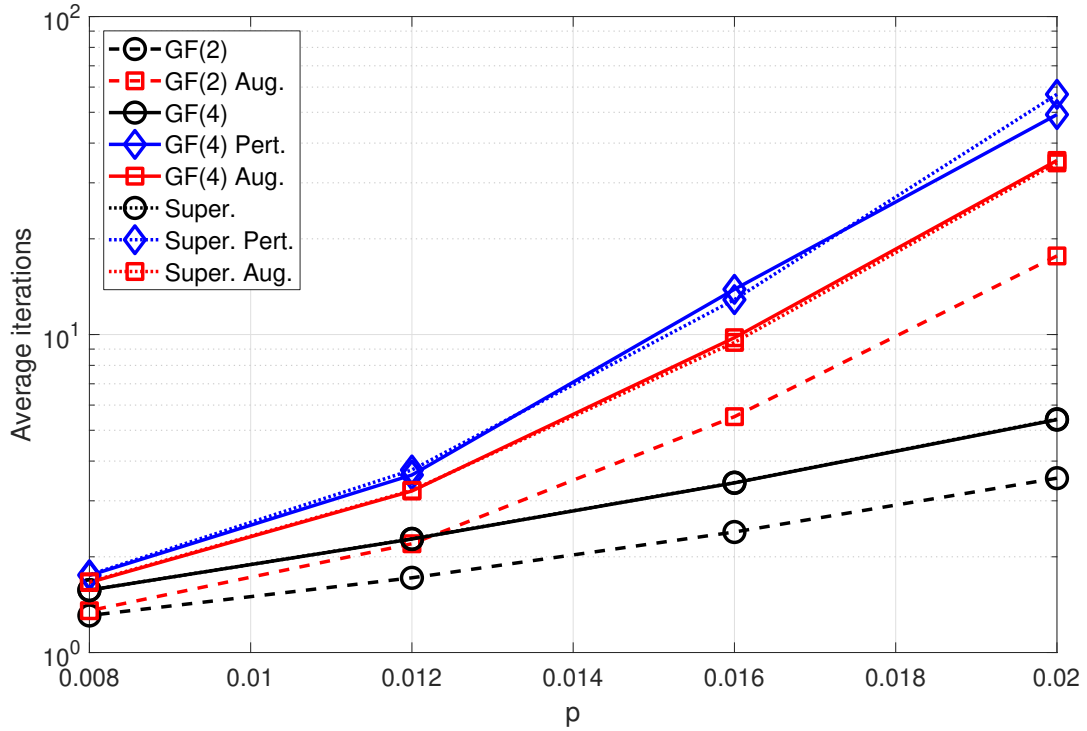

Figure 2.8: Average number of iterations required by decoders with $N = 100$ attempts (where applicable) for the $[[400, 200]]$ bicycle code on the XZ channel.

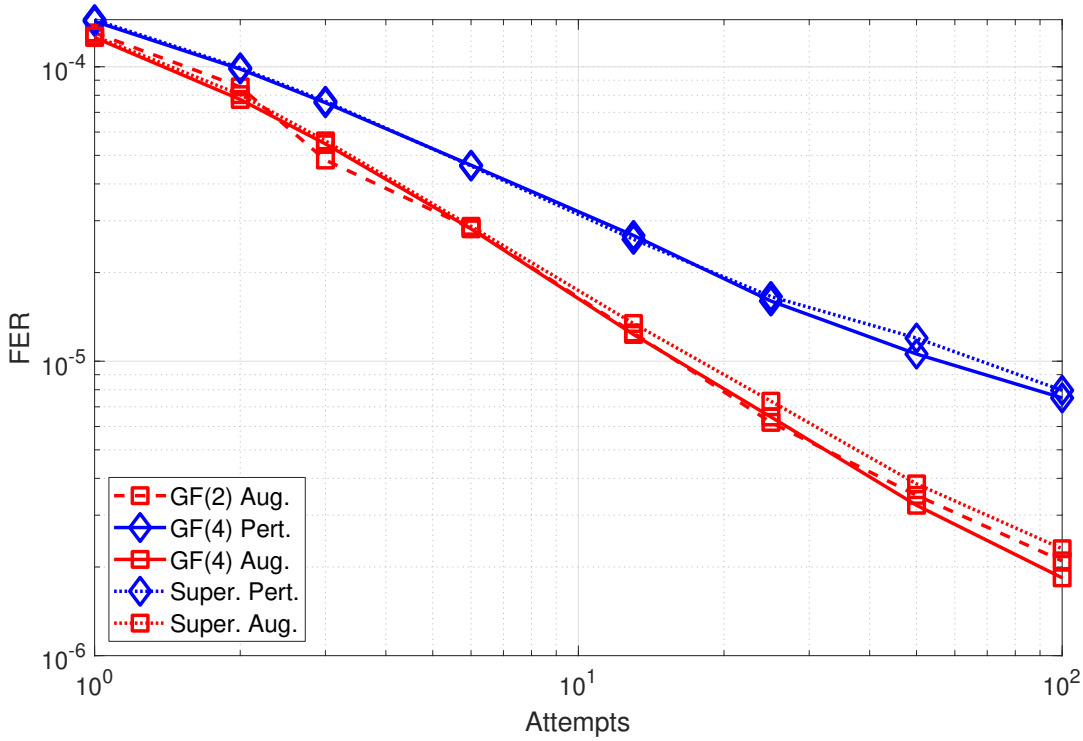

Figure 2.9: FER performance of decoders at $p = 0.008$ with a varying number of decoding attempts for the $[[400, 200]]$ bicycle code on the XZ channel.

trend outlined here. That is, the performance of decoders is essentially independent of the underlying decoder, and the relative performance of the augmented and random perturbation decoders is very similar to that exhibited by the GF(4)-based decoders in the depolarizing case.

2.4.2 BIBD

The second code we have considered is a $[[610, 490]]$ balanced incomplete block design (BIBD) code from Ref. [54]. Like the bicycle code, this is also a dual-containing CSS code. A BIBD $(X, \mathcal{B})$, where $X = \{x_1, \ldots, x_v\}$ and $\mathcal{B} = \{B_1, \ldots, B_b\}$, is a collection of b subsets (blocks) of size k that are drawn from a set X containing v elements. Each pair of elements occurs in λ of the blocks, and every element occurs in r blocks. The $v \times b$ GF(2) incidence matrix A of $(X, \mathcal{B})$ has elements

$$A_{ij} = \begin{cases} 1 & x_i \in B_j, \\ 0 & x_i \notin B_j. \end{cases} \tag{2.67}$$

If λ is even, then A will satisfy $AA^T = 0$ as any two rows will overlap an even number of times. As such, taking $\tilde{H} = \tilde{H}_X = \tilde{H}_Z = A$ defines a dual-containing CSS code. The BIBD that we have selected follows the construction of Ref. [55]. If $6t + 1$ is a prime or prime power and α is a primitive element of GF$(6t + 1)$, then a BIBD $(\text{GF}(6t + 1), \mathcal{B})$ can be constructed with $v = 6t + 1$, $b = t(6t + 1)$, $r = 4t$, $k = 4$, and $\lambda = 2$. To do this, t base blocks $\tilde{B}_i$ are constructed for $0 \leq i \leq t - 1$ with

$$\tilde{B}_i = \{0, \alpha^i, \alpha^{2t+i}, \alpha^{4t+i}\}. \tag{2.68}$$

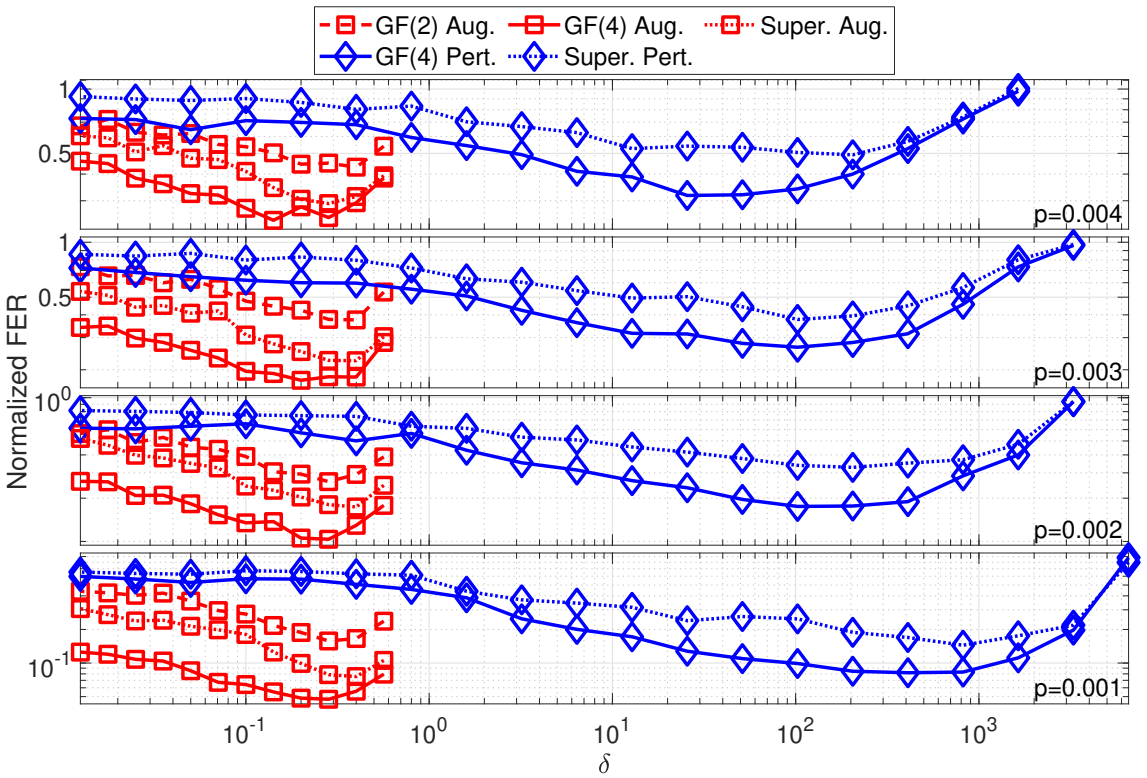

Figure 2.10: The effect of augmentation density and random perturbation strength on decoder performance for the $[[610, 490]]$ BIBD code on the depolarizing channel. Each decoder uses $N = 10$ maximum attempts.

$6t+1$ blocks of the form $\tilde{B}_i + \beta = \{\beta, \alpha^i + \beta, \alpha^{2t+i} + \beta, \alpha^{4t+i} + \beta\}$, where $\beta \in \mathrm{GF}(6t+1)$, can then be constructed from each base block. This gives a total of $t(6t+1)$ blocks and a corresponding incidence matrix of the form

$$\tilde{H} = A = \begin{pmatrix} A_1 & A_2 & \cdots & A_t \end{pmatrix}. \tag{2.69}$$

Here, each A_i is a $(6t+1) \times (6t+1)$ circulant matrix of weight $k = 4$. We have selected $t = 10$ and $\alpha = 2$ for our code.

The results presented for this code and all codes that follow are on the depolarizing channel. The effect of augmentation density and random perturbation strength for decoders with $N = 10$ on this code is shown in Fig. 2.10. Based on these results, we have selected values of $\delta = 0.3$ for all augmented decoders, $\delta = 200$ for the random perturbation GF(4) decoder, and $\delta = 400$ for the random perturbation supernode decoder.

The FER performance and average required iterations for decoders with $N = 100$ maximum attempts are shown in Figs. 2.11 and 2.12, respectively. The results here are quite similar to those for the bicycle code. Again, the adjusted decoder gives performance similar to that of the supernode decoder. Furthermore, the random perturbation and EFB decoders perform similarly to one another. The augmented GF(2) decoder is outperformed by all modified GF(4) and supernode decoders. The combined, augmented GF(4), and augmented supernode decoders again outperform the random perturbation and EFB decoders. Overall, there is less spread in the performance of the decoders on this BIBD code. This can be attributed to the fact that a

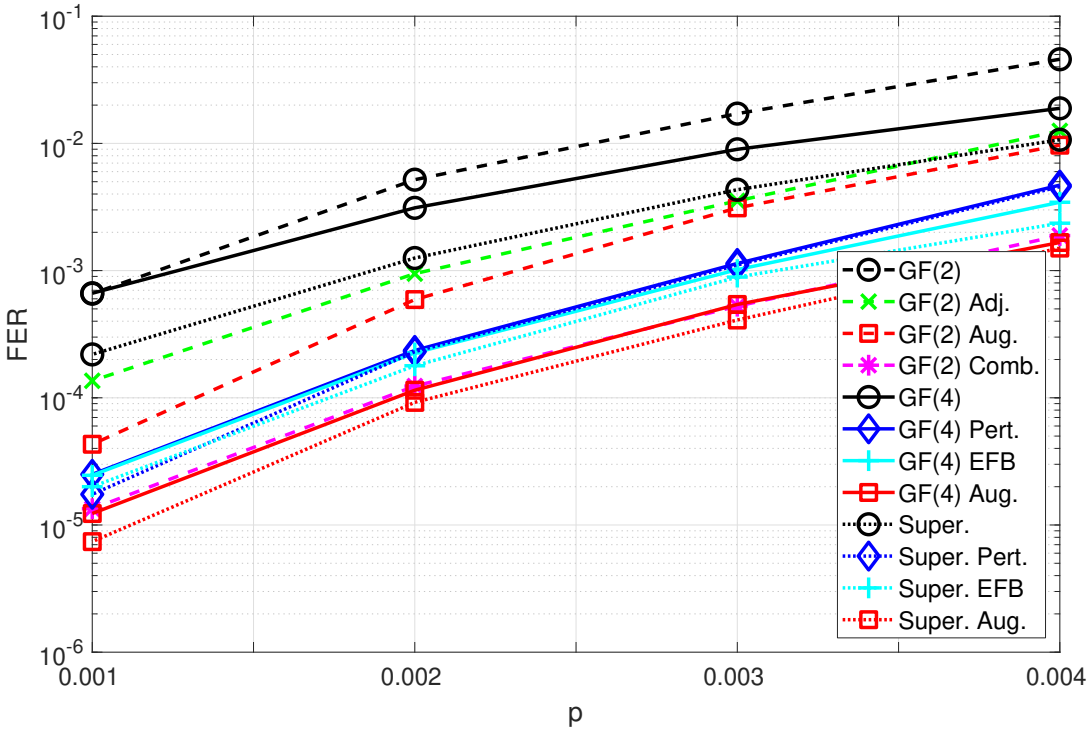

Figure 2.11: FER performance of decoders with $N = 100$ attempts (where applicable) for the $[[610, 490]]$ BIBD code on the depolarizing channel.

large fraction of decoding errors are undetected. For example, approximately 65% of the errors exhibited by the augmented supernode decoder at $p = 0.001$ are undetected. This abundance of undetected errors suggests that decoding is being limited by the code's distance $d \leq 5$ [this value is based on the lowest-weight element of $\widetilde{N(\mathcal{S})}\backslash\tilde{\mathcal{S}}$ that we have observed].

The effect of these undetected errors can also be seen in Fig. 2.13. For the bicycle code, the reduction in FER with increasing maximum number of iterations was approximately linear on a log-log plot. However, the reduction in FER for the BIBD code can be seen to taper off; that is, increasing the maximum number of attempts has diminishing returns. Partially as a result of this, we only require approximately $N = 10$ maximum attempts for our augmented supernode decoder to match the performance of the random perturbation and EFB decoders with $N = 100$.

2.4.3 Quasicyclic

The third code we have considered is a $[[506, 240]]$ quasicyclic code from Ref. [56]. Unlike the first two codes, this is a nondual-containing CSS code. The parity-check submatrices $\tilde{H}_X$ and $\tilde{H}_Z$ can be defined in terms of base matrices $\mathcal{H}_X$ and $\mathcal{H}_Z$, respectively, whose elements belong to the set $\{0, 1, \ldots, P-1\}$. $\tilde{H}_X$ ($\tilde{H}_Z$) is then constructed by replacing each element of $\mathcal{H}_X$ ($\mathcal{H}_Z$) with a $P \times P$ identity matrix shifted circularly to the right by an amount given by the replaced element. The base matrix construction of Ref. [56] gives a parity-check matrix that satisfies $\tilde{H}_Z \tilde{H}_X^T = 0$; it also ensures that the factor graphs associated with $\tilde{H}_X$ and $\tilde{H}_Z$ are free of 4-cycles. These base matrices are constructed from a so-called "perfume" (perfect fulfillment). Let $\mathbb{Z}_P$ be the set of integers $\{0, 1, \ldots, P-1\}$ with addition, subtraction, and multiplication

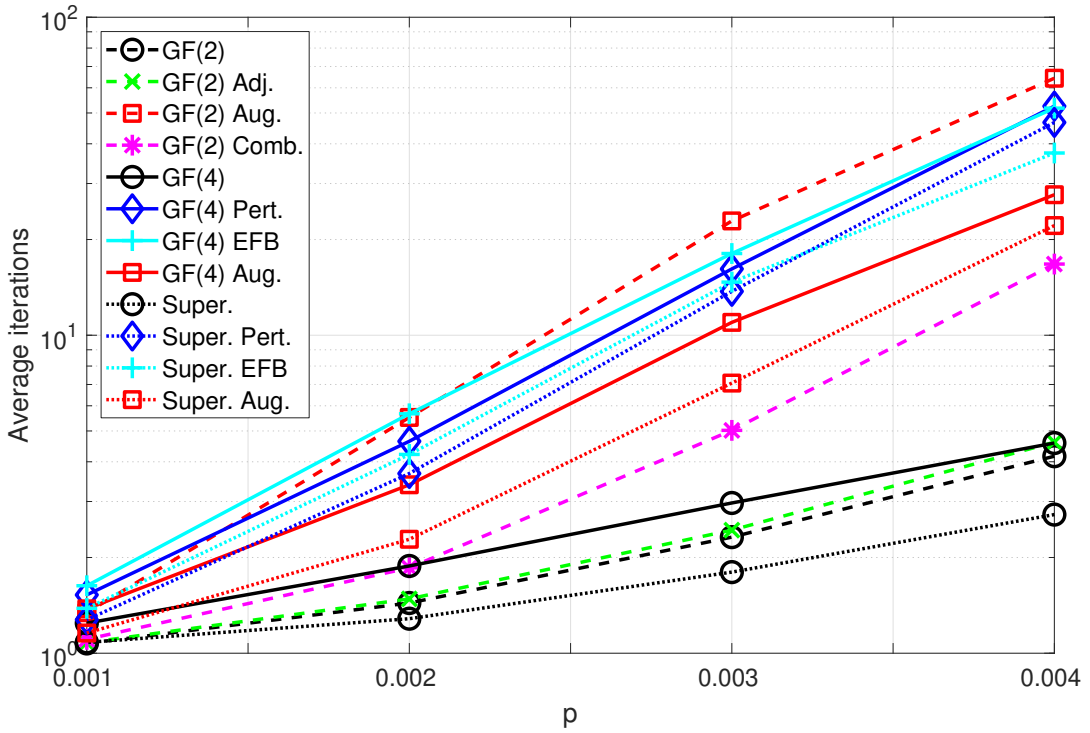

Figure 2.12: Average number of iterations required by decoders with $N = 100$ attempts (where applicable) for the $[[610, 490]]$ BIBD code on the depolarizing channel.

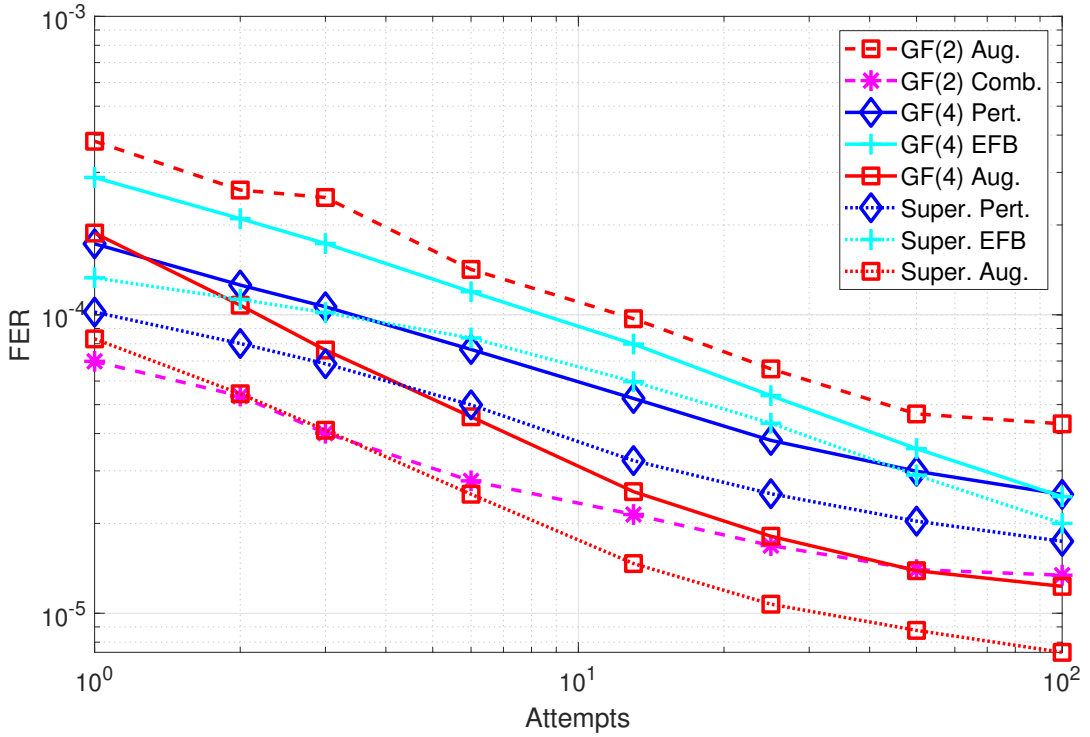

Figure 2.13: FER performance of decoders at $p = 0.001$ with a varying number of decoding attempts for the $[[610, 490]]$ BIBD code on the depolarizing channel.

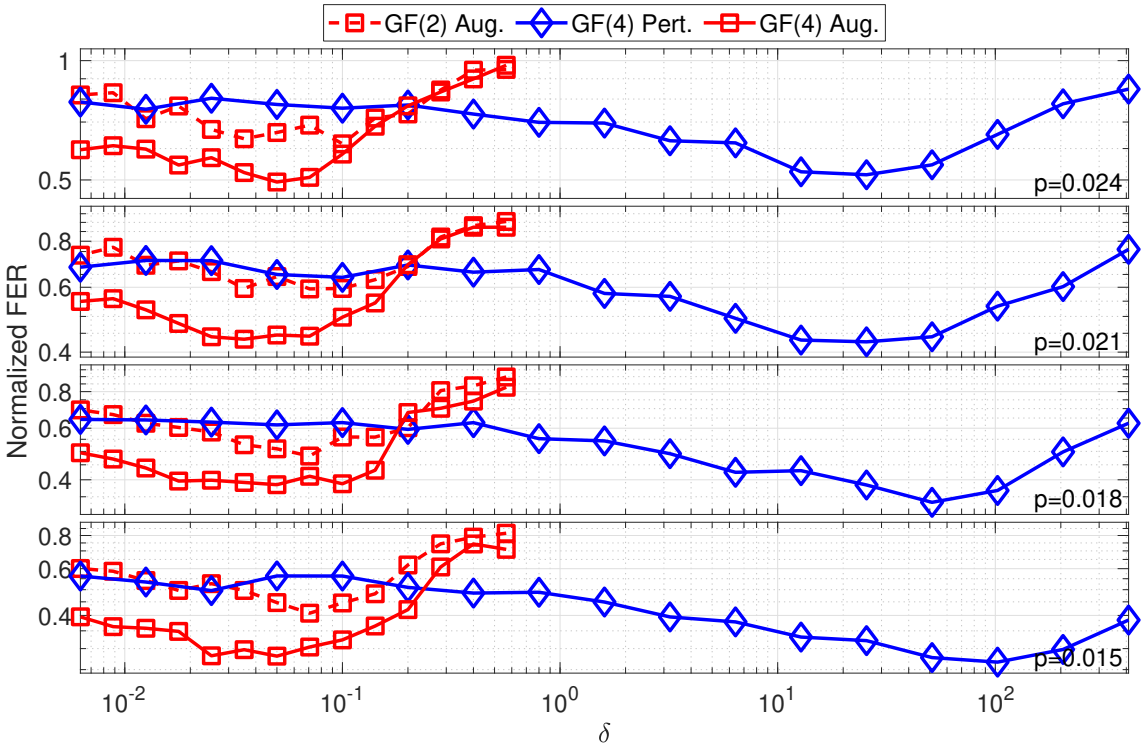

Figure 2.14: The effect of augmentation density and random perturbation strength on decoder performance for the $[[506, 240]]$ quasicyclic code on the depolarizing channel. Each decoder uses $N = 10$ maximum attempts.

modulo P. $\mathbb{Z}_P^*$ is then the abelian multiplicative group $\mathbb{Z}_P^* = \{z \in \mathbb{Z}_P : \gcd(z, P) = 1\}$. For positive integers P and σ, σ is a fulfillment of P if σ is coprime to P and $1 - \sigma^i$ is coprime to P for $1 \leq i < \mathrm{ord}(\sigma)$. Here, $\mathrm{ord}(\sigma)$ is the order of σ in $\mathbb{Z}_P^*$. A triple of positive integers (P, σ, τ) is a perfume if σ is a fulfillment of P, τ is coprime to P, and $\tau \notin \{\sigma, \sigma^2, \ldots, \sigma^{\mathrm{ord}(\sigma)}\}$. Letting $L = 2\mathrm{ord}(\sigma)$, we define

$$c_{jl} = \begin{cases} \sigma^{-j+l} & \text{if } 0 \leq l < \frac{L}{2}, \\ \tau\sigma^{-j+l} & \text{if } \frac{L}{2} \leq l \leq L, \end{cases} \tag{2.70}$$

$$d_{kl} = \begin{cases} -\tau\sigma^{k-l} & \text{if } 0 \leq l < \frac{L}{2}, \\ -\sigma^{k-l} & \text{if } \frac{L}{2} \leq l \leq L. \end{cases} \tag{2.71}$$

Indexing from zero, these are the elements of the $J \times L$ and $K \times L$ base matrices $\mathcal{H}_X$ and $\mathcal{H}_Z$, respectively, where $1 \leq J, K \leq L/2$. To construct our code, we have used the perfume $(23, 8, 20)$ (this gives $L = 22$) and have chosen $J = K = 6$.

The effect of augmentation density and random perturbation strength for decoders with $N = 10$ on this code is shown in Fig. 2.14. Note that for this code, we can only use GF(2)- and GF(4)-based decoders as it is not dual containing. Based on these results, we have selected values of $\delta = 0.07$ for the augmented GF(2) decoder, $\delta = 0.05$ for the augmented GF(4) decoder, and $\delta = 50$ for the random perturbation GF(4) decoder.

The FER performance and average required iterations for decoders with $N = 100$ maximum

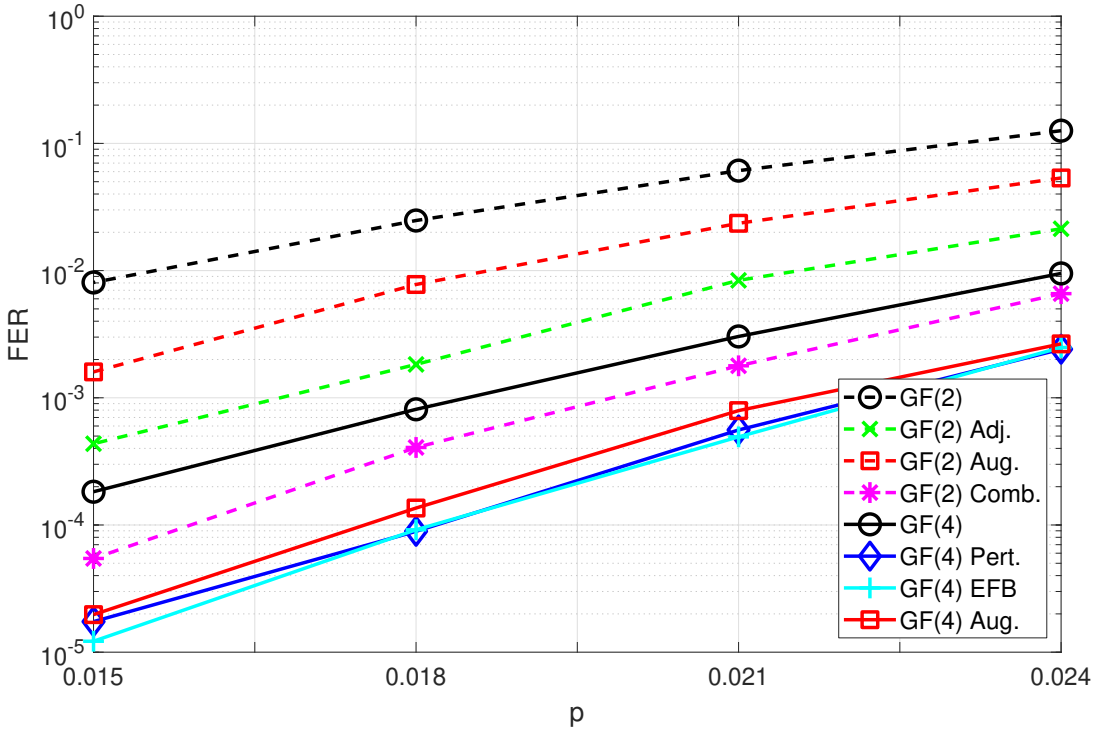

Figure 2.15: FER performance of decoders with $N = 100$ attempts (where applicable) for the $[[506, 240]]$ quasicyclic code on the depolarizing channel.

attempts are shown in Figs. 2.15 and 2.16, respectively. On the previous two codes, the augmented GF(2) decoder gave a similar or lower FER than the adjusted decoder. This is not the case here, with the adjusted decoder giving a significantly lower FER. This suggests that the augmented decoder has some effect in alleviating the effect of 4-cycles in the code's factor graph [none of which are present when using a GF(2) decoder for this code]. The random perturbation, EFB, and augmented GF(4) decoders all perform similarly on this code. The combined decoder performs worse than the modified GF(4) decoders.

Like the bicycle code, all decoding errors observed for this code were detected errors. This is reflected in Fig. 2.17, which shows an approximately linear reduction in FER with an increasing number of maximum attempts on a log-log plot for all decoders considered.

2.4.4 Bicyclelike

The fourth code we have considered is a $[[400, 200]]$ nondual-containing CSS code based on the bicyclelike construction of Ref. [57]. The codes of Ref. [57] are constructed using a BIBD in a similar way to the code of Sec. 2.4.2. $\tilde{H}_X$ is constructed by taking the first a (where a is even) submatrices of the BIBD's adjacency matrix as given in Eq. (2.69); that is,

$$\tilde{H}_X = \begin{pmatrix} A_1 & A_2 & \cdots & A_a \end{pmatrix}. \tag{2.72}$$

$\tilde{H}_Z$ is then a cyclically shifted version of $\tilde{H}_X$, with

$$\tilde{H}_Z = \begin{pmatrix} A_{\frac{a}{2}+1} & A_{\frac{a}{2}+2} & \cdots & A_a & A_1 & A_2 & \cdots & A_{\frac{a}{2}} \end{pmatrix}. \tag{2.73}$$

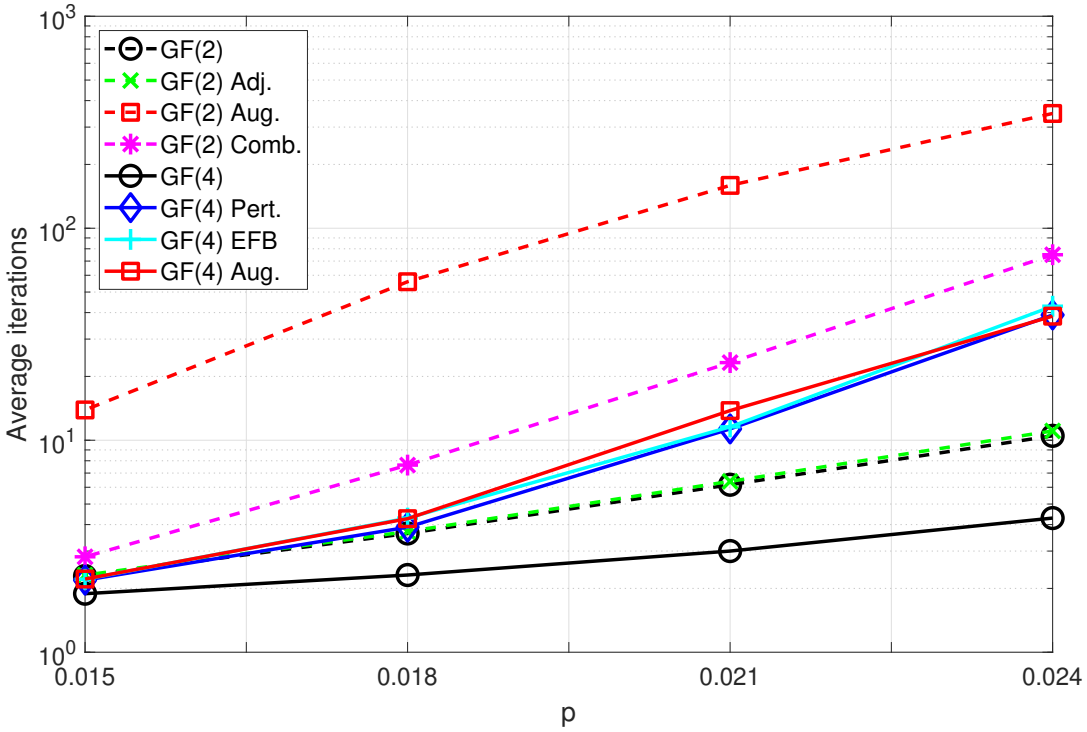

Figure 2.16: Average number of iterations required by decoders with $N = 100$ attempts (where applicable) for the $[[506, 240]]$ quasicyclic code on the depolarizing channel.

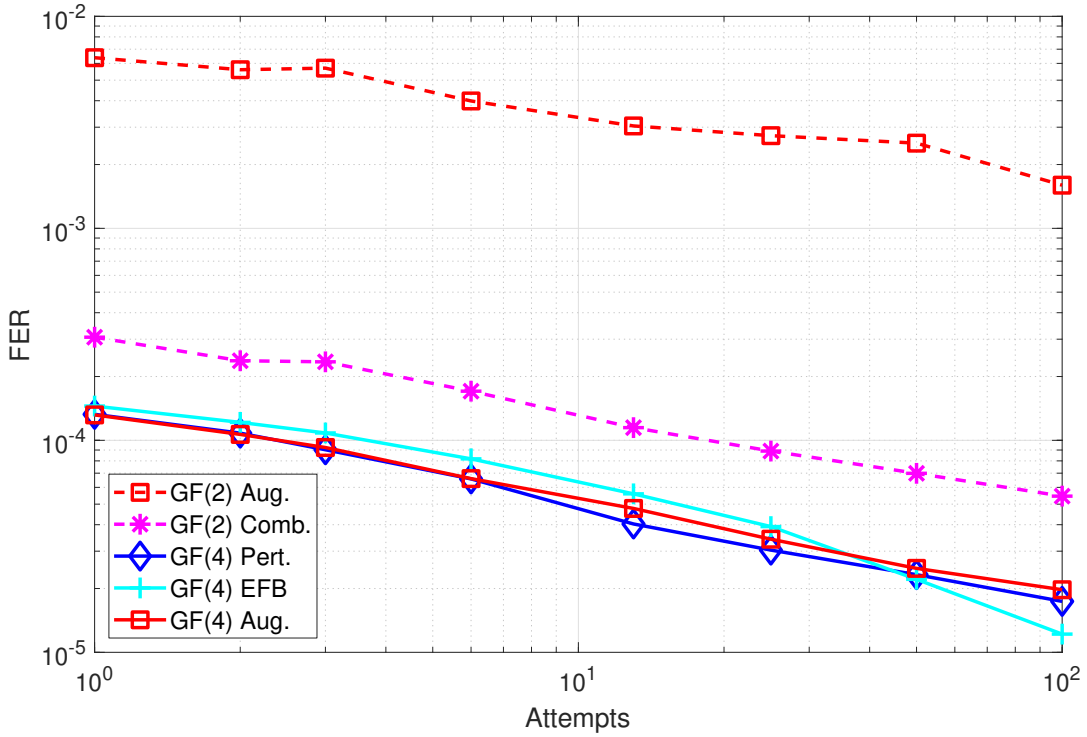

Figure 2.17: FER performance of decoders at $p = 0.015$ with a varying number of decoding attempts for the $[[506, 240]]$ quasicyclic code on the depolarizing channel.

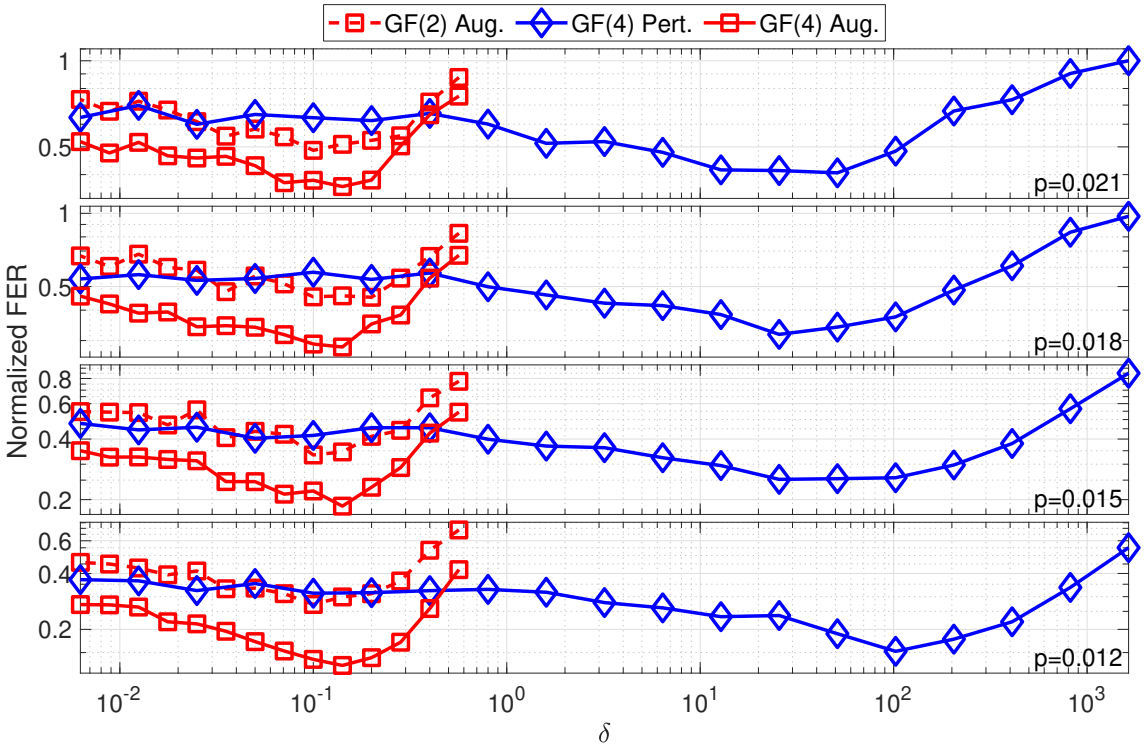

Figure 2.18: The effect of augmentation density and random perturbation strength on decoder performance for the $[[400, 200]]$ bicyclelike code on the depolarizing channel. Each decoder uses $N = 10$ maximum attempts.

The use of a BIBD with $\lambda = 1$ ensures that $\tilde{H}_X$ and $\tilde{H}_Z$ are both free of 4-cycles. However, we have observed that codes constructed in this way have low distances and are therefore not appropriate for comparing decoders. We have found that this distance can be increased by generalizing the construction to allow the circulant matrices $A_1, \ldots, A_a$ to be randomly generated. Note that this comes at the expense of introducing 4-cycles. For our code, we have constructed $\tilde{H}_X$ from four 100×100 circulant matrices of weight five. Each of $\tilde{H}_X$ and $\tilde{H}_Z$ yield factor graphs with 1 700 4-cycles, compared to the 2 737 4-cycles of the bicycle code considered in Sec. 2.4.1.

The effect of augmentation density and random perturbation strength for decoders with $N = 10$ on this code is shown in Fig. 2.18. Based on these results, we have selected values of $\delta = 0.1$ for the augmented GF(2) decoder, $\delta = 0.15$ for the augmented GF(4) decoder, and $\delta = 100$ for the random perturbation GF(4) decoder.

The FER performance and average required iterations for decoders with $N = 100$ maximum attempts are shown in Figs. 2.19 and 2.20, respectively. Again, the adjusted decoder outperforms the augmented GF(2) decoder; however, the gap in their performance is smaller than for the quasicyclic code of Sec. 2.4.3. The EFB and augmented GF(4) decoders perform similarly on this code, both outperforming the random perturbation decoder. The combined decoder is again outperformed by all modified GF(4) decoders, although the performance gap is smaller than in the quasicyclic case.

While our modified construction gives a higher distance than the codes presented in Ref. [57],

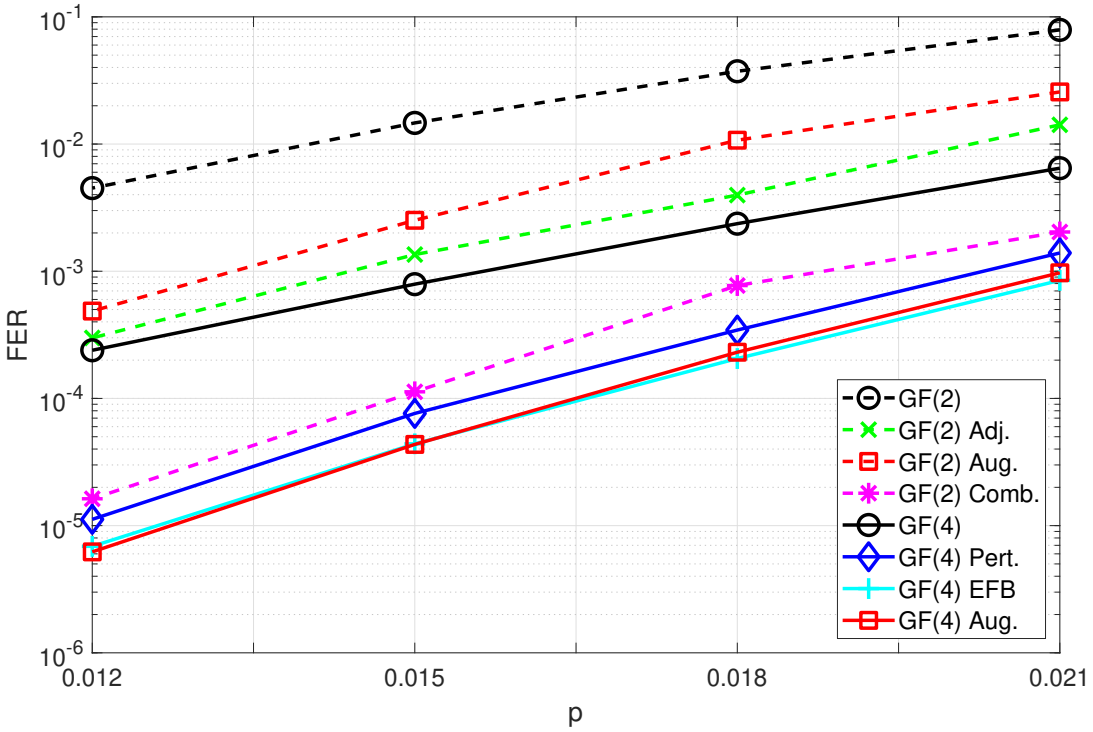

Figure 2.19: FER performance of decoders with $N = 100$ attempts (where applicable) for the $[[400, 200]]$ bicyclelike code on the depolarizing channel.

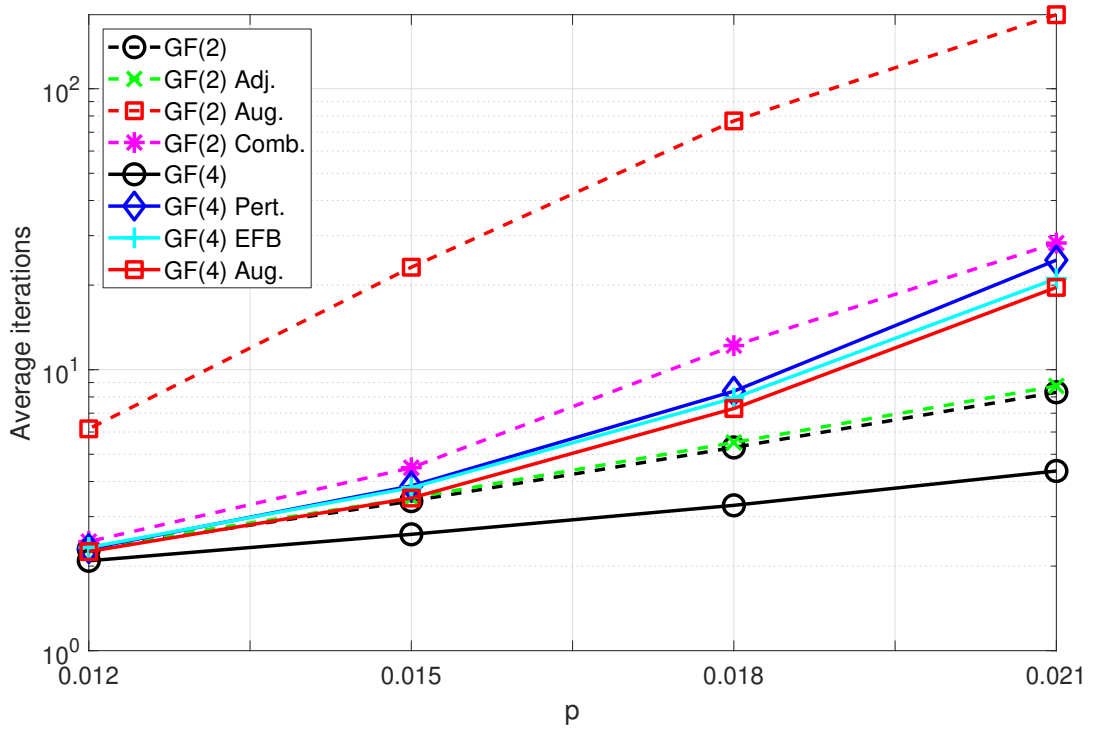

Figure 2.20: Average number of iterations required by decoders with $N = 100$ attempts (where applicable) for the $[[400, 200]]$ bicyclelike code on the depolarizing channel.

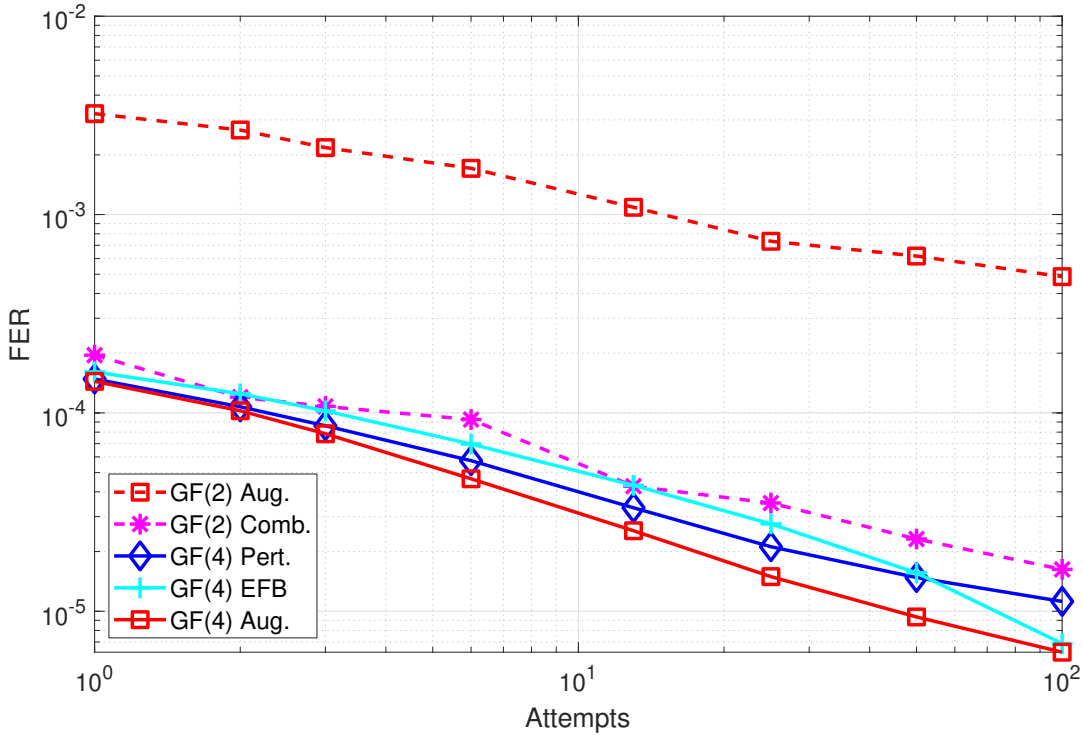

Figure 2.21: FER performance of decoders at $p = 0.012$ with a varying number of decoding attempts for the $[[400, 200]]$ bicyclelike code on the depolarizing channel.

we still observed a moderate number of undetected errors, which can be attributed to the codes moderatly low distance of $d \leq 10$. For example, at $p = 0.015$, approximately 15% of errors are undetected for both the EFB and augmented GF(4) decoders. However, this is not significant enough fraction of errors to prevent the FER reducing near linearly on a log-log plot with an increasing maximum number of attempts as shown in Fig. 2.21.

2.4.5 Non-CSS A

The fifth code we have considered is a $[[400, 202]]$ non-CSS code based on construction three of Ref. [58]. The GF(2) and GF(4) parity-check matrices for this code are defined by the matrices H_X and H_Z as outlined in Eqs. (2.34) and (2.39). For this code, these matrices are of the form

$$H_X = \begin{pmatrix} A_X^{(1)} & A_X^{(2)} & \cdots & A_X^{(a)} \end{pmatrix}, \tag{2.74}$$

$$H_Z = \begin{pmatrix} A_Z^{(1)} & A_Z^{(2)} & \cdots & A_Z^{(a)} \end{pmatrix}. \tag{2.75}$$

The submatrices $A_X^{(i)}$ and $A_Z^{(i)}$ are given by

$$A_X^{(i)} = \begin{pmatrix} B_X^{(i)} & B_X^{(i)T} P_i^T \\ P_i B_X^{(i)T} & P_i B_X^{(i)} P_i^T \end{pmatrix}, \tag{2.76}$$

$$A_Z^{(i)} = \begin{pmatrix} B_Z^{(i)} & B_Z^{(i)T} P_i^T \\ P_i B_Z^{(i)T} & P_i B_Z^{(i)} P_i^T \end{pmatrix}. \tag{2.77}$$

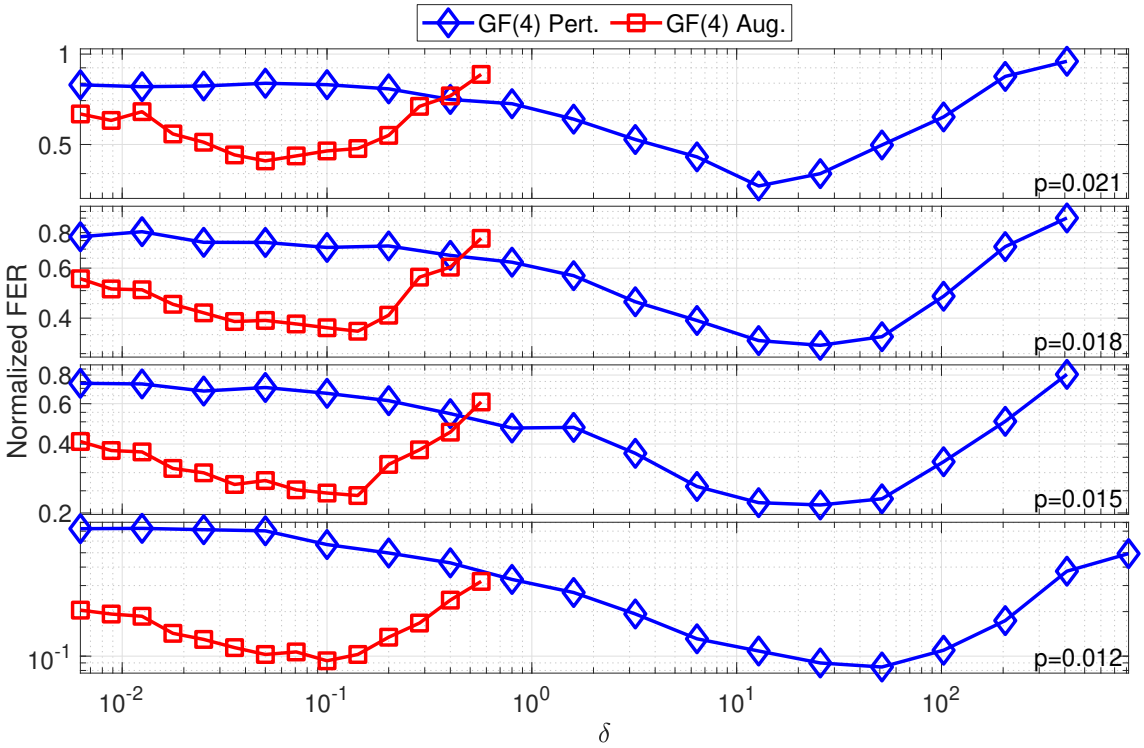

Figure 2.22: The effect of augmentation density and random perturbation strength on decoder performance for the $[[400, 202]]$ non-CSS code A on the depolarizing channel. Each decoder uses $N = 10$ maximum attempts.

Here, $B_X^{(i)}$ and $B_Z^{(i)}$ are square matrices of the same size that are either both symmetric or both circulant; P_i is a square matrix satisfying $P_i^T = P_i^{-1}$. For our code, we have taken $a = 2$, each $B_X^{(i)}$ and $B_Z^{(i)}$ to be a 100×100 circulant matrix of weight three, and each P_i to be a 100×100 permutation matrix.

The effect of augmentation density and random perturbation strength for decoders with $N = 10$ on this code is shown in Fig. 2.22. Note that for non-CSS codes, we can only use GF(4)-based decoders. Based on these results, we have chosen $\delta = 0.1$ for the augmented decoder and $\delta = 25$ for the random perturbation decoder.

The FER performance and average required iterations for decoders with $N = 100$ maximum attempts are shown in Figs. 2.23 and 2.24, respectively. It can be seen that the random perturbation, EFB, and augmented decoders all perform similarly on this code.

The majority of decoding errors are detected errors for this code, which also has distance $d \leq 10$. At $p = 0.012$, only 1-2% of errors are undetected for the random perturbation, EFB, and augmented decoders. This is again reflected in the near-linear reduction in FER with increasing maximum number of attempts on the log-log plot given in Fig. 2.25.

2.4.6 Non-CSS B

The final code we have considered is a $[[400, 201]]$ non-CSS code based on construction four of Ref. [58]. This construction is quite similar to that of the last section, with H_X and H_Z defined

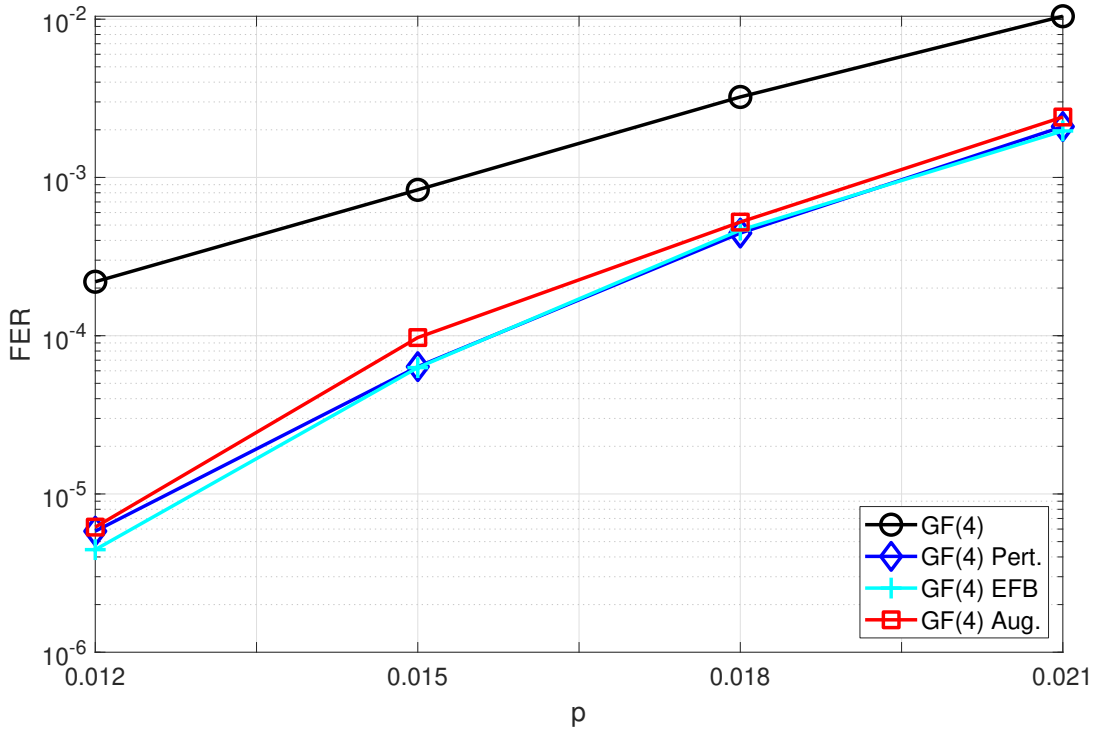

Figure 2.23: FER performance of decoders with $N = 100$ attempts (where applicable) for the $[[400, 202]]$ non-CSS code A on the depolarizing channel.

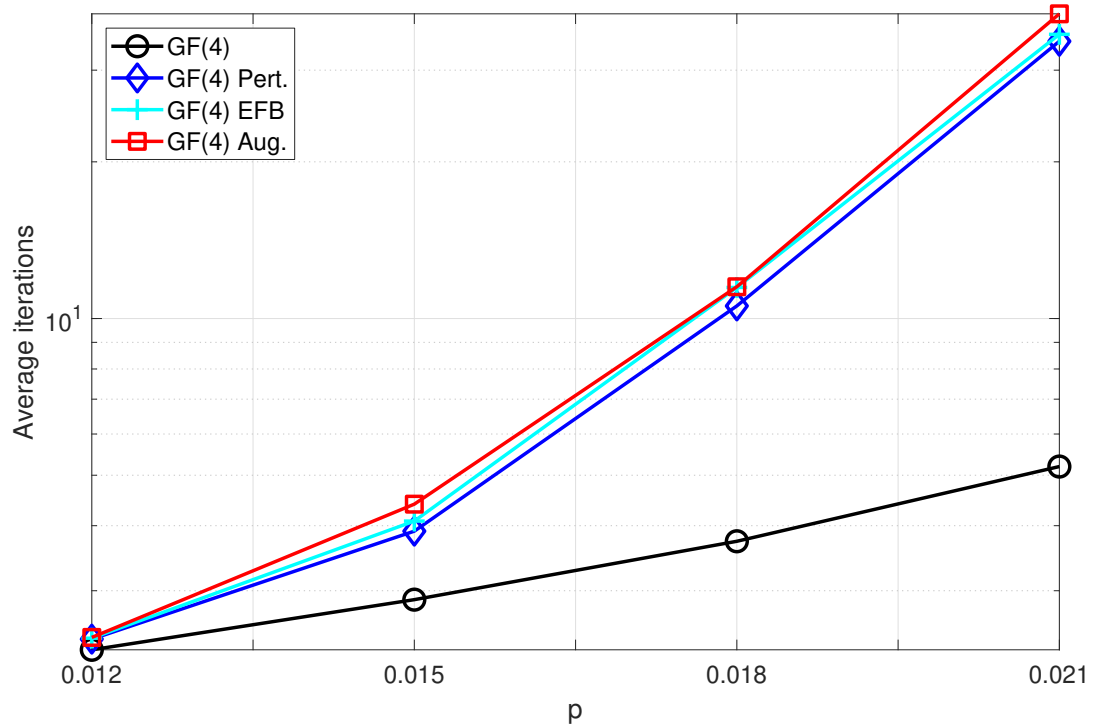

Figure 2.24: Average number of iterations required by decoders with $N = 100$ attempts (where applicable) for the $[[400, 202]]$ non-CSS code A on the depolarizing channel.

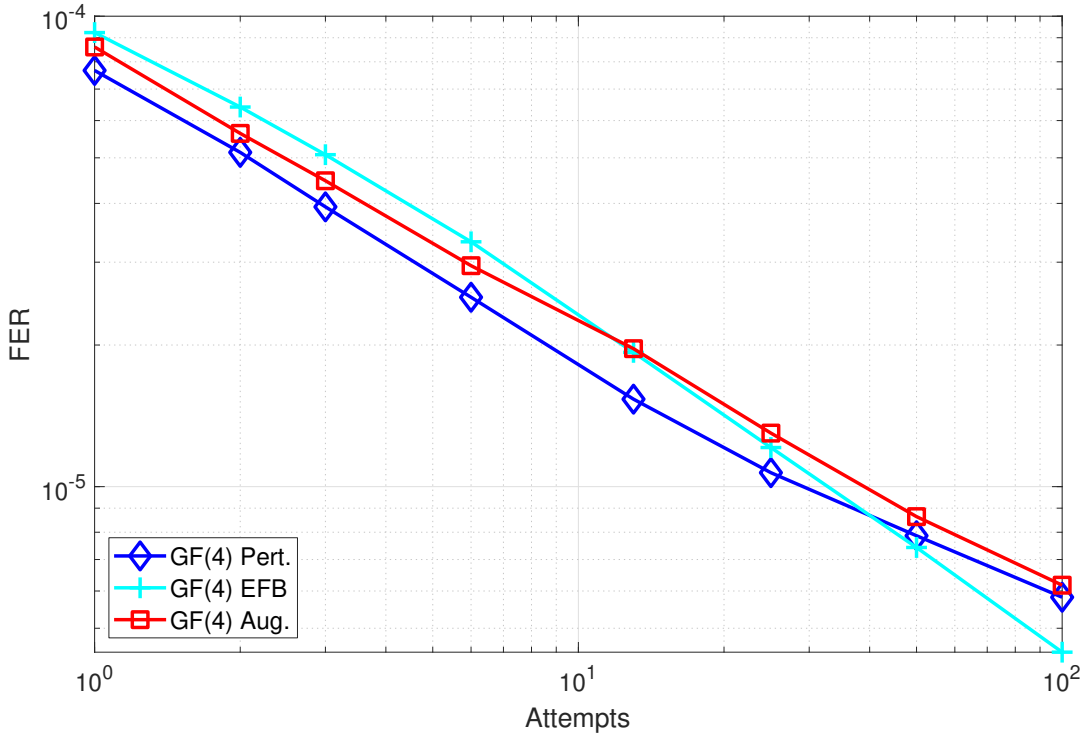

Figure 2.25: FER performance of decoders at $p = 0.012$ with a varying number of decoding attempts for the $[[400, 202]]$ non-CSS code A on the depolarizing channel.

as in Eqs. (2.74) and (2.75), respectively. However, the submatrices $A_X^{(i)}$ and $A_Z^{(i)}$ are now given by

$$A_X^{(i)} = \begin{pmatrix} B_X^{(i)} & B_X^{(i)T} P_i^T \end{pmatrix}, \tag{2.78}$$

$$A_Z^{(i)} = \begin{pmatrix} B_Z^{(i)} & B_Z^{(i)T} P_i^T \end{pmatrix}. \tag{2.79}$$

Here, $B_X^{(i)}$ and $B_Z^{(i)}$ are either both symmetric, both circulant, or $B_X^{(i)} B_Z^{(i)T} + B_X^{(i)T} B_Z^{(i)}$ is symmetric; P_i is a permutation matrix. For our code, we have taken $a = 1$, $B_X^{(1)}$ and $B_Z^{(1)}$ to be 200×200 circulant matrices of weight six and P_1 to be a 200×200 permutation matrix.

The effect of augmentation density and random perturbation strength for decoders with $N = 10$ on this code is shown in Fig. 2.26. Based on these results, we have chosen $\delta = 0.05$ for the augmented decoder and $\delta = 25$ for the random perturbation decoder.

The FER performance and average required iterations for decoders with $N = 100$ maximum attempts are shown in Figs. 2.27 and 2.28, respectively. The results are consistent with those of Sec. 2.4.5, with the random perturbation, EFB, and augmented decoders all performing fairly similarly on this code.

A moderate number of undetected errors were observed for this code, which also has distance $d \leq 10$. For example, at $p = 0.012$, approximately 10-15% of errors are undetected for each decoder. It can also be seen that the reduction in FER with an increasing number of maximum attempts, while still near linear on the log-log plot of Fig. 2.29, tapers of slightly more than was observed for the code of the previous section.

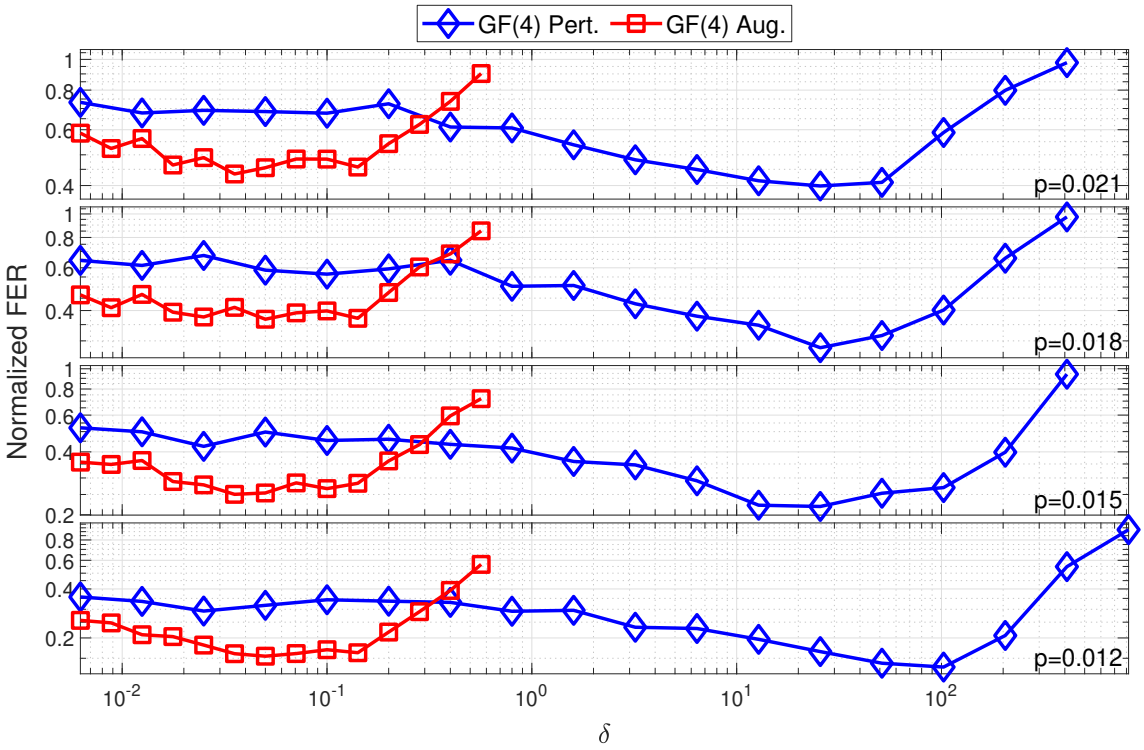

Figure 2.26: The effect of augmentation density and random perturbation strength on decoder performance for the $[[400, 201]]$ non-CSS code B on the depolarizing channel. Each decoder uses $N = 10$ maximum attempts.

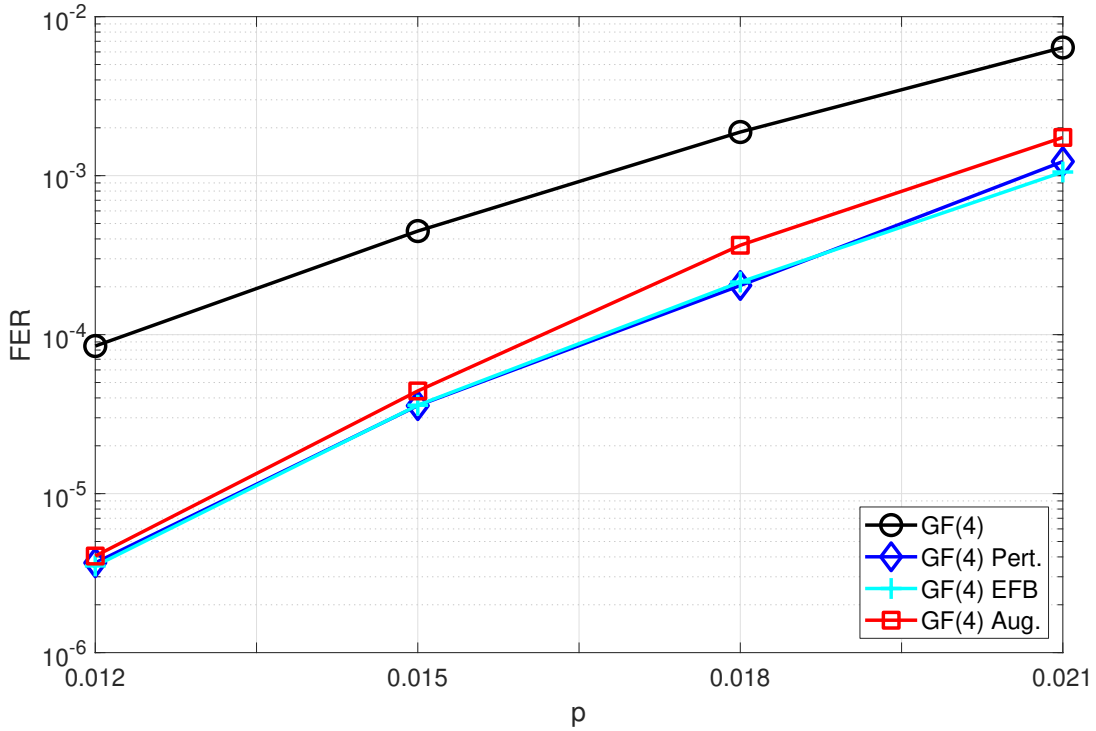

Figure 2.27: FER performance of decoders with $N = 100$ attempts (where applicable) for the $[[400, 201]]$ non-CSS code B on the depolarizing channel.

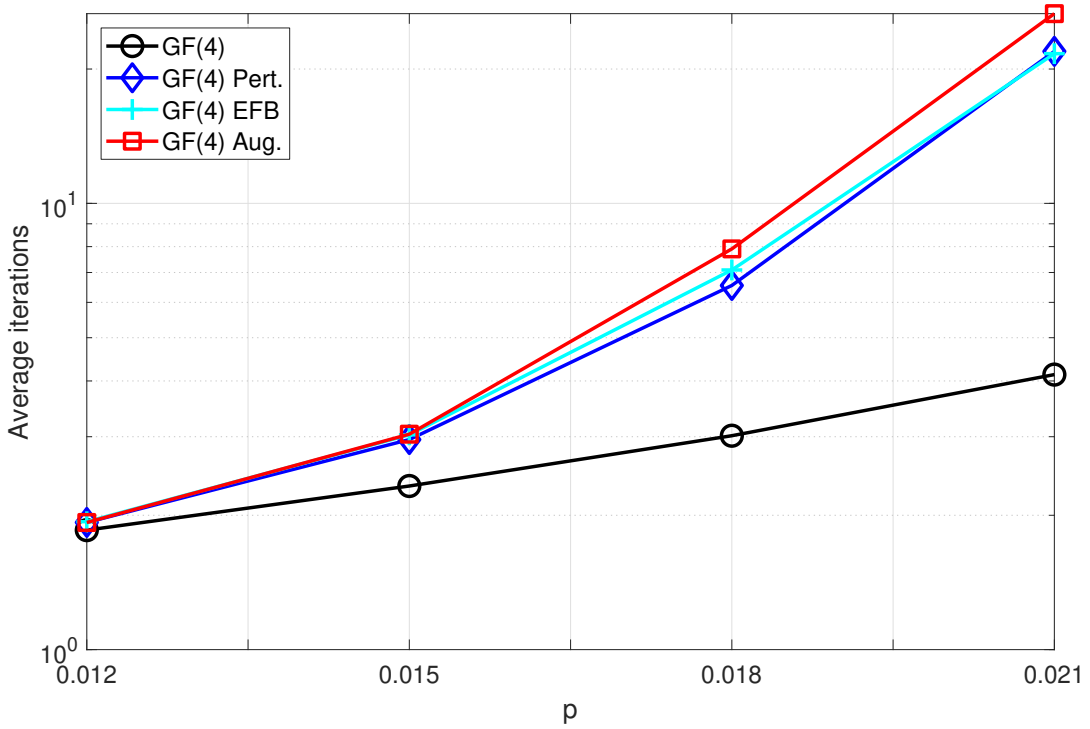

Figure 2.28: Average number of iterations required by decoders with $N = 100$ attempts (where applicable) for the $[[400, 201]]$ non-CSS code B on the depolarizing channel.

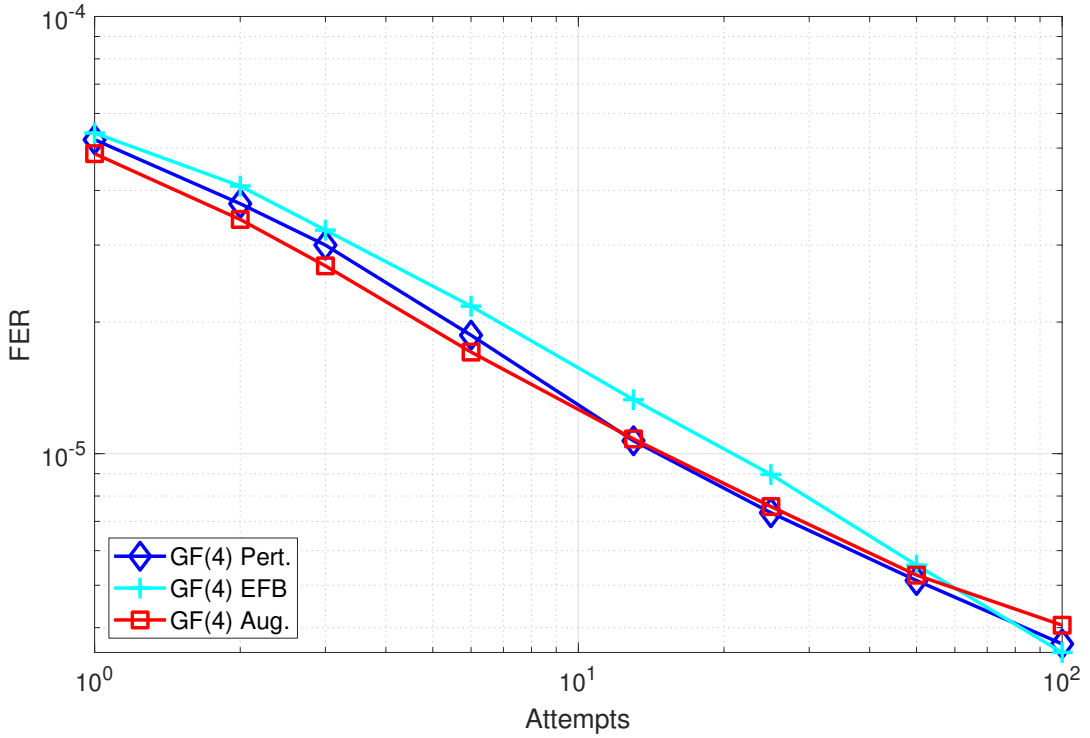

Figure 2.29: FER performance of decoders at $p = 0.012$ with a varying number of decoding attempts for the $[[400, 201]]$ non-CSS code B on the depolarizing channel.

2.5 Conclusion

We have presented modified belief propagation decoders for QLDPC codes that, depending on the code, either outperform or perform similarly to other decoders presented in literature. We have proposed the GF(2)-based adjusted decoder, which uses modified error probabilities to reintroduce correlations between the X and Z components of an error that are lost when using a standard GF(2) decoder. Furthermore, we have demonstrated that the augmented decoder, which has previously been proposed for classical binary LDPC codes, can be applied in the quantum case, and that it can be based on an underlying GF(2), GF(4), or supernode decoder. We have also proposed a combination of the augmented GF(2) and adjusted decoders. For the bicycle- and BIBD-based dual-containing CSS codes tested, the augmented GF(4), augmented supernode, and combined decoders were shown to outperform random perturbation and EFB decoders. For the two nondual-containing CSS codes and the two non-CSS codes considered, augmented GF(4) and supernode decoders were shown to perform similarly to random perturbation and EFB decoders.

2.A Appendix: Check node Fourier transform implementations

2.A.1 Classical decoding

The check constraint of Eq. (2.18) can be written as

$$\sum_{j' \in \mathcal{M}(i) \backslash j} H_{ij'} e_{j'} = \sum_{j' \in \mathcal{M}(i) \backslash j} \tilde{e}_{j'} = z_i - H_{ij} a, \tag{2.80}$$

where $\tilde{e}_{j'} = H_{ij'} e_{j'}$. $\tilde{e}_{j'}$ can be used to define

$$\tilde{\lambda}^a_{i \to j} = \sum_{\boldsymbol{e}: \sum_{j''} \tilde{e}_{j''} = a} \prod_{j'} \mu^{e_{j''}}_{j' \to i} = \sum_{\boldsymbol{e}: \sum_{j''} \tilde{e}_{j''} = a} \prod_{j'} \tilde{\mu}^{\tilde{e}_{j'}}_{j' \to i}, \tag{2.81}$$

where $\tilde{\mu}^{\tilde{e}_{j'}}_{j' \to i} = \mu^{H^{-1}_{ij'} \tilde{e}_{j'}}_{j' \to i}$ (this corresponds to a permutation of elements) and $j', j'' \in \mathcal{M}(i) \backslash j$. Equation (2.81) is a convolution (see Refs. [59, 51] for details) and, as such, can be efficiently computed using a Fourier transform $\mathcal{F}$ as

$$\tilde{\lambda}_{i \to j} = K \mathcal{F}^{-1} \left\{ \prod_{j'} \mathcal{F}\{\tilde{\mu}_{j' \to i}\} \right\}, \tag{2.82}$$

where $\mathcal{F}^{-1}$ is the inverse Fourier transform and the product is element-wise (K is a normalization factor). A Hadamard transform can be used in the binary case; if $\tilde{\mu}_{j' \to i} = (\tilde{\mu}^0_{j' \to i}, \tilde{\mu}^1_{j' \to i})$ is considered as a column vector, then

$$\mathcal{F}\{\tilde{\mu}_{j' \to i}\} = F \tilde{\mu}_{j' \to i}, \tag{2.83}$$

where

$$F \propto \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.84}$$

The inverse transform is also achieved through multiplication by F (up to some unimportant scaling factor). $\lambda_{i\to j}$ is a permuted version of $\tilde{\lambda}_{i\to j}$, with

$$\lambda^a_{i\to j} = \tilde{\lambda}^{z_i - H_{ij}a}_{i\to j}. \tag{2.85}$$

2.A.2 GF(4) stabilizer decoding

The check constraint of Eq. (2.42) can be written as

$$\mathrm{tr}\left(H_{ij}\bar{a} + \sum_{j'\in\mathcal{M}(i)\setminus j} H_{ij'}\bar{e_{j'}}\right) = \mathrm{tr}\left(H_{ij}\bar{a} + \sum_{j'\in\mathcal{M}(i)\setminus j} \tilde{e}_{j'}\right) = z_i, \tag{2.86}$$

where $\tilde{e}_{j'} = H_{ij'}\bar{e_{j'}}$. $\tilde{\lambda}^a_{i\to j}$ is defined in the same was as Eq. (2.81), with $\tilde{\mu}^{\tilde{e}_{j'}}_{j'\to i} = \mu^{(H^{-1}_{ij'}\tilde{e}_{j'})^{-1}}_{j'\to i} = \mu^{H_{ij'}\bar{e}^{-1}_{j'}}_{j'\to i}$. Again, $\tilde{\lambda}_{i\to j}$ can be calculated using the Hadamard transform, with

$$F \propto \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes 2} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}. \tag{2.87}$$

$\tilde{\lambda}^a_{i\to j}$ corresponds to $\sum_{j'}\tilde{e}_{j'} = a$ and, as such, can be used to determine $\lambda^a_{i\to j}$, which corresponds to $e_j = a$. If $z_i = 0$, then $H_{ij}\bar{a} + \sum_{j'}\tilde{e}_{j'} = 0$ or 1; conversely, if $z_i = 1$, then $H_{ij}\bar{a} + \sum_{j'}\tilde{e}_{j'} = \omega$ or $\bar{\omega}$. Therefore, for $z_i = 0$, the elements of $\lambda_{i\to j}$ are

$$\lambda^a_{i\to j} = \frac{1}{2}\left[\tilde{\lambda}^{-H_{ij}\bar{a}}_{i\to j} + \tilde{\lambda}^{1-H_{ij}\bar{a}}_{i\to j}\right], \tag{2.88}$$

and for $z_i = 1$,

$$\lambda^a_{i\to j} = \frac{1}{2}\left[\tilde{\lambda}^{\omega-H_{ij}\bar{a}}_{i\to j} + \tilde{\lambda}^{\bar{\omega}-H_{ij}\bar{a}}_{i\to j}\right]. \tag{2.89}$$

These can be combined to give

$$\lambda^a_{i\to j} = \frac{1}{2}\left[\tilde{\lambda}^{\omega z_i - H_{ij}\bar{a}}_{i\to j} + \tilde{\lambda}^{\omega z_i + 1 - H_{ij}\bar{a}}_{i\to j}\right]. \tag{2.90}$$

Chapter 3

Optimizing short stabilizer codes for asymmetric channels[1]

Abstract

For a number of quantum channels of interest, phase-flip errors occur far more frequently than bit-flip errors. When transmitting across these asymmetric channels, the decoding error rate can be reduced by tailoring the code used to the channel. However, analyzing the performance of stabilizer codes on these channels is made difficult by the #P-completeness of optimal decoding. To address this, at least for short codes, we demonstrate that the decoding error rate can be approximated by considering only a fraction of the possible errors caused by the channel. Using this approximate error rate calculation, we extend a recent result to show that there are a number of $[[5 \leq n \leq 12, 1 \leq k \leq 3]]$ cyclic stabilizer codes that perform well on two different asymmetric channels. We also demonstrate that an indication of a stabilizer code's error rate is given by considering the error rate of a classical binary code related to the stabilizer. This classical error rate is far less complex to calculate, and we use it as the basis for a hill-climbing algorithm, which we show to be effective at optimizing codes for asymmetric channels. Furthermore, we demonstrate that simple modifications can be made to our hill-climbing algorithm to search for codes with desired structure requirements.

[1]This chapter has been published as Ref. [38]: A. Rigby, J. C. Olivier, and P. D. Jarvis, "Optimizing short stabilizer codes for asymmetric channels," *Physical Review A*, vol. 101, no. 3, p. 032326, Mar. 2020, doi.org/10.1103/PhysRevA.101.032326. Only minor typographical and formatting changes have been made.

3.1 Introduction

Quantum codes can be employed to protect quantum information against the effects of a noisy channel. Of particular note are the stabilizer codes, which are defined by a stabilizer $\mathcal{S}$ that is an Abelian subgroup of the n-qubit Pauli group $\mathcal{P}_n$ [15]. An $[[n, k]]$ stabilizer code encodes the state of a k-qubit system in that of an n-qubit system; that is, it is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$ of dimension 2^k. For a Pauli channel, an error E acting on the code is also an element of $\mathcal{P}_n$, with the component acting on any given qubit being I with probability p_I, X with probability p_X, Y with probability p_Y, or Z with probability p_Z. Most stabilizer codes are implicitly designed for good decoding performance (that is, a low decoding error rate) on the depolarizing channel, where $p_X = p_Y = p_Z$. This is achieved by ensuring that the code has large distance d, which is the weight of the lowest-weight error that yields a trivial syndrome while having a nontrivial effect on the code. However, for a number of channels of physical interest, Z errors occur far more frequently than X errors [27, 28]. For these channels, better decoding performance can be achieved by using codes that are tailored to the channel [29, 30].

In this paper, our focus is on the construction of stabilizer codes for two different asymmetric channels. The first of these is the biased XZ channel, for which the X and Z components of an error occur independently at different rates. The second is a Pauli approximation of the combined amplitude damping (AD) and dephasing channel. Both of these channels have two degrees of freedom, which means that the values of p_X, p_Y, and p_Z can be defined via the total error probability $p = p_X + p_Y + p_Z$ and bias $\eta = p_Z/p_X$ [29, 60]. A well-studied approach to constructing codes for asymmetric channels is to restrict consideration to Calderbank-Shor-Steane (CSS) codes [52, 53], which can be designed to have separate X and Z distances d_X and d_Z (typically $d_Z > d_X$) [60, 61, 62, 63, 64, 65]. We wish to take a more direct approach to the problem by actually determining the decoding error rates (frame/block/word error rate in particular) of the codes we construct (this also allows us to meaningfully consider non-CSS codes). However, to do this, we have to overcome the #P-completeness of stabilizer decoding [16], which stems from the equivalence of errors up to an element of the stabilizer. To achieve this, at least for short codes (that is, codes with small n), we first demonstrate that the error rate of an optimal decoder can be approximated by considering only a small subset $\mathcal{E}$ of the 4^n possible Pauli errors. We derive a bound on the relative error in this approximation, and we demonstrate that the independence of error components can be exploited to construct $\mathcal{E}$ without having to enumerate all possible errors. We also show that the performance of a classical $[2n, n+k]$ binary linear code associated with the stabilizer [43, 15] gives an indication of the stabilizer code's performance (note that whenever we mention a code's performance or error rate, we are referring to that of the associated decoder). It is several orders of magnitude faster to calculate this classical error rate, and we show that it can itself be approximated using a limited error set.

We have a particular focus on cyclic codes, which are stabilizer codes based on classical self-orthogonal additive cyclic GF(4) codes [41, 66, 67] [where GF(q) is the q-element finite field]. This is motivated by the recent result of Ref. [29], where it was shown that a $[[7, 1]]$ cyclic code performs near optimally compared to 10 000 randomly constructed codes on the biased

XZ channel for a range of error probabilities and biases. We extend this result by enumerating the $[[5 \leq n \leq 12, 1 \leq k \leq 3]]$ cyclic codes and making use of our approximate error rate calculation. In particular, we demonstrate that there are a number of cyclic codes that perform well compared to the best of 10 000 randomly constructed codes for both the biased XZ and AD channels across a range of p and η values. In some cases, such as $[[n \geq 9, 1]]$ codes for the biased XZ channel, the best cyclic codes significantly outperform the best of the random codes constructed. To improve on the poor performance of the random search, we demonstrate the effectiveness of a simple hill-climbing algorithm that attempts to optimize the performance of the classical binary code associated with a stabilizer. We also show that by modifying the mutation operation employed by this hill-climbing algorithm, we can effectively search for codes with desired structure. In particular, we show that we can search for codes with weight-four generators, CSS codes, and linear codes.

The paper is organized as follows. Section 3.2 gives an overview of classical codes, asymmetric quantum channels, and stabilizer codes. In Sec. 3.3, we detail our methods for calculating approximate error rates. In Sec. 3.4, we demonstrate the performance of cyclic codes, outline our hill-climbing search algorithm, and show its effectiveness. The paper is concluded in Sec. 3.5.

3.2 Background

3.2.1 Classical codes

A classical channel Φ maps a set of inputs $\mathcal{A}_x$ to a set of outputs $\mathcal{A}_y$. We are interested in the case where $\mathcal{A}_x = \mathcal{A}_y = \mathrm{GF}(q)$, for which the action of the channel is given by

$$\Phi(x) = x + e = y, \tag{3.1}$$

where $x \in \mathrm{GF}(q)$ is the channel input, $y \in \mathrm{GF}(q)$ is the channel output, and $e \in \mathrm{GF}(q)$ is an error (or noise) symbol that occurs with probability $P(e)$. Φ is called symmetric if $P(0) = 1 - p$ and $P(e_i) = p/(q-1)$ for $e_i \neq 0$. The noise introduced by the channel can be protected against using a code $\mathcal{C} \subseteq \mathrm{GF}(q)^n$, whose elements are called codewords. The effect of the combined channel Φ^n, which is composed of n copies of Φ, on some codeword $\boldsymbol{x} \in \mathcal{C}$ is

$$\Phi^n(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{e} = \boldsymbol{y}, \tag{3.2}$$

where $\boldsymbol{y} \in \mathrm{GF}(q)^n$ is the channel output and $\boldsymbol{e} \in \mathrm{GF}(q)^n$ is an error "vector." Assuming that error components occur independently, the probability of $\boldsymbol{e} = (e_1, \ldots, e_n)$ occurring is

$$P(\boldsymbol{e}) = \prod_{i=1}^{n} P(e_i), \tag{3.3}$$

where $P(e_i)$ is the probability of the error symbol e_i occurring on Φ. It follows that for a symmetric channel, the probability of an error $\boldsymbol{e}$ occurring depends only on its weight $w(\boldsymbol{e})$, which is the number of nonzero components from which it is composed. The distance d of a

code is the weight of the lowest-weight error mapping one codeword to another. The (minimum) weight $w(\mathcal{C})$ of a code $\mathcal{C}$ is simply the weight of the lowest-weight codeword it contains.

A code is called additive if it forms a group (under addition) and linear if it forms a vector space. Such codes can be described by a generator matrix

$$G^T = \begin{pmatrix} \boldsymbol{b}_1 & \cdots & \boldsymbol{b}_k \end{pmatrix}, \tag{3.4}$$

where $\mathcal{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\}$ is either a generating set or basis, respectively (note that we consider codewords as column vectors). A linear code can also be defined as the kernel of a $\mathrm{GF}(q)$ parity-check matrix H; that is,

$$\mathcal{C} = \{\boldsymbol{x} \in \mathrm{GF}(q)^n : H\boldsymbol{x} = \boldsymbol{0}\}. \tag{3.5}$$

If H has m rows, then $\dim(\mathcal{C}) = k \geq n - m$, with equality when H is full rank. For a linear code, the errors mapping one codeword to another are themselves codewords; therefore, it follows that the distance of a linear code $\mathcal{C}$ is simply $d = w(\mathcal{C})$. A linear code of length n with dimension k and distance d is called an $[n, k]_q$ or $[n, k, d]_q$ code (the q is typically omitted for binary codes, where $q = 2$). More generally, a length-n code of size $|\mathcal{C}| = K$ and distance d is called an $(n, K)_q$ or $(n, K, d)_q$ code.

The dual code of some $\mathcal{C} \subseteq \mathrm{GF}(q)^n$ with respect to the inner product $\langle \cdot, \cdot \rangle : \mathrm{GF}(q)^n \times \mathrm{GF}(q)^n \to \mathrm{GF}(q)$ is

$$\mathcal{C}^\perp = \{\boldsymbol{c} \in \mathrm{GF}(q)^n : \langle \boldsymbol{c}, \boldsymbol{x} \rangle = 0 \ \forall \ \boldsymbol{x} \in \mathcal{C}\}. \tag{3.6}$$

$\mathcal{C}^\perp$ is the annihilator of $\mathcal{C}$ and is therefore a linear code. If $\mathcal{C}^\perp \subseteq \mathcal{C}$, then $\mathcal{C}$ is called dual containing; if $\mathcal{C} \subseteq \mathcal{C}^\perp$, then $\mathcal{C}$ is called self-orthogonal; and if $\mathcal{C}^\perp = \mathcal{C}$, then $\mathcal{C}$ is called self-dual. Note that if $\mathcal{C}$ is dual containing, then $\mathcal{C}^\perp$ is self-orthogonal and vice versa. Unless otherwise specified, the dual code is with respect to the Euclidean inner product

$$\langle \boldsymbol{c}, \boldsymbol{x} \rangle = \boldsymbol{c} \cdot \boldsymbol{x} = \sum_{i=1}^{n} c_i x_i. \tag{3.7}$$

In this case, if $\mathcal{C}$ is linear with generator matrix G, then a necessary and sufficient condition for $\boldsymbol{c} \in \mathcal{C}^\perp$ is $G\boldsymbol{c} = \boldsymbol{0}$; that is, a generator matrix for $\mathcal{C}$ is a parity-check matrix for $\mathcal{C}^\perp$. Conversely, if H is a parity-check matrix for $\mathcal{C}$, then it is a generator matrix for $\mathcal{C}^\perp$.

A decoder uses the output of a channel to infer its input. For a linear code, this inference can be aided by the syndrome

$$\boldsymbol{z} = H\boldsymbol{y} = H(\boldsymbol{x} + \boldsymbol{e}) = H\boldsymbol{e}. \tag{3.8}$$

As channel outputs that differ only by a codeword yield the same syndrome, the q^{n-k} possible syndromes can be associated with the cosets of $\mathrm{GF}(q)^n/\mathcal{C}$. Given some syndrome measurement $\boldsymbol{z}$, an optimal maximum *a posteriori* (MAP) decoder will then return the most probable error

$$\hat{\boldsymbol{e}}_{\boldsymbol{z}} = \operatorname*{argmax}_{\boldsymbol{e} \in \mathrm{GF}(q)^n} P(\boldsymbol{e}|\boldsymbol{z}) \tag{3.9}$$

in the corresponding coset. The channel input can then be inferred as $\hat{\boldsymbol{x}} = \boldsymbol{y} - \hat{\boldsymbol{e}}_{\boldsymbol{z}}$. If $\hat{\boldsymbol{e}} = \hat{\boldsymbol{e}}_{\boldsymbol{z}}$ (and hence $\hat{\boldsymbol{x}} = \boldsymbol{x}$), then decoding is successful; otherwise, a decoding error has occurred. The

probability of such a decoding error, called the frame error rate (FER), is simply

$$F = 1 - \sum_{\boldsymbol{z} \in \mathrm{GF}(q)^{n-k}} P(\hat{\boldsymbol{e}}_z). \tag{3.10}$$

Unfortunately, even in the simple case of a binary code operating on the binary symmetric channel (a symmetric channel with $q = 2$), this decoding problem can be shown to be NP-complete [47]. This complicates the design of highly performant codes (that is, codes yielding a low FER). In practice, when designing codes for symmetric channels, the simpler goal of achieving a large distance is typically settled for. This is motivated by the fact that for low-distance codes, there are many errors in each coset $\hat{\boldsymbol{e}}_z + \mathcal{C}$ with weight, and hence probability, similar to $\hat{\boldsymbol{e}}_z$, which leads to a high FER according to Eq. (3.10) (see Sec. 2.2.1 for a more detailed discussion).

Two codes $\mathcal{C}$ and $\mathcal{C}'$ are called permutation equivalent if they are the same up to a relabeling of coordinates. Permutation-equivalent codes share a large number of properties including length, size, and distance; furthermore, they yield the same FER for channels where the error components are independently and identically distributed. While there are more general notions of code equivalence, whenever we say that two codes are equivalent, we mean that they are permutation equivalent in this paper. Furthermore, if some family (set) of codes $\{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ can be split into M equivalence classes (according to permutation equivalence), then we simply say that M of the codes are inequivalent.

3.2.2 Cyclic codes

Cyclic codes are those for which a cyclic shift of any codeword is also a codeword; that is, for a cyclic code $\mathcal{C}$, if $(c_0, c_1, \ldots, c_{n-1}) \in \mathcal{C}$, then it is also the case that $(c_{n-1}, c_0, \ldots, c_{n-2}) \in \mathcal{C}$ (note that to be consistent with standard convention, we index the codewords of cyclic codes from zero in this section). If $\mathcal{C}$ is linear, then it has a convenient description through the mapping

$$\boldsymbol{c} = (c_0, c_1, \ldots, c_{n-1}) \leftrightarrow c_0 + c_1 x + \cdots + c_{n-1} x^{n-1} = c(x) \tag{3.11}$$

of codewords to polynomials in $\mathrm{GF}(q)[x]$. Cyclic shifts of codewords correspond to a multiplication by x taken modulo $x^n - 1$; that is, $(c_{n-1}, c_0, \ldots, c_{n-2}) \leftrightarrow xc(x) \,(\mathrm{mod}\, x^n - 1)$. As $\mathcal{C}$ is linear, $r(x)c(x) \,(\mathrm{mod}\, x^n - 1)$ is a codeword for any $r(x) \in \mathrm{GF}(q)[x]$, from which it follows that $\mathcal{C}$ corresponds to an ideal $I_{\mathcal{C}} \in \mathrm{GF}(q)[x]/(x^n - 1)$. Any such ideal is principal and is generated by a unique monic polynomial of minimal degree $g(x) \in I_{\mathcal{C}}$ that is a factor of $x^n - 1$ [68]; through slight abuse of notation, we write $\mathcal{C} = \langle g(x) \rangle$. $\mathcal{C}$ has dimension $k = n - \deg(g)$ and has a generator matrix

$$G = \begin{pmatrix} g_0 & \cdots & g_{n-k} & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & g_0 & \cdots & g_{n-k} \end{pmatrix}. \tag{3.12}$$

Furthermore, a parity-check matrix

$$H = \begin{pmatrix} h_k & \cdots & h_0 & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & h_k & \cdots & h_0 \end{pmatrix} \tag{3.13}$$

is given in terms of the check polynomial $h(x) = (x^n - 1)/g(x)$. It follows that the dual code $\mathcal{C}^\perp$ is also cyclic and is generated by $x^k h(x^{-1})$.

In the quantum setting, we are particularly interested in codes over $\mathrm{GF}(4) = \{0, 1, \omega, \omega^2 = \bar{\omega}\}$ that are self-orthogonal with respect to the trace inner product (this will be explained further in Sec. 3.2.4). Note that the trace inner product of $\boldsymbol{a}, \boldsymbol{b} \in \mathrm{GF}(4)^n$ is

$$\boldsymbol{a} * \boldsymbol{b} = \mathrm{tr}(\boldsymbol{a} \cdot \bar{\boldsymbol{b}}) = \mathrm{tr}\left(\sum_{i=1}^{n} a_i \bar{b_i}\right), \tag{3.14}$$

where $\bar{0} = 0$, $\bar{1} = 1$, $\bar{\omega} = \omega^2$, and $\bar{\omega^2} = \omega$; and $\mathrm{tr}(x) = x + \bar{x}$ [that is, $\mathrm{tr}(0) = \mathrm{tr}(1) = 0$ and $\mathrm{tr}(\omega) = \mathrm{tr}(\bar{\omega}) = 1$]. A linear cyclic $\mathrm{GF}(4)$ code $\mathcal{C} = \langle g(x) \rangle$ is self-orthogonal if and only if $g(x)g^\dagger(x) \equiv 0 \,(\mathrm{mod}\, x^n - 1)$ [41], where

$$g^\dagger(x) = \bar{g}_0 + \sum_{j=1}^{n-1} \bar{g}_{n-j} x^j. \tag{3.15}$$

More generally, an $(n, 2^k)_4$ additive cyclic code $\mathcal{C}$ has two generators [41, 66, 67]. Following the formulation of Ref. [41], $\mathcal{C} = \langle \omega p(x) + q(x), r(x) \rangle$, where $p(x), q(x), r(x) \in \mathrm{GF}(2)[x]$; $p(x)$ and $r(x)$ are factors of $x^n - 1$; and $r(x)$ is also a factor of $q(x)(x^n - 1)/p(x)$. In general, the choice of generators is not unique; however, any other representation will be of the form $\mathcal{C} = \langle \omega p(x) + q'(x), r(x) \rangle$, where $q'(x) \equiv q(x) \,[\mathrm{mod}\, r(x)]$. The size of $\mathcal{C}$ is given by $k = 2n - \deg(p) - \deg(r)$, with a generator matrix consisting of $n - \deg(p)$ cyclic shifts of the codeword corresponding to $\omega p(x) + q(x)$ and $n - \deg(r)$ cyclic shifts of the codeword corresponding to $r(x)$. $\mathcal{C}$ is self-orthogonal (with respect to the trace inner product) if and only if

$$p(x)r(x^{n-1}) \equiv p(x^{n-1})r(x) \equiv 0 \,(\mathrm{mod}\, x^n - 1), \tag{3.16}$$

$$p(x)q(x^{n-1})r(x) \equiv p(x^{n-1})q(x) \,(\mathrm{mod}\, x^n - 1). \tag{3.17}$$

It is possible to enumerate all the self-orthogonal $(n, 2^k)_4$ additive cyclic codes through a slight modification of the method presented in Ref. [69]: $r(x)$ ranges over all factors of $x^n - 1$; for each $r(x)$, $p(x)$ ranges over the factors of $x^n - 1$ of degree $2n - k - \deg(r)$ that satisfy Eq. (3.16); and for each pair of $r(x)$ and $p(x)$, $q(x)$ ranges over the polynomials with $\deg(q) \leq \deg(r)$ that satisfy both Eq. (3.17) and $q(x)(x^n - 1) \equiv 0 \,[\mathrm{mod}\, p(x)r(x)]$.

While every additive cyclic code has a "canonical" representation involving two generators, many of them can be described using only one [66, 67] (that is, they have a generating set composed of cyclic shifts of a single codeword). This is guaranteed to be the case if $r(x) = x^n - 1$ or if $p(x) = x^n - 1$ and $q(x)$ is a multiple of $r(x)$. However, these are not necessary conditions for a single-generator representation to exist. For example, there is a $(5, 2^5)_4$ code with $p(x) = 1 + x$,

$q(x) = x^3$, and $r(x) = 1 + x + x^2 + x^3$, which gives a canonical generator matrix

$$G = \begin{pmatrix} \omega & \omega & 0 & 1 & 0 \\ 0 & \omega & \omega & 0 & 1 \\ 1 & 0 & \omega & \omega & 0 \\ 0 & 1 & 0 & \omega & \omega \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}; \tag{3.18}$$

however, it is also has the generator matrix

$$G' = \begin{pmatrix} \omega & \omega & 0 & 1 & 0 \\ 0 & \omega & \omega & 0 & 1 \\ 1 & 0 & \omega & \omega & 0 \\ 0 & 1 & 0 & \omega & \omega \\ \omega & 0 & 1 & 0 & \omega \end{pmatrix}. \tag{3.19}$$

We can express this code compactly as $\mathcal{C} = \langle \omega\omega010, 11111 \rangle_{\text{cyc}} \equiv \langle \omega\omega010 \rangle_{\text{cyc}}$.

3.2.3 Quantum channels

The action of a quantum channel Φ on a quantum state described by the density operator ρ is

$$\Phi(\rho) = \sum_k A_k \rho A_k^\dagger, \tag{3.20}$$

where the A_k, called Kraus operators, satisfy $\sum_k A_k^\dagger A_k = I$ (the identity operator) [14]. We are interested in qubit systems, for which states belong to a two-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^2$. Furthermore, we are concerned with Pauli channels, which are of the form

$$\Phi(\rho) = p_I \rho + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z, \tag{3.21}$$

where $p_I + p_X + p_Y + p_Z = 1$, and in the computational $\{|0\rangle, |1\rangle\}$ basis,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3.22}$$

The action of this channel can be interpreted as mapping a pure state $|\phi\rangle$ to $E|\phi\rangle$, where the error E is I with probability p_I, X with probability p_X, Y with probability p_Y, or Z with probability p_Z [6]. X can be viewed as a bit-flip operator as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Z can be viewed as a phase flip as $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. $Y = iXZ$ can be viewed as a combined bit and phase flip.

The quantum equivalent of the symmetric channel is the depolarizing channel, for which $p_I = 1 - p$ and $p_X = p_Y = p_Z = p/3$. For a number of systems of physical interest, phase-flip errors occur far more frequently than bit-flip errors [27, 28]. We focus on two such asymmetric channels in this paper. The first is the biased XZ channel, for which the X and Z components of an error $E \propto X^u Z^v$, where $u, v \in \text{GF}(2)$, occur independently with probabilities q_X and q_Z, respectively. It follows from the independence of the error components that $p_X = q_X(1-q_Z)$, $p_Z = q_Z(1-q_X)$, and $p_Y = q_X q_Z$. A typical way to specify an asymmetric channel with two degrees of freedom

is through the total error probability $p = p_X + p_Y + p_Z$ and bias $\eta = p_Z/p_X$. Note that while this definition of bias is consistent with Refs. [29, 60], some authors give alternate definitions; for example, bias is defined as $p_Z/(p_X + p_Y)$ in Ref. [30] and $(p_Y + p_Z)/(p_X + p_Y)$ in Ref. [70]. Ultimately, the exact nature of the channel parametrization will have no real impact on our results, which has lead us to select the simplest definition of bias. Explicitly, q_X and q_Z can be determined from p and η using

$$q_Z = \frac{1 + \eta + p\eta - p - \sqrt{(1 + \eta + p\eta - p)^2 - 4p\eta^2}}{2\eta}, \tag{3.23}$$

$$q_X = \frac{q_Z}{\eta - \eta q_Z + q_Z}. \tag{3.24}$$

The second channel of interest is the combined amplitude damping (AD) and dephasing channel, which is described by the non-Pauli Kraus operators

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda-\gamma} \end{pmatrix}, \tag{3.25}$$

$$A_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}, \tag{3.26}$$

$$A_2 = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{\lambda} \end{pmatrix}. \tag{3.27}$$

A Pauli approximation of this channel can be obtained through a process called Pauli twirling [71, 72, 73]. In particular, the approximate channel is [60]

$$\Phi_T(\rho) = \frac{1}{4} \sum_{\sigma \in \{I,X,Y,Z\}} \sigma^\dagger \Phi(\sigma \rho \sigma^\dagger) \sigma \tag{3.28}$$

$$= \frac{2 - \gamma + 2\sqrt{1-\lambda-\gamma}}{4} \rho + \frac{\gamma}{4} X\rho X + \frac{\gamma}{4} Y\rho Y + \frac{2 - \gamma - 2\sqrt{1-\lambda-\gamma}}{4} Z\rho Z. \tag{3.29}$$

Again, this channel has two degrees of freedom (λ and γ) and can therefore be described in terms of the total error probability p and bias $\eta = p_Z/p_X$; explicitly,

$$\gamma = \frac{4p}{\eta + 2}, \tag{3.30}$$

$$\lambda = 1 - \gamma - \left(1 - \frac{\gamma}{2} - \frac{2p\eta}{\eta + 2}\right)^2. \tag{3.31}$$

Note that in the case of $\eta = 1$, Φ_T reduces to the depolarizing channel. For the sake of brevity, we will simply refer to Φ_T as the AD channel.

The Pauli matrices are Hermitian, unitary, and anticommute with each other. Furthermore, they form a group

$$\mathcal{P}_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} = \langle X, Y, Z \rangle \tag{3.32}$$

called the Pauli group. The n-qubit Pauli group $\mathcal{P}_n$ is composed of all n-fold tensor product combinations of elements of $\mathcal{P}_1$. Note that when writing elements of $\mathcal{P}_n$, the tensor products are often implied; for example, we may write $I \otimes I \otimes X \otimes I \otimes Y \otimes Z \otimes I \otimes I \in \mathcal{P}_8$ as $IIXIYZII$. The weight $w(g)$ of some $g \in \mathcal{P}_n$ is the number of nonidentity components from which it is composed. It follows from the commutation relations of the Pauli matrices that any two elements of $\mathcal{P}_n$

commute if their nonidentity components differ in an even number of places; otherwise, they anticommute.

As in the classical case, the noise introduced by a quantum channel can be protected against using a code. In the qubit case, a code is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$ whose elements are again called codewords. These codewords are transmitted across the combined n-qubit channel $\Phi^{\otimes n}$, which in the Pauli case, maps a codeword $|\phi\rangle$ to $E|\phi\rangle$, where $E \in \mathcal{P}_n$. Similar to the classical case of Eq. (3.3), if the error components are independent, then the probability of an error $E = E_1 \otimes \cdots \otimes E_n$ occurring (up to phase) is

$$P(E) = \prod_{i=1}^{n} P(E_i), \tag{3.33}$$

where $P(E_i)$ is the probability of the error E_i occurring (up to phase) on the single-qubit channel Φ. The equivalence of errors up to phase can be addressed more explicitly by instead considering $\tilde{E} = \{E, -E, iE, -iE\} \in \mathcal{P}_n/\{\pm I, \pm iI\} = \tilde{\mathcal{P}}_n$.

3.2.4 Stabilizer codes

Stabilizer codes are defined by an Abelian subgroup $\mathcal{S} < \mathcal{P}_n$, called the stabilizer, that does not contain $-I$ [15]. The code $\mathcal{Q}$ is the space of states that are fixed by every element $s_i \in \mathcal{S}$; that is,

$$\mathcal{Q} = \{|\phi\rangle \in (\mathbb{C}^2)^{\otimes n} : s_i|\phi\rangle = |\phi\rangle \; \forall \, s_i \in \mathcal{S}\}. \tag{3.34}$$

The requirement that $-I \notin \mathcal{S}$ means both that no $s \in \mathcal{S}$ can have a phase factor of $\pm i$, and also that if $s \in \mathcal{S}$, then $-s \notin \mathcal{S}$. If $\mathcal{S}$ is generated by $M = \{M_1, \ldots, M_m\} \subset \mathcal{P}_n$, then it is sufficient (and obviously necessary) for $\mathcal{Q}$ to be stabilized by every M_i. Assuming that the set of generators is minimal, which will be the case for all codes considered in this paper, it can be shown that $\dim(\mathcal{Q}) = 2^k$, where $k = n - m$ [6]; that is, $\mathcal{Q}$ encodes the state of a k-qubit system.

Suppose an error E occurs, mapping some codeword $|\phi\rangle \in \mathcal{Q}$ to $E|\phi\rangle$. A projective measurement of a generator M_i will give the result $+1$ if $[E, M_i] = EM_i - M_iE = 0$ or -1 if $\{E, M_i\} = EM_i + M_iE = 0$. These measurement values define the syndrome $\boldsymbol{z} \in \mathrm{GF}(2)^{n-k}$, with

$$z_i = \begin{cases} 0 & \text{if } [E, M_i] = 0, \\ 1 & \text{if } \{E, M_i\} = 0. \end{cases} \tag{3.35}$$

Defining $\tilde{\mathcal{S}} = \{\tilde{s} = \{s, -s, is, -is\} : s \in \mathcal{S}\}$, the syndrome resulting from $\tilde{E} \in \tilde{\mathcal{P}}_n$ depends only on which coset of $\tilde{\mathcal{P}}_n/\widetilde{N(\mathcal{S})}$ it belongs to, where $N(\mathcal{S})$ is the normalizer of $\mathcal{S}$ in $\mathcal{P}_n$ and $\widetilde{N(\mathcal{S})} \subseteq \tilde{\mathcal{P}}_n$ is defined in the same way as $\tilde{\mathcal{S}}$ was. Note that as $\tilde{\mathcal{S}} \lhd \widetilde{N(\mathcal{S})}$, the 2^{n-k} cosets of $\tilde{\mathcal{P}}_n/\widetilde{N(\mathcal{S})}$ are each the union of 2^{2k} cosets of $\tilde{\mathcal{P}}_n/\tilde{\mathcal{S}}$. In the classical case, the distance d of a linear code is equal to the weight of the lowest-weight error yielding a trivial syndrome while having a nontrivial effect on the code. This extends to the quantum case, with the distance d of a stabilizer code being the weight of the lowest-weight element in $\widetilde{N(\mathcal{S})} \backslash \tilde{\mathcal{S}}$ [15]. An n-qubit

code of dimension 2^k and distance d is called an $[[n,k]]$ or $[[n,k,d]]$ code (the double brackets differentiate it from a classical code).

Given the equivalence of errors up to an element of the stabilizer, a MAP decoder will determine the most likely coset

$$\hat{A}_{\boldsymbol{z}} = \underset{A \in \tilde{\mathcal{P}}_n/\tilde{\mathcal{S}}}{\operatorname{argmax}} P(A|\boldsymbol{z}) \tag{3.36}$$

that is consistent with the syndrome measurement. If $\hat{A}_{\boldsymbol{z}}$ has the representative $\hat{\tilde{E}} = \{\hat{E}, -\hat{E}, i\hat{E}, -i\hat{E}\}$, then the decoder attempts correction by applying $\hat{E}$ to the channel output. If $\tilde{E} \in \hat{A}_{\boldsymbol{z}}$, and hence $\hat{\tilde{E}}\tilde{E} \in \tilde{\mathcal{S}}$, then decoding is successful; otherwise, a decoding error has occurred. It therefore follows that the FER is

$$F_{\text{MAP}} = 1 - \sum_{\boldsymbol{z} \in \text{GF}(2)^{n-k}} P(\hat{A}_{\boldsymbol{z}}). \tag{3.37}$$

Unfortunately, this decoding problem has been shown to be #P-complete [16]. Furthermore, the simpler decoding problem of determining the single most likely error

$$\hat{\tilde{E}}_{\boldsymbol{z}} = \underset{\tilde{E} \in \tilde{\mathcal{P}}_n}{\operatorname{argmax}} P(\tilde{E}|\boldsymbol{z}) \tag{3.38}$$

corresponding to the observed syndrome is essentially the same as the classical decoding problem outlined in Sec. 3.2.1 and hence is also NP-complete [74, 75, 76]. The FER for this decoder is

$$F_{\text{MAP-SE}} = 1 - \sum_{\boldsymbol{z} \in \text{GF}(2)^{n-k}} P(\hat{\tilde{E}}_{\boldsymbol{z}} \tilde{\mathcal{S}}), \tag{3.39}$$

where "SE" stands for "single error."

Two stabilizers (or the codes they define) are permutation equivalent if they are equal up to a relabeling of qubits. As in the classical case, if two stabilizer codes are permutation equivalent, then they are both $[[n,k,d]]$ codes; furthermore, they will yield the same FERs (both F_{MAP} and $F_{\text{MAP-SE}}$) when the error components are independently and identically distributed, which is the case for the channels that we consider. Again, while there are more general notions of quantum code equivalence, we are always referring to permutation equivalence in this paper.

The links between stabilizer codes and classical codes can be made more concrete by representing the elements of $\tilde{\mathcal{P}}_n$ as elements of $\text{GF}(2)^{2n}$ [43, 15]. This is achieved via the isomorphism

$$X^{u_1} Z^{v_1} \otimes \cdots \otimes X^{u_n} Z^{v_n} = X^{\boldsymbol{u}} Z^{\boldsymbol{v}} \leftrightarrow (\boldsymbol{u}|\boldsymbol{v}), \tag{3.40}$$

with the product of elements in $\tilde{\mathcal{P}}_n$ corresponding to addition in $\text{GF}(2)^{2n}$. Furthermore, representatives of elements in $\tilde{\mathcal{P}}_n$ commute if the symplectic inner product of the binary representations is zero, where the symplectic inner product of $\boldsymbol{a} = (\boldsymbol{u}|\boldsymbol{v}), \boldsymbol{b} = (\boldsymbol{u}'|\boldsymbol{v}') \in \text{GF}(2)^{2n}$ is $\boldsymbol{a} \circ \boldsymbol{b} = \boldsymbol{u} \cdot \boldsymbol{v}' + \boldsymbol{u}' \cdot \boldsymbol{v}$. Utilizing this isomorphism, the generators of some stabilizer $\mathcal{S}$ can be used to define the rows of an $m \times 2n$ binary matrix

$$H = (H_X|H_Z), \tag{3.41}$$

where H_X and H_Z are each $m \times n$ matrices. Under this mapping, the requirement that all

stabilizer generators commute becomes

$$H_X H_Z^T + H_Z H_X^T = 0. \tag{3.42}$$

Conversely, a $[2n, n+k]$ linear binary code $\mathcal{C}$ with a parity-check matrix H satisfying this constraint can be used to define a stabilizer $\mathcal{S}$. Technically, this only specifies $\tilde{\mathcal{S}}$; however, as previously outlined, it is $\tilde{\mathcal{S}}$ that dictates the effect of an error on a stabilizer code, which means that the 2^{n-k} stabilizers corresponding to $\tilde{\mathcal{S}}$ will all have the same error correction properties (the codes corresponding to each such stabilizer actually form a partition of $(\mathbb{C}^2)^{\otimes n}$ [77, 78]). Without loss of generality, we can therefore map $\tilde{\mathcal{S}}$ to a particular stabilizer $\mathcal{S}$ by arbitrarily selecting a phase factor of $+1$ for all the generators.

A subclass of stabilizer codes are the Calderbank-Shor-Steane (CSS) codes [52, 53], which have a binary representation of the form

$$H = \left(\begin{array}{c|c} \tilde{H}_X & 0 \\ 0 & \tilde{H}_Z \end{array}\right). \tag{3.43}$$

For such codes, the commutation condition of Eq. (3.42) becomes $\tilde{H}_Z \tilde{H}_X^T = 0$, which is satisfied when $\mathcal{C}_X^\perp \subseteq \mathcal{C}_Z$, where $\mathcal{C}_X$ and $\mathcal{C}_Z$ are classical codes defined by the parity-check matrices $\tilde{H}_X$ and $\tilde{H}_Z$, respectively. If $\mathcal{C}_X = \mathcal{C}_Z$, then this reduces to $\mathcal{C}_X^\perp \subseteq \mathcal{C}_X$, in which case, the CSS code is called dual containing (DC).

As previously mentioned, the decoding problem of Eq. (3.38) is essentially the same as the classical decoding problem. This link can be made more explicit by expressing errors within the binary framework using the mapping $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z} \leftrightarrow \boldsymbol{e} = (\boldsymbol{e}_X^T | \boldsymbol{e}_Z^T)^T$ (where $\boldsymbol{e}_X$, $\boldsymbol{e}_Z$, and $\boldsymbol{e}$ are column vectors for consistency with the classical case). If the generators of a stabilizer define the parity-check matrix H for the binary code $\mathcal{C}$, then the syndrome corresponding to E can be found by taking the symplectic inner product of $\boldsymbol{e}$ with each row of H, which can be written compactly as

$$\boldsymbol{z} = H \left(\begin{array}{c} \boldsymbol{e}_Z \\ \boldsymbol{e}_X \end{array}\right) = HP\boldsymbol{e}, \tag{3.44}$$

where

$$P = \left(\begin{array}{cc} 0 & I \\ I & 0 \end{array}\right). \tag{3.45}$$

With this slight modification to classical syndrome calculation, determining $\tilde{\hat{E}}_{\boldsymbol{z}}$ in Eq. (3.38) corresponds precisely to determining $\hat{\boldsymbol{e}}_{\boldsymbol{z}}$ in Eq. (3.9). Note that some authors avoid this difference in syndrome calculation by using the mapping $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z} \leftrightarrow \boldsymbol{e} = (\boldsymbol{e}_Z^T | \boldsymbol{e}_X^T)^T$ [23], which gives $\boldsymbol{z} = H\boldsymbol{e}$ as in the classical case of Eq. (3.8). For a CSS code, the syndrome associated with an error $E \propto X^{\boldsymbol{e}_X} Z^{\boldsymbol{e}_Z}$ is

$$\boldsymbol{z} = \left(\begin{array}{c} \tilde{H}_X \boldsymbol{e}_Z \\ \tilde{H}_Z \boldsymbol{e}_X \end{array}\right) = \left(\begin{array}{c} \boldsymbol{z}_Z \\ \boldsymbol{z}_X \end{array}\right). \tag{3.46}$$

This allows the X and Z components of the error to be treated separately. In particular, $\boldsymbol{e}_Z$ can be inferred from $\tilde{H}_X \boldsymbol{e}_Z = \boldsymbol{z}_Z$, while $\boldsymbol{e}_X$ can be inferred from $\tilde{H}_Z \boldsymbol{e}_X = \boldsymbol{z}_X$. However, this

approach is only guaranteed to determine the single most likely error if the X and Z components of E occur independently, which is the case for the biased XZ channel but not for the AD channel among others (see Sec. 2.2.5 for a more detailed discussion).

Elements of $\tilde{\mathcal{P}}_n$ can also be represented as elements of $\mathrm{GF}(4)^n$ according to the isomorphism [41, 15]

$$X^{\boldsymbol{u}}Z^{\boldsymbol{v}} \leftrightarrow \boldsymbol{u} + \omega\boldsymbol{v}, \tag{3.47}$$

with the product of elements in $\tilde{\mathcal{P}}_n$ corresponding to addition in $\mathrm{GF}(4)^n$. Representatives of elements in $\tilde{\mathcal{P}}_n$ commute if the trace inner product [see Eq. (3.14)] of the corresponding elements of $\mathrm{GF}(4)^n$ is zero. Utilizing this isomorphism, any $(n, 2^{n-k})_4$ additive $\mathrm{GF}(4)$ code $\mathcal{C}$ that is self-orthogonal with respect to the trace inner product can be used to define an $[[n, k]]$ stabilizer code (it is for this reason that stabilizer codes are sometimes called additive codes). Furthermore, the generators of the stabilizer $\mathcal{S}$ can be associated with the rows of a generator matrix G for $\mathcal{C}$. We can describe a stabilizer code based on properties of $\mathcal{C}$; for example, if $\mathcal{C}$ is linear and/or cyclic, then we will also call $\mathcal{S}$ (and the code $\mathcal{Q}$ it defines) linear and/or cyclic.

Similar to the classical case, when designing a stabilizer code for the depolarizing channel, the complexity of determining its FER can be avoided by instead using code distance as something of a proxy. However, for asymmetric channels, distance becomes a less accurate metric as the probability of an error occurring no longer depends only on its weight. One approach in this case is to design codes with different X and Z distances, which are called $[[n, k, d_X/d_Z]]$ codes. For these so-called asymmetric codes, d_X and d_Z are the maximal values for which there is no $\tilde{E} \in \widetilde{N(\mathcal{S})}\backslash\tilde{\mathcal{S}}$ where $E \propto X^{\boldsymbol{e}_X}Z^{\boldsymbol{e}_Z}$ and both $w(\boldsymbol{e}_X) < d_X$ and $w(\boldsymbol{e}_Z) < d_Z$. Such codes are typically constructed within the CSS framework, where $d_X = w(\mathcal{C}_Z\backslash\mathcal{C}_X^{\perp})$ and $d_X = w(\mathcal{C}_X\backslash\mathcal{C}_Z^{\perp})$ [79]. Outside of the CSS framework, where the X and Z components of an error cannot be considered separately, the distances d_X and d_Z are somewhat less meaningful and potentially not even unique. For example, the $(7, 2^6)_4$ additive cyclic code $\langle\omega 10\omega 100\rangle_{\mathrm{cyc}}$ maps to the $[[7, 1, 3]]$ cyclic stabilizer code with $\mathcal{S} = \langle XZIZXII\rangle_{\mathrm{cyc}}$, which can be considered as a $[[7, 1, 7/1]]$, $[[7, 1, 1/7]]$, or $[[7, 1, 2/3]]$ code. Some examples of asymmetric codes (for qubits) can be found in Refs. [60, 61, 62, 63, 64, 65].

3.3 Approximate FER calculation

In this paper, we wish to construct stabilizer codes that perform well on asymmetric channels. In particular, we wish to gauge their performance directly; that is, we wish to accurately determine the FER exhibited by a MAP decoder as given in Eq. (3.37). As previously noted, determining this error rate is an #P-complete problem. In this section, we therefore investigate lower complexity methods of approximating F_{MAP} and derive bounds on the relative error of these approximations.

3.3.1 Limited error set

In most cases, many of the errors in $\tilde{\mathcal{P}}_n$ occur with very low probability. It seems reasonable to assume that ignoring these low-probability errors will have little effect on the FER calculation of Eq. (3.37). In particular, suppose we only consider a subset of errors $\mathcal{E} \subset \tilde{\mathcal{P}}_n$. We can calculate an approximate FER using $\mathcal{E}$ by first partitioning it by syndrome into the sets $B_1, \ldots, B_r$, where $r \leq 2^{n-k}$. Each of these B_i is then further partitioned by equivalence up to an element of $\tilde{\mathcal{S}}$ to give the sets $A_{i1}, \ldots, A_{is}$, where $s \leq 2^{2k}$. The approximate FER is then

$$F_{\mathcal{E}} = 1 - \sum_{i=1}^{r} \max_j P(A_{ij}) = 1 - \sum_{i=1}^{r} P(\hat{A}_i), \tag{3.48}$$

where

$$\hat{A}_i = \operatorname*{argmax}_{A_{ij} \in B_i} P(A_{ij}). \tag{3.49}$$

Note that if we wish to explicitly associate a stabilizer $\mathcal{S}$ with $F_{\mathcal{E}}$, then we write $F_{\mathcal{E}}^{\mathcal{S}}$. In the best case, $\mathcal{E}$ will contain every $\hat{A}_{\boldsymbol{z}}$ in its entirety, which gives $\sum_{\boldsymbol{z}} P(\hat{A}_{\boldsymbol{z}}) = \sum_{i=1}^{r} P(\hat{A}_i)$ and hence $F_{\mathcal{E}} = F_{\text{MAP}}$. In the worst case, $\sum_{i=1}^{r} P(\hat{A}_i) = \sum_{\boldsymbol{z}} P(\hat{A}_{\boldsymbol{z}}) - [1 - P(\mathcal{E})]$, which gives $F_{\mathcal{E}} = F_{\text{MAP}} + [1 - P(\mathcal{E})]$. In general,

$$0 \leq F_{\mathcal{E}} - F_{\text{MAP}} \leq 1 - P(\mathcal{E}), \tag{3.50}$$

which leads to

$$\begin{aligned} \delta_{\mathcal{E}} &= \frac{F_{\mathcal{E}} - F_{\text{MAP}}}{F_{\text{MAP}}} \\ &\leq \frac{1 - P(\mathcal{E})}{F_{\text{MAP}}} \\ &\leq \frac{1 - P(\mathcal{E})}{F_{\mathcal{E}} - [1 - P(\mathcal{E})]} \qquad (3.51) \\ &= \Delta_{\mathcal{E}}. \qquad (3.52) \end{aligned}$$

This bound $\Delta_{\mathcal{E}}$ on the relative error $\delta_{\mathcal{E}}$ in the approximate FER calculation is of practical use as it does not require any knowledge of F_{MAP}.

There are two desirable attributes of the set $\mathcal{E} \subset \tilde{\mathcal{P}}_n$ used to calculate $F_{\mathcal{E}}$. The first of these, which follows from Eq. (3.51), is for $1 - P(\mathcal{E})$ to be less than some predetermined value as this affects the accuracy of $F_{\mathcal{E}}$. The second is for $|\mathcal{E}|$ to be small as this reduces the complexity of calculating $F_{\mathcal{E}}$. It is possible to construct such a set without enumerating $\tilde{\mathcal{P}}_n$ in its entirety by exploiting the independence of error components, which means that the probability of an error occurring depends only on the number of I, X, Y, and Z components it contains. Explicitly, the probability of some error $\tilde{E} \in \tilde{\mathcal{P}}_n$ occurring is

$$P(\tilde{E}) = \prod_{\sigma \in \{I,X,Y,Z\}} p_\sigma^{n(\sigma)}, \tag{3.53}$$

where $n(\sigma)$ is the number of tensor components of E that are equal to σ up to phase. Further-

more, the number of errors in $\tilde{\mathcal{P}}_n$ with a given distribution of components is [80]

$$N = \frac{n!}{n(I)!n(X)!n(Y)!n(Z)!}. \tag{3.54}$$

Therefore, to construct $\mathcal{E}$, we first enumerate all of the possible combinations of $n(I)$, $n(X)$, $n(Y)$, and $n(Z)$ such that $n(I)+n(X)+n(Y)+n(Z) = n$, which is a straightforward variation of the integer partition problem [81]. These combinations are sorted in descending order according to their associated probability as given in Eq. (3.53). In an iterative process, we then work through this list of combinations, adding the N distinct errors associated with each one to $\mathcal{E}$ until we reach the desired value of $1-P(\mathcal{E})$. This construction has the added benefit of ensuring that $\mathcal{E}$ is permutation invariant, which guarantees that $F_{\mathcal{E}}$ will be the same for equivalent codes.

For the approximate error rate calculation presented in this section to be of any real use, it must be accurate even when $\mathcal{E}$ is relatively small. To demonstrate that this is in fact the case, we have first constructed 1 000 random $[[7,1]]$ codes. To produce a random stabilizer $\mathcal{S} = \langle M_1, \ldots, M_{n-k}\rangle$, we iteratively select $\tilde{M}_i = \{M_i, -M_i, iM_i, -iM_i\}$ at random from $N(\langle \tilde{M}_1, \ldots, \tilde{M}_{i-1}\rangle)\backslash\langle \tilde{M}_1, \ldots, \tilde{M}_{i-1}\rangle$ (note that we arbitrarily use a phase factor $+1$ for each M_i as outlined in Sec. 3.2.4). Our only structure constraint on $\mathcal{S}$ is that it must involve every qubit; that is, for all $1 \leq j \leq n$, there must be some $M_i^{(j)} \not\propto I$, where $M_i^{(j)}$ is the jth tensor component of M_i (if a stabilizer does not satisfy this constraint, we simply discard it and construct a new one). For biased XZ channels with $p = 0.1$, 0.01, or 0.001 and $\eta = 1$, 10, or 100, we have then determined the fraction of the 1 000 codes that yield a relative error $\delta_{\mathcal{E}} \leq 0.01$ or relative error bound $\Delta_{\mathcal{E}} \leq 0.01$ for varying $|\mathcal{E}|$. The results of this are shown in Fig. 3.1, where it can be seen that depending on the channel parameters, only 1-5% of $\tilde{\mathcal{P}}_n$ needs to be considered to yield $\delta_{\mathcal{E}} \leq 0.01$ for every code. As is to be expected, a slightly larger fraction of $\tilde{\mathcal{P}}_n$ is required to ensure a relative error bound of $\Delta_{\mathcal{E}} \leq 0.01$; however, in every case, this can still be achieved by only considering between 1-10% of $\tilde{\mathcal{P}}_n$. Interestingly, for higher p, increasing η reduces the number of errors that need to be considered, while for lower p, this trend is reversed. Figure 3.2 shows the results of a similar analysis for codes with $5 \leq n \leq 7$ and $1 \leq k \leq 3$ on a biased XZ channel with $p = 0.01$ and $\eta = 10$. It can be seen that increasing k for fixed n reduces the fraction of errors that must be considered, which makes sense given that encoding a larger number of qubits will lead to a higher FER. Furthermore, increasing n for fixed k reduces the fraction of errors that need to be considered, which bodes well for the analysis of longer codes. We note that changing p and/or η has little effect on these observations.

3.3.2 Most likely error

We now consider the decoder of Eq. (3.38) that determines the single most likely error given a syndrome measurement, which has an error rate as given in Eq. (3.39). Note that $F_{\text{MAP-SE}}$ is simpler to calculate than F_{MAP} as it does not require a complete partitioning of $\tilde{\mathcal{P}}_n$ to form $\tilde{\mathcal{P}}_n/\tilde{\mathcal{S}}$. When using $F_{\text{MAP-SE}}$ as an approximation of F_{MAP}, the best-case scenario is that the most likely coset $\hat{A}_{\boldsymbol{z}}$ will contain $\hat{\tilde{E}}_{\boldsymbol{z}}$ for every $\boldsymbol{z}$, which gives $F_{\text{MAP-SE}} = F_{\text{MAP}}$. In the worst-case scenario, two things will occur. Firstly, the probability distributions over every $\hat{A}_{\boldsymbol{z}}$ will be

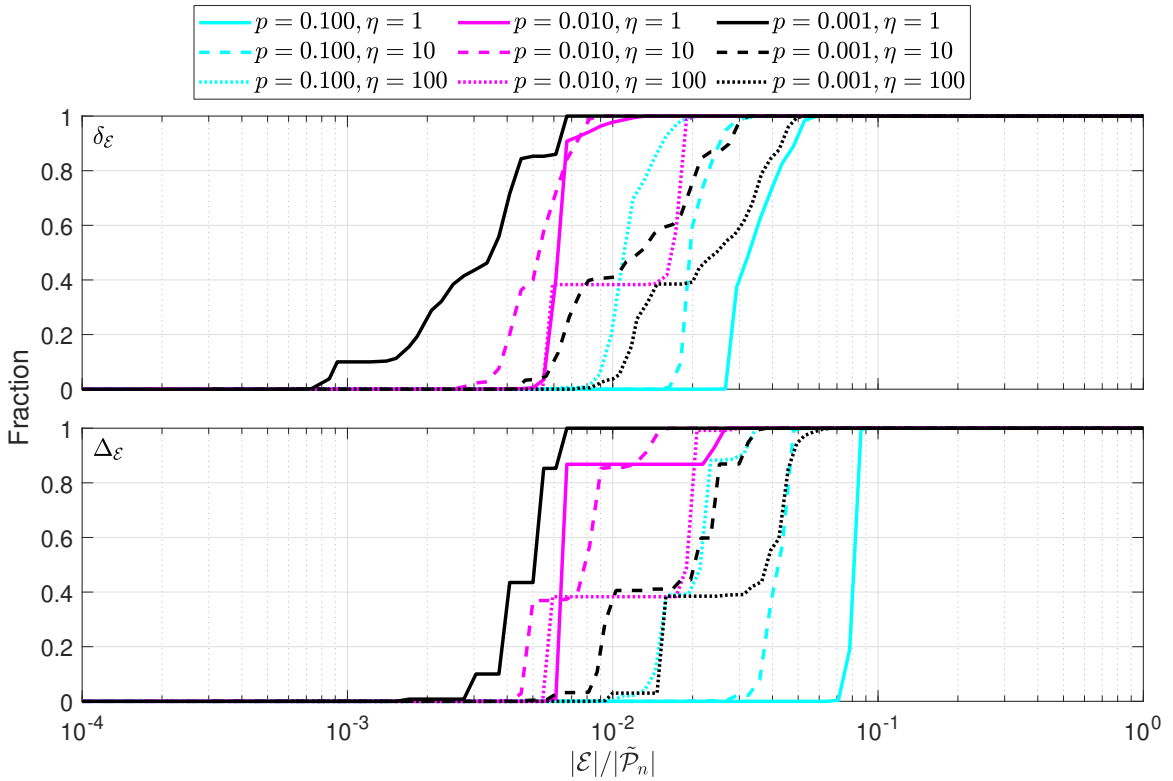

Figure 3.1: The fraction of 1 000 randomly generated $[[7, 1]]$ codes that yield a relative error $\delta_\mathcal{E} \leq 0.01$ or relative error bound $\Delta_\mathcal{E} \leq 0.01$ for varying $|\mathcal{E}|$ and biased XZ channel parameters.

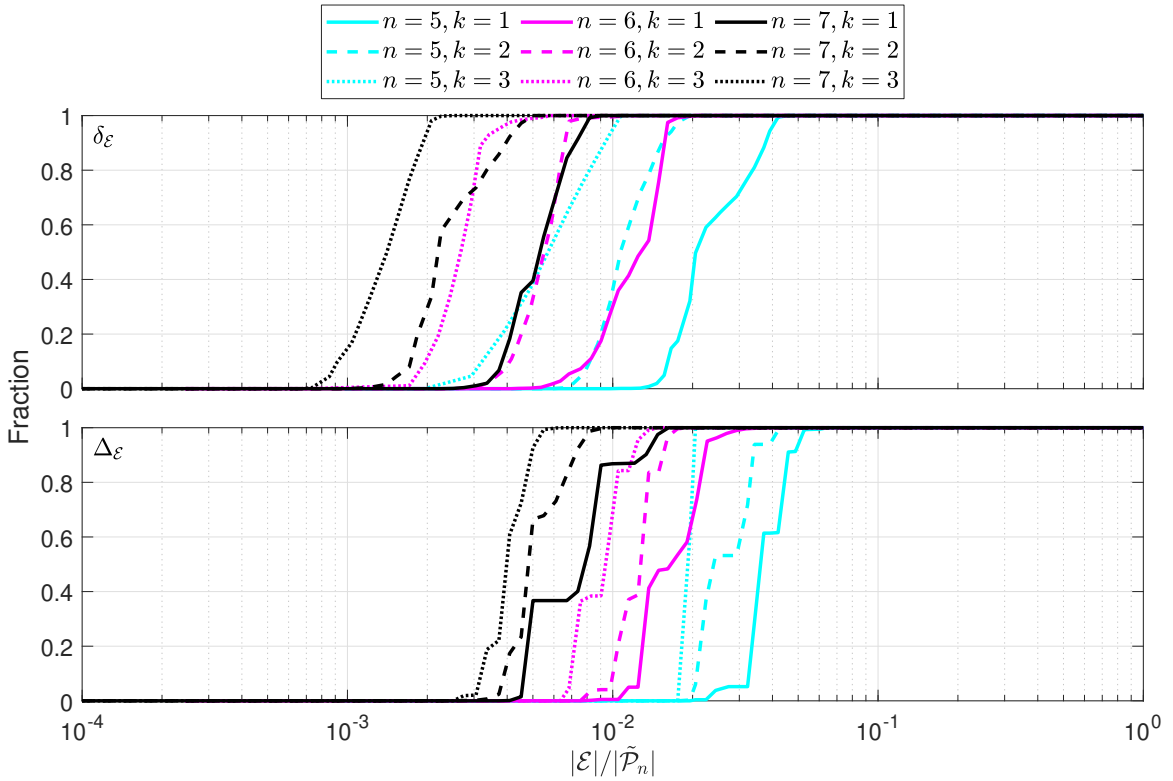

Figure 3.2: The fraction of 1 000 randomly generated $[[5 \leq n \leq 7, 1 \leq k \leq 3]]$ codes that yield a relative error $\delta_\mathcal{E} \leq 0.01$ or relative error bound $\Delta_\mathcal{E} \leq 0.01$ for a biased XZ channel ($p = 0.01$ and $\eta = 10$) and varying $|\mathcal{E}|$.

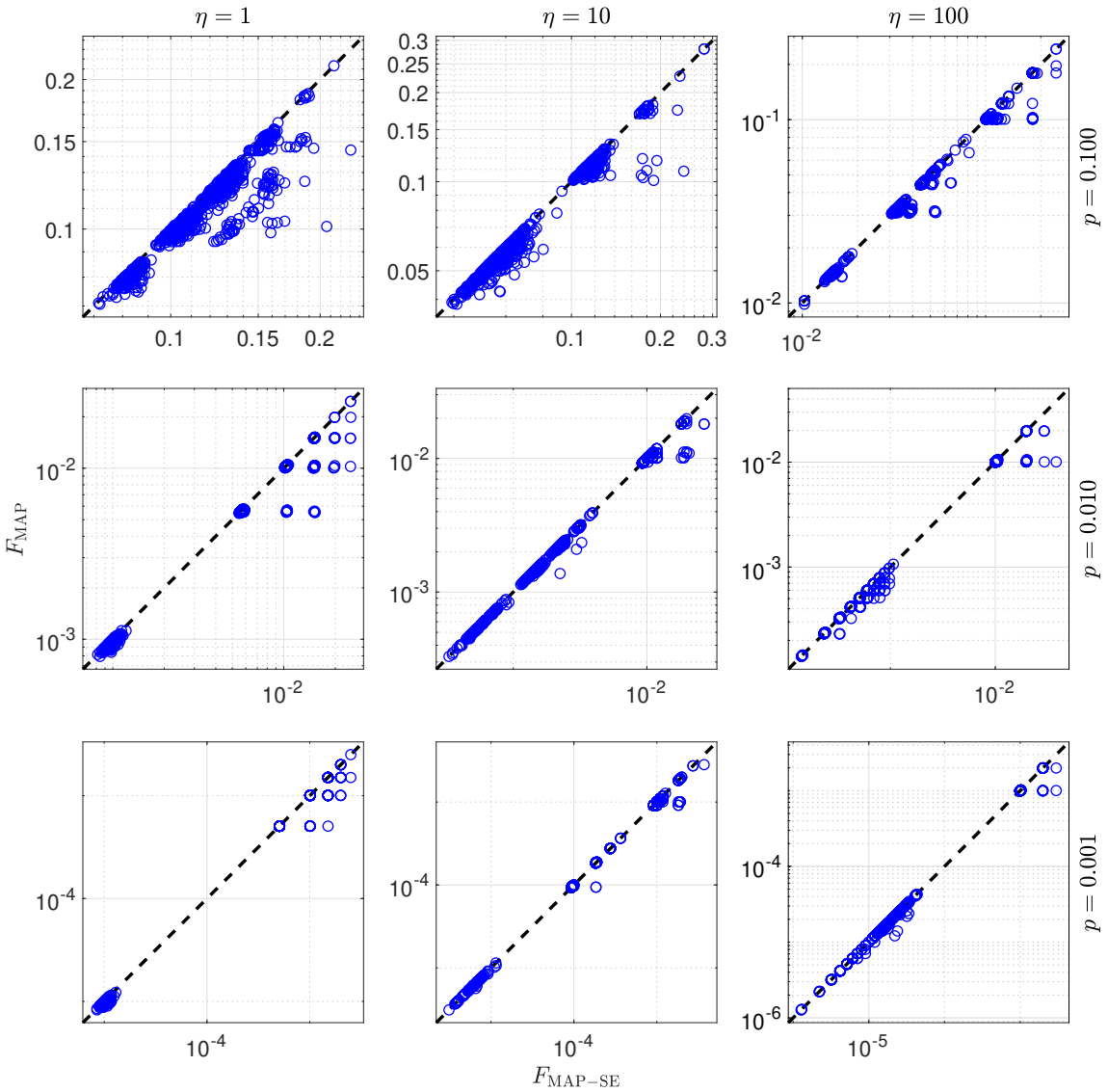

Figure 3.3: F_{MAP} versus $F_{\mathrm{MAP-SE}}$ for 1 000 random $[[7, 1]]$ codes on biased XZ channels with varying parameters. The dotted lines give $F_{\mathrm{MAP}} = F_{\mathrm{MAP-SE}}$.

uniform; that is, $P(\hat{A}_{\boldsymbol{z}})/|\tilde{\mathcal{S}}| = P(\hat{A}_{\boldsymbol{z}})/2^{n-k}$ for all $\boldsymbol{z}$. Secondly, the distributions over every $\tilde{\hat{E}}_{\boldsymbol{z}}\tilde{\mathcal{S}}$ will be sharply peaked without $P(\tilde{\hat{E}}_{\boldsymbol{z}}\tilde{\mathcal{S}})$ being large; that is, for every $\boldsymbol{z}$, $P(\tilde{\hat{E}}_{\boldsymbol{z}}) = P(\hat{A}_{\boldsymbol{z}})/2^{n-k}+\varepsilon$ and $P(\tilde{\hat{E}}_{\boldsymbol{z}}\tilde{\mathcal{S}}\backslash\tilde{\hat{E}}_{\boldsymbol{z}}) = \varepsilon'$ for some small $\varepsilon, \varepsilon' \geq 0$. In general, it is therefore the case that

$$F_{\mathrm{MAP}} \leq F_{\mathrm{MAP-SE}} < 1 - \frac{1 - F_{\mathrm{MAP}}}{2^{n-k}}. \tag{3.55}$$

This upper bound on $F_{\mathrm{MAP-SE}}$ is very loose, and in practice, $F_{\mathrm{MAP-SE}}$ tends to be quite close to F_{MAP}. To demonstrate this, we have again constructed 1 000 random $[[7, 1]]$ codes. For each code, we have then determined both F_{MAP} and $F_{\mathrm{MAP-SE}}$ for the same nine biased XZ channel parameter combinations considered in Sec. 3.3.1 ($p = 0.1$, 0.01, or 0.001 and $\eta = 1$, 10, or 100). The results of this are shown in Fig. 3.3. Especially for the codes yielding a low F_{MAP}, which are the codes of greatest interest, it can be seen that the difference between $F_{\mathrm{MAP-SE}}$ and F_{MAP} is often negligible.

$F_{\mathrm{MAP-SE}}$ can itself be approximated using a limited error set $\mathcal{E}$. We call this approximation

$F_{\mathcal{E}-\mathrm{SE}}$, and it can be calculated in much the same manner as $F_{\mathcal{E}}$. Again, $\mathcal{E}$ is first partitioned by syndrome to give $B_1, \ldots, B_r$. For each $1 \leq i \leq r$, we then determine the most likely error $\tilde{\hat{E}}_i \in B_i$, which we use to define $\hat{A}_i = \{\tilde{E} \in B_i : \tilde{\hat{E}}_i \tilde{E} \in \tilde{\mathcal{S}}\}$. With this altered definition of $\hat{A}_i$, $F_{\mathcal{E}-\mathrm{SE}}$ is given by the right-hand side of Eq. (3.48). Furthermore, the relative error bound of Eq. (3.51) also holds for $F_{\mathcal{E}-\mathrm{SE}}$ with respect to $F_{\mathrm{MAP}-\mathrm{SE}}$. We emphasize that $F_{\mathcal{E}-\mathrm{SE}}$ can be calculated faster than $F_{\mathcal{E}}$ as there is no need to fully partition each B_i.

3.3.3 Most likely error only

As outlined in Sec. 3.2.4, the single most likely error decoder for an $[[n, k]]$ stabilizer code can be viewed as a decoder for an associated $[2n, n+k]$ classical code $\mathcal{C}$. However, the calculation of $F_{\mathrm{MAP}-\mathrm{SE}}$ as in Eq. (3.39) is more complicated than determining the FER of a classical MAP decoder as the cosets $\tilde{\hat{E}}_{\boldsymbol{z}}\tilde{\mathcal{S}}$ still need to be enumerated. If we ignore the coset nature of the error correction, then we get

$$F_{\mathrm{MAP}-\mathrm{SEO}} = 1 - \sum_{\boldsymbol{z} \in \mathrm{GF}(2)^{n-k}} P(\tilde{\hat{E}}_{\boldsymbol{z}}), \tag{3.56}$$

where "SEO" stands for "single error only." Note that this is exactly the FER of the classical decoder for $\mathcal{C}$ as in Eq. (3.10). Given the nature of the assumptions leading to Eq. (3.55), it also holds for $F_{\mathrm{MAP}-\mathrm{SEO}}$. Again, it is a very loose upper bound, and as can be seen in Fig. 3.4, $F_{\mathrm{MAP}-\mathrm{SEO}}$ does tend to be somewhat close to F_{MAP}. In particular, it can be seen that the codes yielding a minimal value of $F_{\mathrm{MAP}-\mathrm{SEO}}$ also often yield a near-minimal value of F_{MAP}.

$F_{\mathrm{MAP}-\mathrm{SEO}}$ can also be approximated using a limited error set to yield $F_{\mathcal{E}-\mathrm{SEO}}$. This involves first partitioning $\mathcal{E}$ to form $B_1, \ldots, B_r$ and then determining the most likely error $\tilde{\hat{E}}_i$ in B_i. By defining $\hat{A}_i = \tilde{\hat{E}}_i$, $F_{\mathcal{E}-\mathrm{SEO}}$ is also given by the right-hand side of Eq. (3.48). Note that as no partitioning of each B_i is required, calculating $F_{\mathcal{E}-\mathrm{SEO}}$ is less complex than calculating $F_{\mathcal{E}-\mathrm{SE}}$ (or, indeed, $F_{\mathcal{E}}$). The upper bound on relative error given in Eq. (3.51) again holds for $F_{\mathcal{E}-\mathrm{SEO}}$ with respect to $F_{\mathrm{MAP}-\mathrm{SEO}}$. Assuming that $\mathcal{E}$ contains the most likely errors in $\tilde{\mathcal{P}}_n$, which is the case for the construction given in Sec. 3.3.1, we can derive another simple bound. In particular, if $\mathcal{E}$ contains errors corresponding to r different syndromes, then an error $\tilde{E}' \notin \mathcal{E}$ yielding one of the other $2^{n-k}-r$ possible syndromes must have probability $P(\tilde{E}') \leq \min_{\tilde{E} \in \mathcal{E}} P(\tilde{E})$ (as otherwise it would be an element of $\mathcal{E}$). This gives

$$F_{\mathcal{E}-\mathrm{SEO}} - F_{\mathrm{MAP}-\mathrm{SEO}} \leq (2^{n-k} - r) \min_{\tilde{E} \in \mathcal{E}} P(\tilde{E}) = \alpha, \tag{3.57}$$

which leads to a combined bound on the relative error of

$$\begin{aligned} \delta_{\mathcal{E}-\mathrm{SEO}} &= \frac{F_{\mathcal{E}-\mathrm{SEO}} - F_{\mathrm{MAP}-\mathrm{SEO}}}{F_{\mathrm{MAP}-\mathrm{SEO}}} \\ &\leq \frac{\min[1 - P(\mathcal{E}), \alpha]}{F_{\mathcal{E}-\mathrm{SEO}} - \min[1 - P(\mathcal{E}), \alpha]} \qquad (3.58) \\ &= \Delta_{\mathcal{E}-\mathrm{SEO}}. \qquad (3.59) \end{aligned}$$

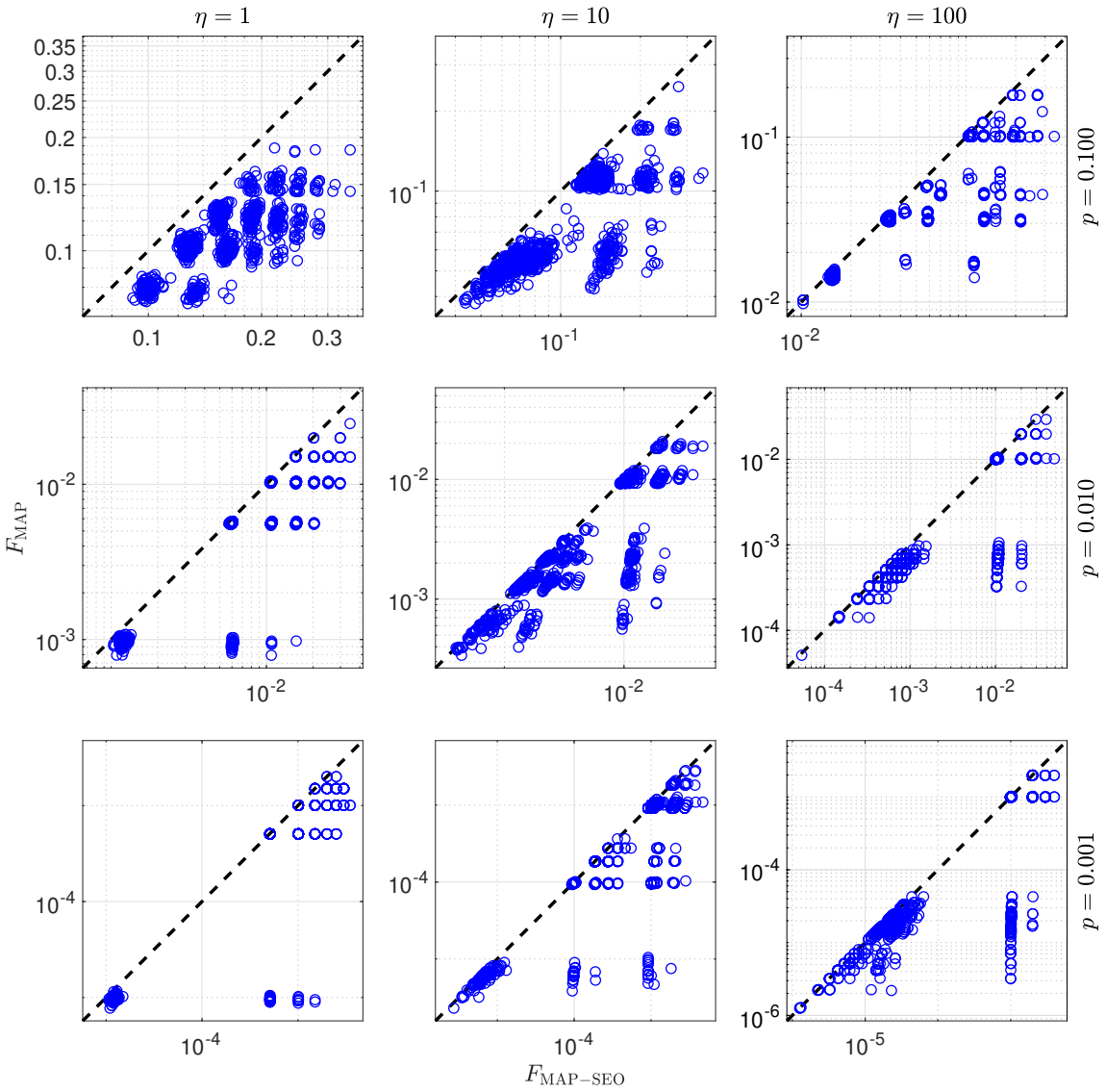

Figure 3.4: F_{MAP} versus $F_{\mathrm{MAP-SEO}}$ for 1 000 random $[[7,1]]$ codes on biased XZ channels with varying parameters. The dotted lines give $F_{\mathrm{MAP}} = F_{\mathrm{MAP-SEO}}$.

3.4 Code performance

In this section, we employ the approximate FER calculation methods outlined in Sec. 3.3 to investigate the performance of various families of codes on biased XZ and AD channels. There is a particular focus on the performance of cyclic codes as it has previously been shown that a $[[7,1,3]]$ cyclic code with $\mathcal{S} = \langle XZIZXII \rangle_{\mathrm{cyc}}$ performs near optimally on the biased XZ channel for a range of error probabilities and biases [29].

3.4.1 $[[7,1]]$ codes

To demonstrate our approach, we first consider the case of $[[7,1]]$ codes. We have constructed all of the $[[7,1]]$ cyclic codes by enumerating the self-orthogonal additive cyclic $(7,2^6)_4$ codes as outlined in Sec. 3.2.2. There are 11 such codes, six of which are inequivalent. Following the lead

of Ref. [29], we have also constructed 10 000 random codes to serve as a point of comparison. Our random construction, as detailed in Sec. 3.3.1, differs to that of Ref. [29] in that we do not require our codes to have weight-four generators or distance $d \geq 3$. For both biased XZ and AD channels with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000, we have determined $F_{\mathcal{E}}$ for each code, ensuring that in every case, $\mathcal{E}$ is large enough to give $\Delta_{\mathcal{E}} \leq 0.01$. This can be achieved without having to construct a new $\mathcal{E}$ for every FER calculation. For some channel type (biased XZ or AD), channel parameter combination (p and η pair), and code family (random or cyclic), we first construct $\mathcal{E}$, as outlined in Sec. 3.3.1, such that $1 - P(\mathcal{E}) \leq 0.1$ and then calculate $F_{\mathcal{E}}$ for every code in the family. If $\Delta_{\mathcal{E}} > 0.01$ for any of these codes, we then add errors to $\mathcal{E}$ until $1 - P(\mathcal{E}) \leq 0.01$ and recalculate $F_{\mathcal{E}}$ for these codes. This proceeds iteratively, reducing $1 - P(\mathcal{E})$ by a factor of 10 each time, until $\Delta_{\mathcal{E}} \leq 0.01$ for every code.

For each channel type, channel parameter combination, and code family, we report two values. The first of these is simply the lowest FER of any code in the family, which can be viewed as a performance measure of the family as a whole. The second is the FER of the code that performs the best on average across all channel parameter combinations. We quantify this average performance by taking the geometric mean of a code's FERs across the associated channels. That is, we take the best code to be the one with stabilizer

$$\mathcal{S}_{\text{best}} = \underset{\mathcal{S} \in \mathcal{F}}{\operatorname{argmin}} \left(\prod_{i=1}^{N} F_{\mathcal{E}_i}^{\mathcal{S}} \right)^{1/N}, \tag{3.60}$$

where $\mathcal{F}$ is the family of stabilizers and $\mathcal{E}_i$ is the error set associated with one of the $N = 16$ channels. Figure 3.5 shows these values for the biased XZ channel. It can be seen that for every parameter combination, there is a cyclic code that performs nearly as well as the best random code. Furthermore, there is a single cyclic code that performs optimally (among the cyclic codes) on all channels. In fact, there are three such codes; however, they are all equivalent to the code with stabilizer $\mathcal{S} = \langle XZIZXII \rangle_{\text{cyc}}$. The values for the AD channel are shown in Fig. 3.6, where the code with stabilizer $\langle XZIZXII \rangle_{\text{cyc}}$ again performs optimally among the cyclic codes; however, in some cases, it is outperformed by the best random code by quite a margin, particularly at lower error probabilities (note that for consistency, we have used the same random codes for both channel types). At these low error probabilities, it can also be seen that unlike the biased XZ channel, increasing the bias does little to decrease the error rate. Interestingly, the code with stabilizer $\langle YZIZYII \rangle_{\text{cyc}}$, which is not equivalent to $\langle XZIZXII \rangle_{\text{cyc}}$, yields the same performance. This is a result of the fact that $p_X = p_Y$ for the AD channel, which means that applying the permutation $X \leftrightarrow Y$ to a code's stabilizer on any subset of qubits has no effect on its performance.

Note that the relative error of a geometric mean of FERs, such as the one in Eq. (3.60), is bounded by the relative error of the least accurate individual FER. This follows from

$$\left(\prod_{i=1}^{N} F_{\mathcal{E}_i} \right)^{1/N} \leq \left[\prod_{i=1}^{N} (1 + \Delta_{\mathcal{E}_i}) F_{\text{MAP}_i} \right]^{1/N} \leq \max_i (1 + \Delta_{\mathcal{E}_i}) \left(\prod_{i=1}^{N} F_{\text{MAP}_i} \right)^{1/N}, \tag{3.61}$$

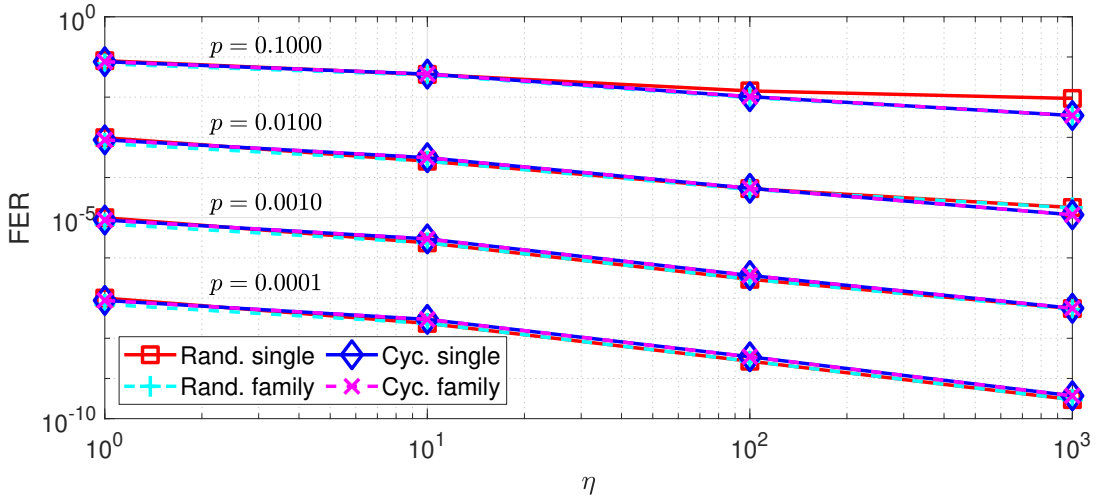

Figure 3.5: FER performance of the best cyclic and random $[[7, 1]]$ codes on biased XZ channels.

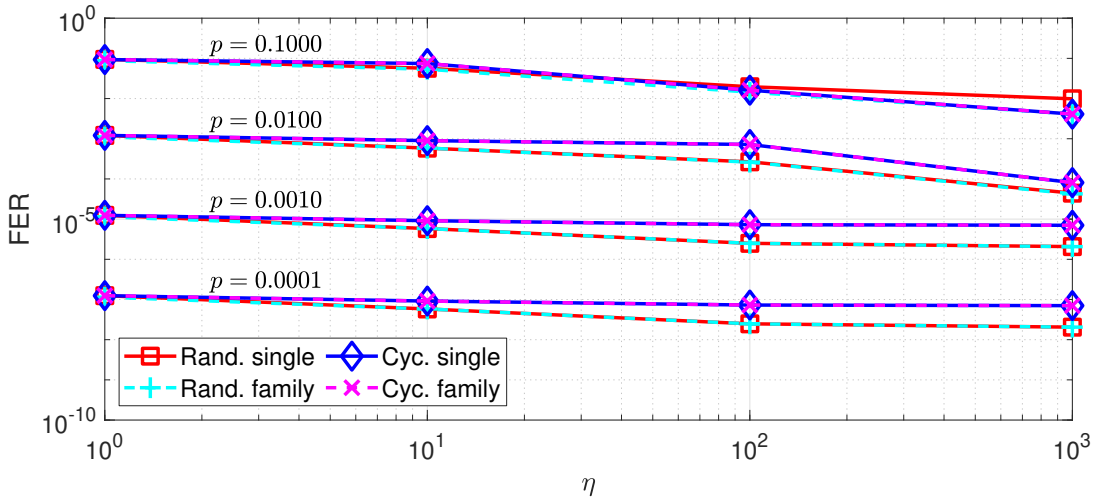

Figure 3.6: FER performance of the best cyclic and random $[[7, 1]]$ codes on AD channels.

which gives

$$\frac{\left(\prod_{i=1}^{N} F_{\mathcal{E}_i}\right)^{1/N} - \left(\prod_{i=1}^{N} F_{\mathrm{MAP}_i}\right)^{1/N}}{\left(\prod_{i=1}^{N} F_{\mathrm{MAP}_i}\right)^{1/N}} \leq \max_i \Delta_{\mathcal{E}_i}. \tag{3.62}$$

3.4.2 Other parameters

We have repeated the analysis of Sec. 3.4.1 for codes with $5 \leq n \leq 12$ and $1 \leq k \leq 3$. For each combination of n and k, this has again begun by constructing 10 000 random codes and enumerating the cyclic stabilizer codes. The number of these cyclic codes is given in the first column of Table 3.1. The first value in each row gives the number of inequivalent codes, while the value in brackets gives the total number of distinct codes. Note that for odd n, the number of distinct codes we report is consistent with Ref. [66]. To the best of our knowledge, neither the number of distinct codes with even n or the number of inequivalent codes with any n has previously been published (Ref. [67] does give total number of distinct $[[n, k \leq n]]$ cyclic codes,

Table 3.1: The number of inequivalent (distinct) $[[n, k]]$ cyclic codes, single-generator cyclic codes, cyclic codes with weight-four generators, cyclic CSS codes, dual-containing CSS codes, and linear cyclic codes.

$[[n, k]]$	Cyc.	One gen.	$w = 4$	CSS	DC CSS	Lin.
$[[5, 1]]$	4 (5)	4 (5)	4 (5)	2 (2)	0	1 (2)
$[[5, 2]]$	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
$[[5, 3]]$	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
$[[6, 1]]$	21 (21)	18 (18)	15 (15)	6 (6)	0	0 (0)
$[[6, 2]]$	35 (42)	30 (36)	17 (21)	9 (9)	2	2 (3)
$[[6, 3]]$	12 (15)	12 (15)	3 (6)	4 (4)	0	0 (0)
$[[7, 1]]$	6 (11)	5 (9)	6 (11)	3 (4)	1	1 (2)
$[[7, 2]]$	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
$[[7, 3]]$	15 (54)	15 (54)	0 (0)	4 (8)	0	0 (0)
$[[8, 1]]$	57 (87)	30 (48)	24 (33)	8 (8)	0	0 (0)
$[[8, 2]]$	46 (79)	27 (48)	19 (25)	7 (7)	3	1 (1)
$[[8, 3]]$	33 (63)	21 (48)	12 (15)	6 (6)	0	0 (0)
$[[9, 1]]$	15 (27)	15 (27)	9 (21)	4 (4)	1	0 (0)
$[[9, 2]]$	15 (27)	15 (27)	0 (0)	4 (4)	0	0 (0)
$[[9, 3]]$	5 (9)	5 (9)	3 (3)	2 (2)	1	0 (0)
$[[10, 1]]$	42 (63)	39 (60)	21 (33)	6 (6)	0	0 (0)
$[[10, 2]]$	14 (21)	13 (20)	11 (15)	3 (3)	6	2 (3)
$[[10, 3]]$	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
$[[11, 1]]$	9 (33)	9 (33)	9 (33)	2 (2)	2	0 (0)
$[[11, 2]]$	0 (0)	0 (0)	0 (0)	0 (0)	0	0 (0)
$[[11, 3]]$	0 (0)	0 (0)	0 (0)	0 (0)	3	0 (0)
$[[12, 1]]$	300 (465)	162 (288)	51 (75)	20 (20)	0	0 (0)
$[[12, 2]]$	536 (768)	288 (432)	65 (81)	35 (35)	11	2 (3)
$[[12, 3]]$	312 (528)	198 (360)	27 (30)	26 (26)	0	0 (0)

but it does not include the number for each specific k). Note that in some cases, there are no cyclic codes.

For each channel type, code family, and pair of n and k, we report two values. The first of these is the geometric mean of the FERs for the single best code as defined in Eq. (3.60); that is,

$$\lambda = \min_{\mathcal{S} \in \mathcal{F}} \left(\prod_{i=1}^{N} F_{\mathcal{E}_i}^{\mathcal{S}} \right)^{1/N}. \tag{3.63}$$

The second value is the geometric mean of the minimum FERs of all codes in a family for each channel; that is,

$$\mu = \left(\prod_{i=1}^{N} \min_{\mathcal{S} \in \mathcal{F}} F_{\mathcal{E}_i}^{\mathcal{S}} \right)^{1/N}, \tag{3.64}$$

which can again be viewed as a performance measure of the family as a whole. Figure 3.7 shows these values for the biased XZ channel. It can be seen that for both the random and cyclic codes, there is typically a single code that performs nearly as well as the family as a whole across the 16 different channels considered. Furthermore, when $[[n, k]]$ cyclic codes exist, there is often one that performs as well as or better than the best random code we have created. In fact, for $n \geq 9$ and $k = 1$, the best cyclic codes significantly outperform the best random codes.

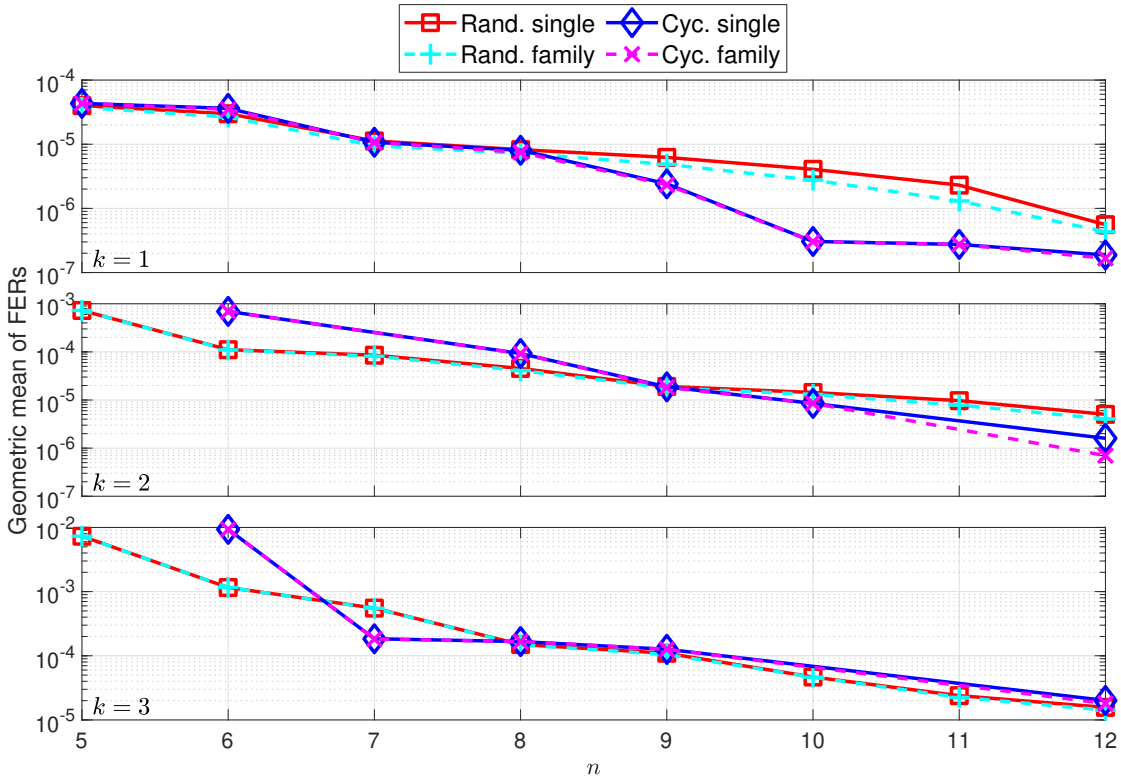

Figure 3.7: The geometric mean of FERs for codes on biased XZ channels with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000.

The results for the AD channel are given in Fig. 3.8. Again, where $[[n, k]]$ cyclic codes exist, they typically perform favorably compared to the random codes. However, any performance advantages over the random codes are less pronounced than in the biased XZ case.

Generators for the best cyclic codes on both the biased XZ and AD channels can be found in Table 3.2 (for reference, we also give their distances). In particular, we list generators for all codes that yield a geometric mean of FERs within 1% of the minimum value we have observed (these are all codes that could conceivably be optimal within our margin of error). There are a few notable properties of these codes. The first of these is that they can all be expressed using a single generator. While, as shown in the second column of Table 3.1, a large number of codes have such a representation, this is still a somewhat surprising result. It can also be seen that in nearly every case, there are codes that perform well for both the biased XZ and AD channels (the only exceptions to this are the $[[6, 1]]$, $[[6, 2]]$, and $[[10, 2]]$ cases). A third property of note is that the codes for the AD channel typically come in pairs, one being an $X \leftrightarrow Y$ permuted version of the other. This is to be expected given the partial channel symmetry outlined in Sec. 3.4.1. The only two exceptions to this are the $[[5, 1]]$ and $[[10, 2]]$ cases, where the single code given is invariant under an $X \leftrightarrow Y$ permutation (up to a permutation of qubit labels).

3.4.3 Hill climbing

The results of Sec. 3.4.2, particularly those for $[[n \geq 9, 1]]$ codes on the biased XZ channel, show that constructing 10 000 random codes is not a reliable way of finding a good code for larger n.

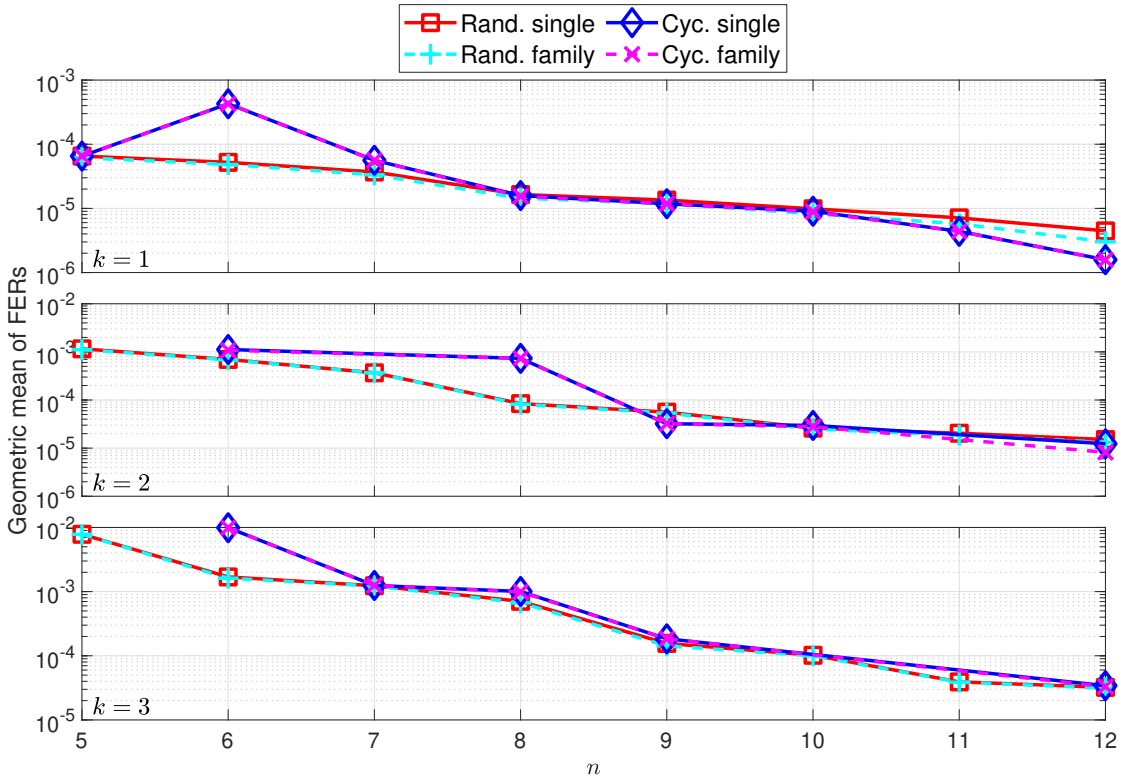

Figure 3.8: The geometric mean of FERs for codes on AD channels with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000.

One approach to find better codes would be to simply increase the size of the random search. However, even with the reduction in error set size afforded by the approach of Sec. 3.3.1, this quickly becomes computationally impractical. As such, we need a more efficient search strategy. To achieve this, we use the observation of Sec. 3.3.3 that codes yielding a low $F_{\mathcal{E}-\mathrm{SEO}}$ tend to also yield a low $F_{\mathcal{E}}$ (recall that $F_{\mathcal{E}} \leq F_{\mathcal{E}-\mathrm{SEO}}$). We can therefore reduce the search to finding codes that yield a low $F_{\mathcal{E}-\mathrm{SEO}}$, which is beneficial as it is typically several orders of magnitude faster to calculate $F_{\mathcal{E}-\mathrm{SEO}}$ than it is to calculate $F_{\mathcal{E}}$ to the same accuracy.

We start by considering the problem of finding codes that perform well for a single channel parameter combination. That is, we want to find a stabilizer $\mathcal{S}$ that yields a low $F^{\mathcal{S}}_{\mathcal{E}-\mathrm{SEO}}$. We have found a simple hill-climbing search strategy to be effective at this. This involves first constructing $\mathcal{S}$ at random. $\mathcal{S}$ is then mutated (modified) somehow to produce $\mathcal{S}'$, and if $F^{\mathcal{S}'}_{\mathcal{E}-\mathrm{SEO}} \leq F^{\mathcal{S}}_{\mathcal{E}-\mathrm{SEO}}$, then $\mathcal{S}$ is replaced with $\mathcal{S}'$. This process repeats for a predetermined number of iterations, after which we calculate $F^{\mathcal{S}}_{\mathcal{E}}$ to quantify the actual performance of the code. Similar to the random search outlined in Sec. 3.4.1, we ensure that the relative error of all approximate FER calculations is less than 1%. To achieve this, we again initially construct $\mathcal{E}$ such that $1 - P(\mathcal{E}) \leq 0.1$, and if $\Delta_{\mathcal{E}-\mathrm{SEO}} > 0.01$ ($\Delta_{\mathcal{E}} > 0.01$) for any calculation of $F_{\mathcal{E}-\mathrm{SEO}}$ ($F_{\mathcal{E}}$), then we add errors to $\mathcal{E}$ to reduce $1 - P(\mathcal{E})$ by a factor of 10 and recalculate the error rate. To better explore the space of possible stabilizers, we run a number of these hill-climbing instances in parallel (this is often called hill climbing with random restarts [82]).

The choice of a mutation operator that maps $\mathcal{S}$ to $\mathcal{S}'$ is limited by the requirement that $\mathcal{S}'$ must be a stabilizer. We consider two types of mutation that satisfy this constraint. The first of these

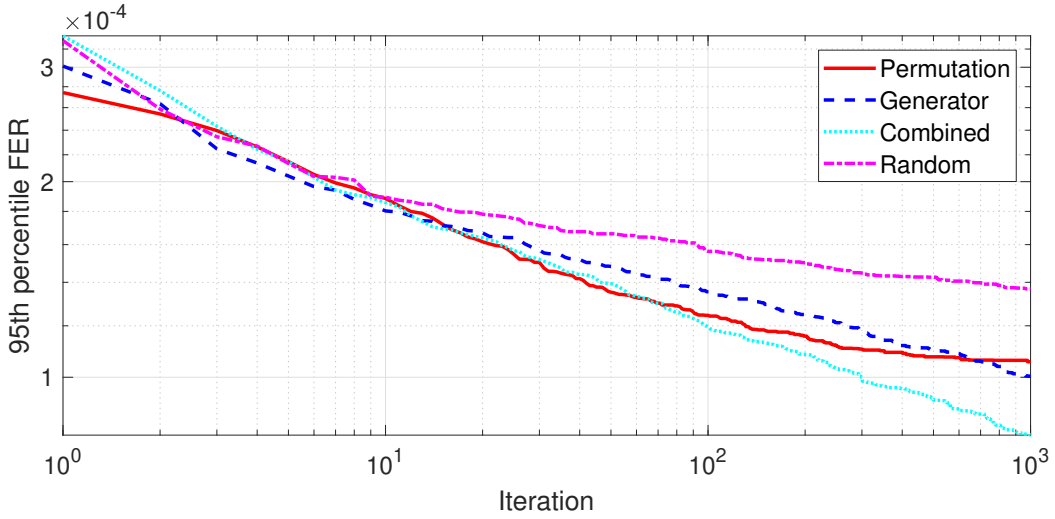

Figure 3.9: The 95th percentile $F_{\mathcal{E}-\mathrm{SEO}}$ found by 1 000 hill-climbing instances based on various mutation methods for $[[9,1]]$ codes on a biased XZ channel ($p = 0.01$ and $\eta = 10$).

involves permuting the nonidentity Pauli matrices of all stabilizer elements at any given index $1 \leq i \leq n$ with probability $1/n$. Note that these permutations correspond to a multiplication of coordinates of the associated classical GF(4) code by a nonzero scalar $\alpha \in \mathrm{GF}(4)$ followed by a possible conjugation. The second mutation method involves first removing any given generator M_i of $\mathcal{S} = \langle M_1, \ldots, M_{n-k} \rangle$ with probability $1/(n-k)$ and then adding generators as outlined in Sec. 3.3.1 to form $\mathcal{S}'$. When performing this generator mutation, we still require that all qubits are involved in the stabilizer; if this is not achieved after adding the new generators, we remove them and try again. To compare these two mutation operators, we consider $[[9,1]]$ codes on the biased XZ channel with $p = 0.01$ and $\eta = 10$. We have run 1 000 hill-climbing instances, each for a maximum of 1 000 iterations. Across all of these instances, Fig. 3.9 shows the 95th percentile $F_{\mathcal{E}-\mathrm{SEO}}$ at each iteration; that is, it shows the 50th lowest $F_{\mathcal{E}-\mathrm{SEO}}$ (we have chosen to show this value as it reflects the performance of the best codes while having less potential variance than showing the FER of the single best code). As a control, we have also tested random mutation, which involves simply creating $\mathcal{S}'$ at random (this reduces hill climbing to a random search). It can be seen that both the permutation and generator mutation outperform this random mutation, with the permutation mutation performing best initially but then tapering off somewhat. Finally, we have tested a combination of the two mutation methods (a generator mutation followed by a permutation mutation), which can be seen to perform better than either of the methods individually.

3.4.4 Multiobjective hill climbing

The results of Sec. 3.4.2 suggest that there are typically codes that perform well across a range of channel parameter combinations. We can search for such codes by building on the hill-climbing algorithm outlined in Sec. 3.4.3. In particular, instead of comparing $F^{\mathcal{S}'}_{\mathcal{E}-\mathrm{SEO}}$ to $F^{\mathcal{S}}_{\mathcal{E}-\mathrm{SEO}}$, we compute and compare the geometric means $(\prod_{i=1}^{N} F^{\mathcal{S}'}_{\mathcal{E}_i-\mathrm{SEO}})^{1/N}$ and $(\prod_{i=1}^{N} F^{\mathcal{S}}_{\mathcal{E}_i-\mathrm{SEO}})^{1/N}$ of the FERs for N channel parameter combinations. Following Eq. (3.62), we ensure that these

geometric means are accurate to within 1% by keeping each of the individual $\Delta_{\mathcal{E}_i-\mathrm{SEO}} \leq 0.01$ as outlined in Sec. 3.4.3. Again, we run a number of these hill-climbing instances in parallel, and at the end of each one, we calculate $(\prod_{i=1}^{N} F_{\mathcal{E}_i}^{\mathcal{S}})^{1/N}$. Note that for $N = 1$, this search reduces to that of Sec. 3.4.3.

We have performed such searches for the same cases considered in Sec. 3.4.2 (that is, codes with $5 \leq n \leq 12$ and $1 \leq k \leq 3$ for biased XZ and AD channels with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000). For each combination of n, k, and channel type, we have run 1 000 hill-climbing instances based on the combined generator and permutation mutation, each for 1 000 iterations. Figure 3.10 compares the performance (that is, the geometric mean of FERs) of the best codes found in this way to that of the best cyclic codes (the other values shown will be detailed in Secs. 3.4.5 to 3.4.7). It can be seen that in all but the $[[10, 1]]$ case, the best code found via hill climbing is either as good as or better than the best cyclic code. Very similar results can be seen in Fig. 3.11 for the AD channel, where the best code found via hill climbing performs as well as or better than the best cyclic code in every instance. Generators for the best codes we have found for the biased XZ and AD channels can be found in Tables 3.3 and 3.4, respectively.

3.4.5 Weight-four codes

Through slight modification of the hill-climbing algorithm, we can search for good codes that satisfy structure constraints. The first constraint we consider is the requirement that the stabilizer has a representation involving only weight-four generators; such codes are of practical interest as their syndrome measurements involve fewer qubits, and are hence less complex, than those for codes with high-weight generators. The first modification required to search for these codes, which is somewhat obvious, is to ensure the initial random stabilizer has weight-four generators. This also extends to the generator permutation; that is, any generator added to replace a removed one must also have weight four. No change to the permutation mutation is required as it preserves the weight of stabilizer elements. We compare the codes found via this constrained hill-climbing search to the cyclic codes with a weight-four generator representation. The number of such cyclic codes is given in the third column of Table 3.1, where it can be seen that they are reasonably plentiful.

The performance of the weight-four codes found via hill climbing for the biased XZ channel is shown in Fig. 3.10. It can be seen that in a lot of cases, these codes perform nearly as well as those found using unconstrained hill climbing in Sec. 3.4.4. The performance of the weight-four cyclic codes is more varied. In some cases, they are optimal (among the cyclic codes), while in others, they perform relatively poorly. Figure 3.11 shows that the performance of the weight-four codes found via hill climbing for the AD channel is somewhat mixed, ranging from outperforming the unconstrained $[[9, 1]]$ codes to performing very poorly for $k = 3$ and $n \geq 8$. The performance of the weight-four cyclic codes relative to the best unconstrained cyclic codes is much the same as for the biased XZ channel. Generators for the best weight-four codes found

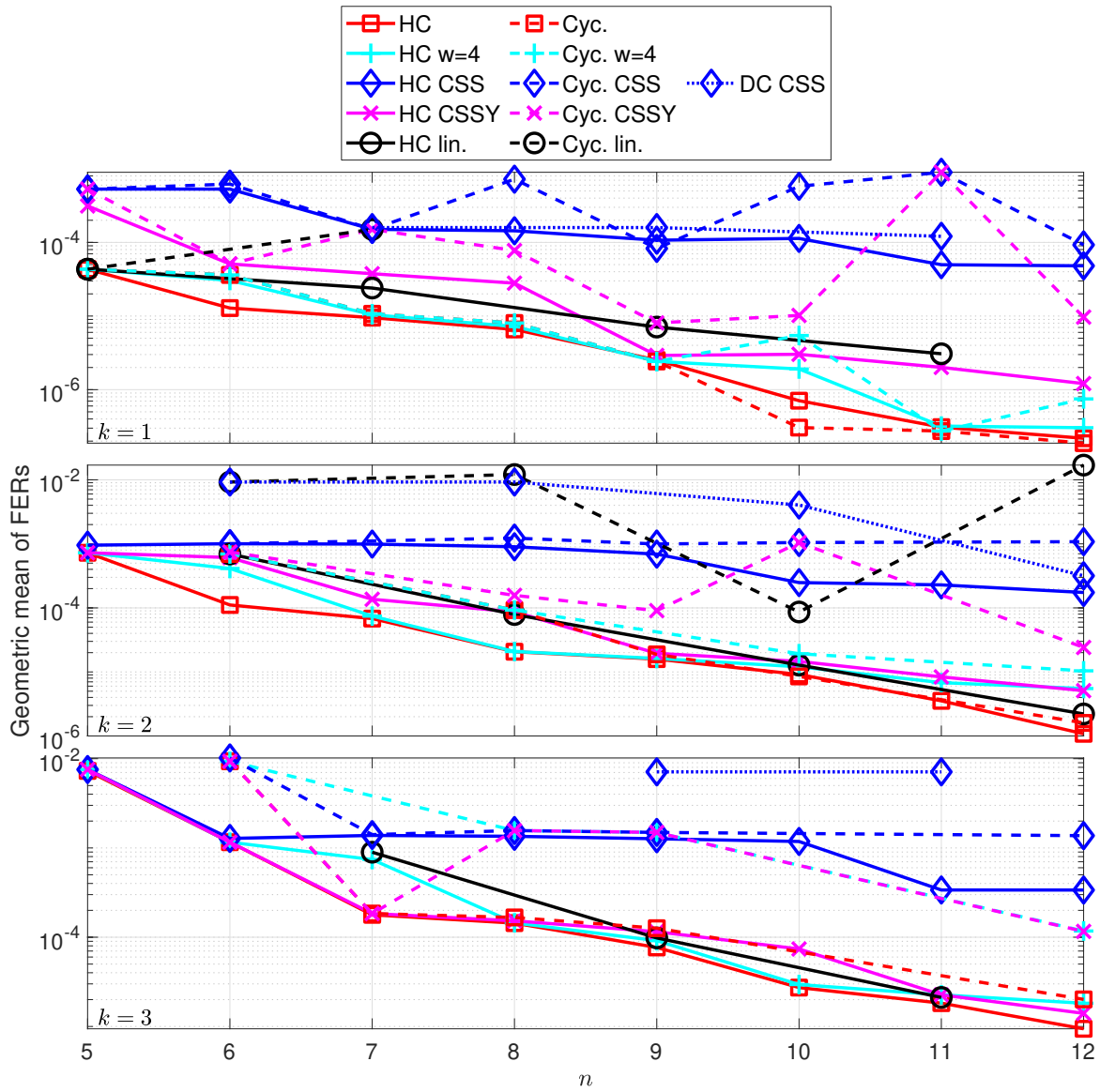

Figure 3.10: The performance (geometric mean of FERs) of the best $[[5 \leq n \leq 12, 1 \leq k \leq 3]]$ codes found via hill climbing for biased XZ channels with with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000. Also shown is the performance of the best cyclic codes and dual-containing CSS codes.

via hill climbing can be found in Tables 3.5 and 3.6, and generators for the best cyclic codes are given in Table 3.7.

3.4.6 CSS codes

We next consider CSS codes, which as outlined in Sec. 3.2.4, are codes that can be represented using generators that contain either only X or only Z matrices as their nonidentity elements. Similar to the search for weight-four codes, we must modify both the initial stabilizer construction and the generator permutation. In particular, when adding a new generator, we will select a suitable X-only element half the time and a Z-only element the other half. Another required modification is the removal of the permutation mutation as, in general, it does not map CSS

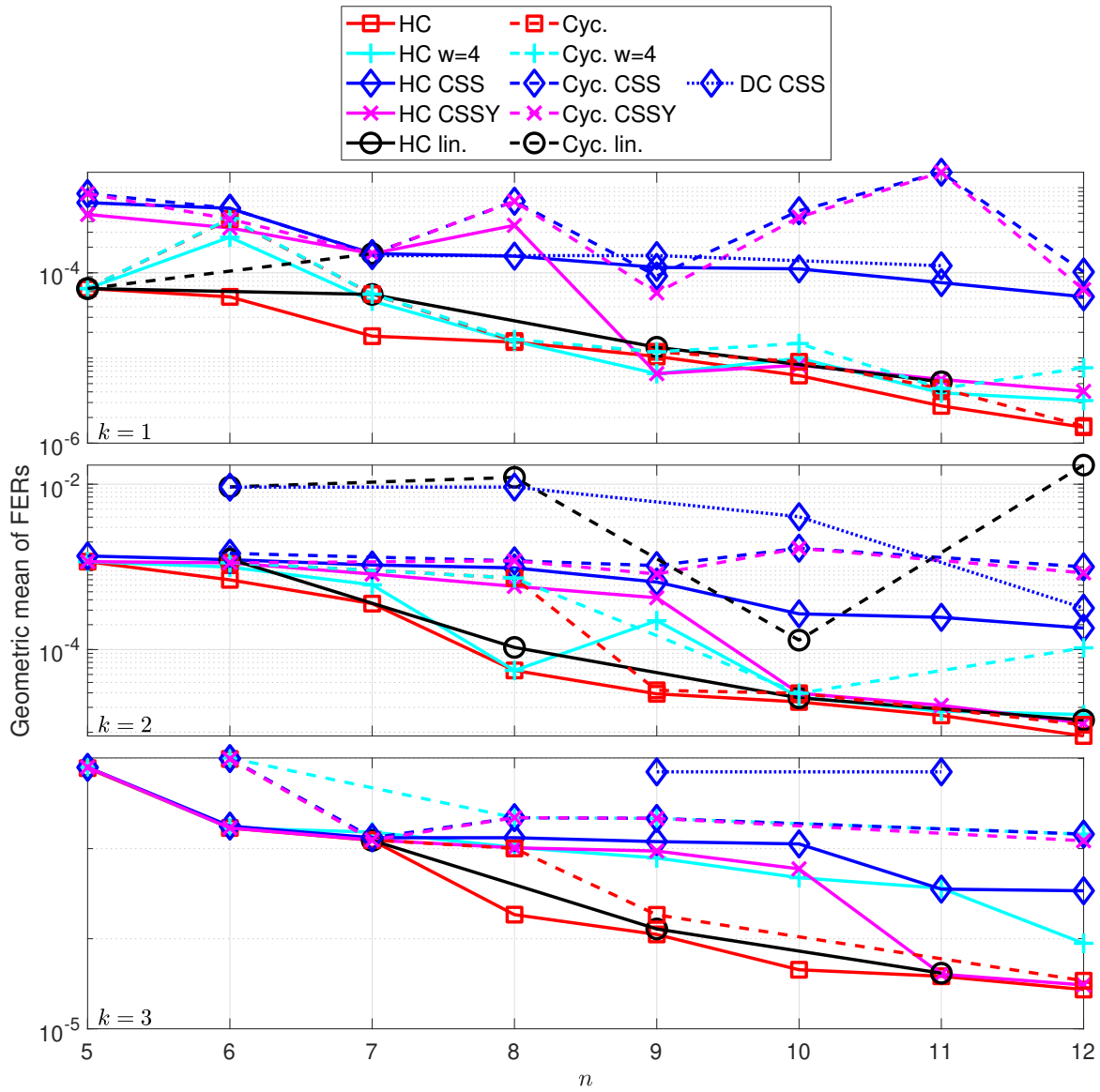

Figure 3.11: The performance (geometric mean of FERs) of the best $[[5 \leq n \leq 12, 1 \leq k \leq 3]]$ codes found via hill climbing for AD channels with with $p = 0.1$, 0.01, 0.001, or 0.0001 and $\eta = 1$, 10, 100, or 1 000. Also shown is the performance of the best cyclic codes and dual-containing CSS codes.

codes to CSS codes. We also consider cyclic CSS codes, which can be thought of in two equivalent ways. They can be viewed as codes with a binary representation where $\tilde{H}_X$ and $\tilde{H}_Z$ each correspond to a binary cyclic code. Alternatively, they can be considered in the GF(4) framework as additive cyclic codes that can be represented by a 1-only cyclic generator and/or an ω-only cyclic generator. The number of these cyclic CSS codes is given in the fourth column of Table 3.1. We also consider the family of dual-containing CSS codes to generalize the result of Ref. [29], where it was shown that the $[[7, 1, 3]]$ Steane code [83], which has

$$\tilde{H}_X = \tilde{H}_Z = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{3.65}$$

performs poorly on the biased XZ channel. We have constructed these codes by enumerating all of the inequivalent binary self-orthogonal codes using SAGEMATH [84] (recall that a generator matrix for a binary-self orthogonal code is the parity-check matrix for a dual-containing code). The number of such codes is given in the fifth column of Table 3.1. Note that there can only be an $[[n, k]]$ dual-containing CSS code if $n - k$ is even; furthermore, even when $n - k$ is even, not many of them exist for the parameters considered.

As can be seen for the biased XZ channel in Fig. 3.10, both the CSS codes found via hill climbing and the cyclic CSS codes perform poorly compared to their non-CSS counterparts. This performance can be improved by following the modification outlined in Ref. [30], which involves applying the permutation $Z \leftrightarrow Y$ to the code's generators (this is motivated by the fact that Z-only generators commute with any Z-only error, meaning that they often provide no information about an error when η is large). Given the nature of this modification, we call such codes CSSY codes. We have performed a hill-climbing search for CSSY codes, and it can be seen that they perform significantly better than the standard CSS codes; however, they are still outperformed by non-CSS codes in most instances. Similarly, while the cyclic CSSY codes perform better than the cyclic CSS codes, there is often a significant performance gap to the non-CSS cyclic codes. The dual-containing CSS codes perform poorly across the board, which can at least partially be attributed to the fact that they must have $d_X = d_Z$. Furthermore, their performance cannot be improved as they are invariant under a $Z \leftrightarrow Y$ permutation. As shown in Fig. 3.11, the results on the AD channel are similar to those for the biased XZ channel. Both the CSS codes found via hill climbing and the cyclic CSS codes perform poorly compared to the non-CSS codes. In this case, the performance gain of the CSSY codes over the CSS codes is less pronounced. A notable exception to this is the $[[9, 1]]$ case, where the best CSSY code found via hill climbing outperforms the best unrestricted code found. Somewhat surprisingly, after applying an $X \leftrightarrow Y$ permutation to the second, fourth, fifth, sixth, and ninth qubits, this code is equivalent to the best code with weight-four generators found in Sec. 3.4.5. Again, the performance of the dual-containing CSS codes is very poor compared to nearly all other codes considered. Generators for the best CSSY codes found via hill climbing can be found in Tables 3.8 and 3.9. We omit the standard CSS codes found via hill climbing and the cyclic CSS(Y) codes due to their poor performance.

3.4.7 Linear codes

The dual-containing CSS codes considered in the previous section are examples of linear stabilizer codes. An additive $(n, 2^{n-k})_4$ code $\mathcal{C}$ is linear if and only if it has a generating set of the form $\mathcal{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{(n-k)/2}, \omega\boldsymbol{b}_1, \ldots, \omega\boldsymbol{b}_{(n-k)/2}\}$. This corresponds to the stabilizer having generators of the form $\mathcal{S} = \langle M_1, \ldots, M_{(n-k)/2}, \bar{M}_1, \ldots, \bar{M}_{(n-k)/2}\rangle$, where $\bar{M}_i$ is a version of M_i that has been subjected to the permutation $(X, Y, Z) \to (Z, X, Y)$. To search for such codes, we must first modify the initial construction and generator mutations. In particular, we add or remove the generators M_i and $\bar{M}_i$ in pairs. To preserve linearity, the permutation mutation has to be restricted to permutations corresponding to a multiplication of a coordinate of $\mathcal{C}$ by ω or $\bar{\omega}$.

That is, the permutation must either be $(X, Y, Z) \to (Z, X, Y)$ or $(X, Y, Z) \to (Y, Z, X)$. We also consider linear cyclic codes, the structure of which is outlined in Sec. 3.2.2. The number of such codes is given in the sixth column of Table 3.1. Like the dual-containing CSS codes, $[[n, k]]$ linear codes can only exist for even $n-k$; furthermore, while $n-k$ is even for $[[5, 3]]$ codes, there are no linear codes with these parameters that involve every qubit.

As shown in Fig. 3.10, the linear codes found via hill climbing perform reasonably well on the biased XZ channel. The performance of the linear cyclic codes is somewhat less impressive, with there being a significant gap in performance to the more general additive cyclic codes. This can potentially be attributed to the fact that at least for the code parameters considered, there are very few linear codes. As can be seen in Fig. 3.11, the linear codes found via hill climbing for the AD channel perform better than those on the biased XZ channel, particularly in the $k = 3$ case. However, the linear cyclic codes still perform poorly. The best linear codes found via hill climbing are given in Tables 3.10 and 3.11. We omit the linear cyclic codes due to their poor performance.

3.5 Conclusion

We have shown that the error rate of an optimal stabilizer code decoder can be effectively approximated by considering only a limited subset $\mathcal{E}$ of the 4^n possible Pauli errors, and we have outlined how to construct $\mathcal{E}$ without having to enumerate all of these errors. Utilizing this approximate calculation, we have demonstrated that there are a number of $[[5 \leq n \leq 12, 1 \leq k \leq 3]]$ cyclic stabilizer codes that perform very well on both the biased XZ and AD channels across a range of error probabilities and biases. We have also shown that an indication of the performance of a stabilizer code can be obtained by considering the error rate of an associated $[2n, n+k]$ classical code. We have used this as the basis for a hill-climbing algorithm, which we have shown to be effective at optimizing codes for both of the asymmetric channels considered. Furthermore, we have demonstrated that by modifying the mutation operation of this hill-climbing algorithm, it is possible to search for highly performant codes that satisfy structure constraints. In particular, we have successfully performed searches for codes with weight-four generators, CSS(Y) codes, and linear codes.

Table 3.2: Generators and distances for the best-performing inequivalent cyclic codes on the biased XZ and AD channels. Note that each stabilizer can be expressed using a single generator; that is, each generator given corresponds to a different code. The generators of codes performing well on both channel types are given in bold.

	Biased XZ		AD	
$[[n,k]]$	Generators	d	Generators	d
$[[5,1]]$	***YZIZY***	3	***YZIZY***	3
$[[6,1]]$	*YIZZIY*	2	*XZZZZX*	2
			YZZZZY	2
$[[6,2]]$	*YZIZYI*	2	*XIZIXY*	2
			YIZIYX	2
			XZIZXY	2
			YZIZYX	2
$[[6,3]]$	***XZXXZX***	2	***XZXXZX***	2
	XZZXZZ	2	***XZZXZZ***	2
	XIYXIY	2	*YZYYZY*	2
	YZIYZI	2	*YZZYZZ*	2
$[[7,1]]$	***XZIZXII***	3	***XZIZXII***	3
			YZIZYII	3
$[[7,3]]$	***XZZZXZX***	2	***XZZZXZX***	2
	XZIIYZY	2	*YZZZYZY*	2
	YIIZYZX	2		
$[[8,1]]$	***YIIZIZZX***	3	***YIIZIZZX***	3
	ZZYIIIIY	3	*XIIZIZZY*	3
$[[8,2]]$	***YIIXIIYX***	2	***YIIXIIYX***	2
	YIZZIIXZ	2	***YIZZIIXZ***	2
	XIIYZIYY	2	***XIIYZIYY***	2
	YIIZIIYZ	2	*XIIYIIXY*	2
	YZIZIZYZ	2	*YIIXZIXX*	2
	XZZZZZXZ	2	*XIZZIIYZ*	2
$[[8,3]]$	***YIXIIYZY***	2	***YIXIIYZY***	2
	XZIIZXYY	2	***XZIIZXYY***	2
	YZIZIXYX	2	*XIYIIXZX*	2
			YZIIZYXX	2
$[[9,1]]$	***ZIZYIIIIY***	3	***ZIZYIIIIY***	3
			ZIZXIIIIX	3
$[[9,2]]$	***IZIXIZIYY***	3	***IZIXIZIYY***	3
			IZIYIZIXX	3
$[[9,3]]$	***YZZIZZYII***	3	***YZZIZZYII***	3
			XZZIZZXII	3
$[[10,1]]$	***YZIZIIZIZY***	4	***YZIZIIZIZY***	4
			XZIZIIZIZX	4
$[[10,2]]$	*YZZIIIZZYI*	2	*IYXIIIIIXY*	3
$[[11,1]]$	***IYIIZIIZIIY***	3	***IYIIZIIZIIY***	3
			IXIIZIIZIIX	3
$[[12,1]]$	***YIXIXIIIIIZX***	4	***YIXIXIIIIIZX***	4
			XIYIYIIIIIZY	4
$[[12,2]]$	***IIZZIIXZZIXY***	4	***IIZZIIXZZIXY***	4
	YXZIXIIIIIYX	4	*IIZZIIYZZIYX*	4
$[[12,3]]$	***ZZXIYIIIIYIX***	3	***ZZXIYIIIIYIX***	3
	IZZIXIZZIYXY	3	*ZZYIXIIIIXIY*	3

Table 3.3: Generators and distances for the best codes found for the biased XZ channel using hill climbing.

	$k=1$		$k=2$		$k=3$	
n	Generators	d	Generators	d	Generators	d
5	*IXXZZ* *YZYIZ* *IZZYY* *XZIZX*	3	*XXYZI* *XXZXZ* *XIIZY*	1	*XYZYZ* *IIIXX*	1
6	*IXXXYZ* *YIYIZZ* *IYYIII* *XZXIYX* *ZZXYIZ*	3	*ZIZYXY* *XXZXYX* *IYYZYZ* *XZIZYX*	2	*IIYYZY* *XXZYZZ* *XIIIYX*	1
7	*XZZXYYI* *XYYXZXY* *XIXZZZZ* *YZIZXXY* *ZYIXYIY* *YIYZYZY*	3	*IIZYXZY* *ZIXYIXY* *XIZZYXZ* *XYIXYIY* *ZXZXYII*	2	*YZZYIXX* *ZYXZIZY* *XYIXXIY* *ZZXXZXZ*	2
8	*ZIIYXIYX* *ZYYIYYXX* *IZYIYXXI* *XYXZXZII* *YIYYZIZY* *ZXXXYIII* *YXXZYYZX*	3	*YYIIIXYI* *ZIXYZZIY* *ZYXZYXXX* *IZXYIZYI* *YZZZIZIX* *IXXIZIXI*	2	*XZZZYYIY* *YZIYXIXX* *XZXXYYYI* *IYZYIYZI* *YYZZZXIY*	2
9	*YXYXIIIII* *YZIYIZIZI* *XYIIYIIIX* *IXIXIZZZZ* *XYXYIXZXI* *XIXXXYYYY* *XXYYYYXXZ* *YZXXZYZIZ*	3	*XIZZZZXYX* *IIZIYYIIX* *IXYYZZYYZ* *IIZXXXXZI* *ZZXIXXIXY* *YXXZIZIYI* *ZYYYIXZXI*	2	*ZIZZXXYIZ* *ZIIYXIIYY* *IYZXXXIIX* *ZXZYYZZZI* *YZYYIYYXI* *YYZIZZXII*	2
10	*XYYXYIXYXX* *XYYYIZXZYY* *ZIXZXYIZYI* *YXYYXXIIIX* *IZZZIIZYYZ* *XXZXYZYXYZ* *XXIZXIXXIX* *ZXXIYYIXIX* *XZXZYYXXXI*	3	*ZXZIXZIIIY* *YIIXYZIZII* *XIYIIIZXZY* *ZIXIYYIIYX* *XZZYIXIYXX* *YYIXYXYXII* *ZZYZZXXZXI* *ZXZYXZYYZI*	3	*ZZXXIIXZZY* *IYZYZZYIZY* *YZIYXZZZIX* *ZXYZIIYXXY* *ZYXIXZIYZY* *XZYXYYIYXY* *YYYYIZXIII*	2
11	*IZXZXZXXIZX* *ZXIZXYIIIYY* *ZXYIIXYYYXI* *YIYZXXZIYXX* *IYZYXXIYZYX* *IYXXYXIYXZZ* *ZIXZYIZXZIX* *YYXZXYXYZXX* *ZIYZXYXZIYI* *YYZIXZZIZIZ*	3	*YZXIXIZYXZX* *ZXIIYIXYZXI* *YYXZXYIIXXX* *ZXIIZXZYYZX* *IXZYIIXYIIZ* *ZXIZIXYXIXY* *IZIXZZZYXXX* *ZYXIZYXXXXX* *XZIZIIIXYIY*	3	*YZXZXXZXXYY* *YZIYXIIYIXX* *YYIZYXYZXXX* *ZYIZIIYYZYX* *YYZIIYXIZXY* *YXZXYXZZYII* *YIIIIZYZIYX* *IZXXYXZIYIX*	2
12	*YYIIXIYYIXZZ* *YXIYZIXZIZIY* *XXXXIZXIZXXZ* *ZXZYIIIIZYXI* *ZYYYZIZIYIZX* *IZYYXXXYIYIY* *ZYYYYIYYXIYX* *ZYIZYXIXIYXY* *IZIXIXYXXYYZ* *IXXZZYIIXXXY* *ZZIZYXIZXYZX*	4	*ZXZXZYXXZZYI* *ZIZYZIXIXZIY* *IYYYXZXXZYYY* *IXIIIYYZIZYZ* *IIYIIXYYZIIZ* *IIIXYYIZXYIX* *ZIXYIIZXIZZY* *YIYZZXYZIXXX* *YYXXYXXZYYYI* *YZYXZZYYYYII*	3	*IIYYYIZXIXII* *IZXZZXXIIXZZ* *ZZZYZXYIZZII* *YZIXIZXIIIXX* *XYZZZIXZYYYI* *XIYZYXIXXYIY* *YIZZYXXYZIII* *ZYXZXIIZIIZI* *ZXZZYYXZXIYY*	3

Table 3.4: Generators and distances for the best codes found for the AD channel using hill climbing.

	$k=1$		$k=2$		$k=3$	
n	Generators	d	Generators	d	Generators	d
5	*YYXIX* *IXYYX* *IZXXZ* *XZIZX*	3	*YXZZX* *IIYYX* *IYIYX*	1	*XZZYX* *IXIXI*	1
6	*XZZZIY* *IZYXYY* *YIZXZI* *YYXIXZ* *IZZYXZ*	3	*XYIYZY* *XYYIXZ* *IIXXZZ* *XZYZZI*	1	*YIXIYZ* *IYXYYI* *IZXYZY*	1
7	*XZZIYIX* *XIYZYXZ* *IYYXXXZ* *ZIXXIYI* *XZZZIZI* *XIXYZIX*	3	*ZYYXYXX* *XYZZIZI* *XZIZZYZ* *YZZXIXY* *XYIYXXZ*	2	*IZXIZYX* *YZYXIIZ* *XYIIZIZ* *IYYXXIY*	1
8	*ZXXIYIYX* *YXXZXIIX* *YIYZIIXZ* *IXXXXYIZ* *ZYXZIZZX* *YIZXYZZY* *IYXZYZZI*	3	*YIYYYXIZ* *YYXXZYXZ* *ZYXIXYIZ* *ZZZXXIIZ* *YZIIXXXY* *IXIIXIZY*	3	*XZXIZYZI* *ZXXYXXZY* *YZZIXXIZ* *XIXXXZIX* *IZXYIZYZ*	3
9	*ZIXYXIYYI* *ZYYZXXYIX* *IYIXIYIZI* *YYZZIXIZI* *IXYYYXYYY* *IIIZIXZIY* *ZXZYIYYZX* *ZZXZZXIYY*	3	*IIYYXYIYI* *XZYXXZXII* *IXIZXXYIZ* *YZXXIYZYI* *YXZYXYZXY* *YZYYZIYZX* *YYYIYZYIX*	3	*IXIYZYZZI* *YZIIYZZXZ* *ZYIZIYIIX* *YZXXIXYII* *YXIIIIXYX* *YZZIXYIZX*	3
10	*YIIZYYXYYY* *XYZXIZZXYZ* *IXYXZYYYYX* *XYIIZZXXXZ* *XYIYIYXXIX* *XZZYZYYIXX* *ZIXIYZYZII* *IIXXXIZIXI* *XXYYXXXXZI*	3	*YZXYZYYIYY* *YYXZIIXZXI* *XIIYIIXXYI* *IYZZIIXXXX* *YXZIIXZIYY* *XXYYZIZZYI* *YYZYZZYIYZ* *YXYXXIZYYX*	3	*ZZZZYZZXZX* *XXZXZZXXYX* *XXYZXZXZXY* *YIZYIXZXYX* *ZXYYZXIYYY* *XIXYXZIZYX* *IIIZYXIYXZ*	3
11	*ZXYZXYIZZXX* *XZXZYIXXYZY* *YZXZIZYXXYY* *YYYIXZIIYZY* *ZZZZXZIXZYZ* *IZZXIIIXZXI* *ZZIXYZXZYXY* *IZZIYXIZXXY* *XYZZXZZXZIZ* *YIYYIIYIZXI*	3	*XIYZIZIIXZZ* *YZZYXIYXYYY* *IYIIZIYZYZY* *ZZIYZZYZIIZ* *IZYYZXXYYZX* *YYZZIYYXZYX* *ZZZZYZYXYIX* *ZXYIXXZYXYY* *IIZXIYYYYYY*	3	*ZIYIZXXXIZZ* *XZIYXZIIXZX* *IYZZIXIYXYX* *ZYIYIYZZXXI* *YXXZYIIXZYI* *YYZYYZXYYZX* *IYYXXXIZIXX* *YXXXYXIXIZY*	3
12	*IXIIZZIZIXYI* *XXIZXXIIIXII* *ZXXYIZYXIZYY* *ZYYYYIXZYXIX* *YZYYXYXZZXZI* *IZIZZYIXIZIZ* *ZZIXYYIZYYYZ* *YYIXZXZYIIZZ* *ZZXXXXZZXYYY* *ZZIIXXZIZIXI* *XYXZIYZXIXZZ*	3	*IIYXXIXZYIYI* *ZIXXZIZYIIIX* *YZZZZIYYIIZI* *IZXIYYZZXYII* *ZZYYIYZXYZXZ* *IYXIXYYYIYXX* *XYXYZXYIXYZZ* *XIXIYIZZZZYY* *XXYXXXXYXIIX* *YXZYXIIIZIXI*	3	*XYZYZIZIIXXX* *XXYIYIIZIIXZ* *YZZZYYYXIZII* *ZIIIXYXYXIYI* *XIZZZYYIXIZY* *IXXIIZXYXYXI* *ZZXIIYZZYIZI* *ZYZZIZXIIZZZ* *ZZXIYYZXIYXZ*	3

Table 3.5: Generators and distances for the best weight-four codes found for the biased XZ channel using hill climbing.

	$k=1$		$k=2$		$k=3$	
n	Generators	d	Generators	d	Generators	d
5	*ZIXZX* *IZXXZ* *ZYYIZ* *XZZIX*	3	*XYYIX* *ZXIXX* *ZYYIZ*	1	*XIYZZ* *IXYZZ*	1
6	*IIXZZX* *YIIYXX* *ZZIIXY* *IXZIXX* *YIZIZY*	2	*XIZXYI* *IYYXXI* *IZIZXX* *IZYIZY*	1	*IXYYIX* *IIXZYX* *XXYIXI*	1
7	*IZXIIZY* *ZZIXXII* *YIXZIZI* *IXIZXIZ* *IZZIIXY* *ZYYIZII*	3	*IIYIYZX* *IZIXIYY* *ZYIIYIX* *XIIYXXI* *YIZXIIY*	2	*YIZIIYY* *XIXYIIY* *IZYIXIZ* *IYIZIZX*	1
8	*IZZIYIYI* *IIIYYIYY* *XXYIIIIY* *YIIXIXIX* *IIYZZYII* *YYIXXIII* *IIIIXZZY*	3	*XIIZXIIX* *YIIYIXXI* *IIYXZYII* *XIIIZIYY* *IYXIIZYI* *ZXIIIYIZ*	2	*IYIYIYYI* *IIIYXIYY* *XIIIYXXI* *YYXIIIYI* *IXYXIIIZ*	2
9	*IXYYIXIII* *ZIIYZIIIY* *IIIIYIZYZ* *ZIYIIIYZI* *IIZYIZIYI* *IZIIIYIZY* *YIYIIIIXX* *IYIXYXIII*	3	*YIXIIIXIX* *YYIIZXIII* *XIIIXIYYI* *IIXXIIXXI* *XIZIIYIYI* *IIYIZIIZY* *IXIYXIIYI*	2	*XXXIXIIII* *YYIIIZIXI* *ZIYIIYXII* *YIIYZIYII* *IIYZYIIIY* *IIIYIYIYZ*	2
10	*XIIXYIIXII* *IIYYZIXIII* *IIIXIIYIZY* *IIZIZIYYII* *ZIIIIXIYIY* *IZYIIZIXII* *IYIXIXYIII* *IZIIYIZIYI* *YZIYIIIIIZ*	3	*IIIIZYIXIY* *XIIIIIYIXY* *IIIYZIZIYI* *IYXIIIXIZI* *XIYXIIYIII* *YIXIIXIIIX* *IXIIYXYIII* *YIIIXIXYII*	2	*XIIIIYIYIX* *IXYXIIYIII* *XYXIYIIIII* *XYIYIIIIYI* *IIIYIXZIIY* *IIYIXYYIII* *ZIYIIIIXXI*	2
11	*IIIIYZIZYII* *IZYYYIIIIII* *YIIIYIIXXII* *IIIYIIZIIZY* *IIYIIIZIIYZ* *XIIIIIYIYXI* *YYIZIIIIIYI* *ZIIYIYIIZII* *IIXIXIIYIIY* *IYXIIIXIIYI*	3	*IIIIIYZYIZI* *IZIYIIIYIIX* *IIIIYXIIZYI* *YXIIIIXXIII* *XIXIIIYIIYI* *XYIIIIIIZIY* *YXYXIIIIIII* *IXIXXIIIYII* *IIIIIIXIYXX*	2	*IIYYIYIIZII* *YIIIXIYIIXI* *IYXZIIIIIIY* *XIIZYIIIYII* *YZIYIIIYIII* *XIIIIIIXIYY* *IIXIYXXIIII* *YIYIXIIIIIX*	2
12	*IZIIIIYZYIII* *IIIIZYYIIZII* *IYXIIIIIZIIY* *XIIIIYIIYIIX* *IYIZXIZIIIII* *YXIXIIIIXIII* *IXZIIIXIIYII* *IIZIIIIYXIIY* *XIIYZIIIIIXI* *YIIIIZIIIYZI* *ZIIIIIIIYIYZ*	3	*IIYXIIIYIIIZ* *IIXIZYIIIIIY* *ZIIYIIIIIYIY* *IIIIIIYIYYZI* *YIIZIIIZYIII* *IIIIYZXIIIYI* *YIIIIYZIIXII* *IIIIIIIXXIYY* *IYIIXXIIIYII* *IXYIXYIIIIII*	2	*IZZIIIYIIIIY* *XIIXIZYIIIII* *YIIYYIIYIIII* *IIIYIYIIYIZI* *IYIIIIIYYIIX* *IIYIIXXIIIYI* *IIIIXIIXZIYI* *YIZIIYIIIYII* *IXYIIIIIIZIY*	2

Table 3.6: Generators and distances for the best weight-four codes found for the AD channel using hill climbing.

	$k=1$		$k=2$		$k=3$	
n	Generators	d	Generators	d	Generators	d
5	*IXZXZ* *YZIYZ* *IZYZY* *ZXIZX*	3	*ZXYIY* *ZXYYI* *XIYXX*	1	*XYIXZ* *XYYIZ*	1
6	*YIXXYI* *YXIIYZ* *ZIXIXZ* *IXIYZY* *XIZZXI*	1	*YIYXXI* *IZYIXY* *YIZIYY* *ZXIYIZ*	2	*IYYXYI* *XXYZII* *ZYIIYX*	1
7	*ZYIIXXI* *ZIIYIXZ* *XIIXYIZ* *YZYZIII* *IYXZIIY* *IIZXYYI*	3	*ZYIIXIX* *IXYIYYI* *ZIXZXII* *YXIYIYI* *IIIXZXY*	2	*XYYIXII* *XIIXXXI* *ZIXYIIY* *ZZIIXYI*	1
8	*ZIYIIYXI* *IZXYIZII* *YIIIZIZX* *IIIZIXYY* *YYIIIYZI* *IIXIXIZZ* *IIYYZYII*	3	*XIYIIYYI* *IYIIIZXX* *IIXIXXIY* *IYYXYIII* *XXIZIIIY* *YIIYXIXI*	3	*IXYXIIIZ* *YIIXYYII* *IIIZIXYX* *IYIIZZIY* *IZIIYIZY*	1
9	*IIYIIXYYI* *IIIXXXIIX* *IYXYIYIII* *XYXIIIIXI* *IIXYIIXIY* *YIYIIXIIX* *IXYIXIYII* *XIIIYYXII*	3	*XIXIIXIYI* *IIIYIZXXI* *IIYIXIIXY* *YIIXYIIZI* *YYIIIYYII* *IXIXIIZIY* *IXXIIIXIX*	2	*YIYIIIYIZ* *XXIIIXIIX* *IXXIZIXII* *XIZIXIIXI* *YIIXIYIZI* *IYIIXIXIY*	1
10	*YIIYYIIIIY* *IIIXIXIXIX* *IYYIIIIYIY* *XXXIXIIIII* *IYIIYIYIZI* *IIIXXXIIXI* *ZIYIXIXIII* *IIIIIZXIYY* *XIXXIIIXII*	3	*IIYXIYYIII* *IIIIYXZYII* *IXZIXIXIII* *IYYIIIIIXY* *YZIIYIIIIX* *IIIYIXIIZX* *IIIZIIXXXI* *XIIIXYIIXI*	3	*IIIIZXXIXI* *YYXIIIIIXI* *XIYZIIIIIX* *YIIXIXIYII* *IIIIYYIXIX* *IIZIXIIIYY* *IIIYIZYIIY*	1
11	*YIYIIYIIIIY* *IIXIYZIYIII* *IXIYIIZYIII* *IXYIXIIIIXI* *IIZIIXIIYYI* *IIIXIYIXIXI* *XIIXIIXIIIX* *ZXIIIIIIYIZ* *YIIIIIZIXXI* *IYZIIIIXIIX*	3	*IIIIIIXIYXY* *IIIZIIYXIIX* *IYIYIXIZIII* *IYZIIIIIXYI* *YIXIYIIIYII* *IXIIYIIXYII* *ZIYIIXIIIXI* *IIIIXXYIXII* *YIIXIZXIIII*	3	*YIIIIIXIXIY* *XIIIIYIXZII* *IIXYXIIIIIY* *IYIIXIIXIZI* *IIXIIZIYIXI* *IXIXZIXIIII* *IIZXIXIIXII* *XZIIIIYIIXI*	1
12	*IXIIYIIXXIII* *IIXIIIIXIYIX* *IIIIXIIIYXIY* *IIIIXYIYIIIY* *IIIYYXIIIYII* *ZXIIIIZIIIXI* *IZIIIIYIZYII* *IIYXIYIIIIIY* *YIIXIYIIIIYI* *YIIIIIXIXYII* *XIXYIIYIIIII*	3	*IIYZZIIIIIYI* *IYIIIZIIIYYI* *IIXIYIIYIIIX* *IIIIXYIIIIZY* *YIIIIXIIIXIX* *YIYIIIYXIIII* *IXIYIIIIXIZI* *ZXIIIYXIIIII* *IYIIIIZYZIII* *IIIIIIIXXYIY*	3	*IIIYXZIIXIII* *XIIYIIIIIIXX* *IIYZIXIIIIIY* *IIIIZIIIYYXI* *ZIIIIIYXIIIY* *YIIIIIIYIZYI* *IYIIIIIXXXII* *IXYIIIZYIIII* *IZZIIYIIYIII*	3

Table 3.7: Generators and distances for the best-performing inequivalent cyclic codes with weight-four generators on the biased XZ and AD channels. If a code requires two generators, they are grouped in brackets; otherwise, a single generator is given as in Table 3.2. The generators of codes performing well on both channel types are given in bold, while the generators for codes previously appearing in Table 3.2 are marked with an asterisk.

	Biased XZ		AD	
$[[n,k]]$	Generators	d	Generators	d
$[[5,1]]$	***YZIZY****	3	***YZIZY****	3
$[[6,1]]$	*YIZZIY* *	2	*YZIIZY* *XZIIZX*	2 2
$[[6,2]]$	*YZIZYI* *	2	*XIZIXY* * *YIZIYX* *	2 2
$[[6,3]]$	***XIYXIY**** ***YZIYZI****	2 2	***XIYXIY*** ***YZIYZI*** *XZIXZI*	2 2 2
$[[7,1]]$	***XZIZXII****	3	***XZIZXII**** *YZIZYII* *	3 3
$[[8,1]]$	***ZZYIIIIY****	3	***ZZYIIIIY*** *ZZXIIIIX*	3 3
$[[8,2]]$	***YIIXIIYX**** *YIIZIIYZ* *	2 2	***YIIXIIYX**** *XIIYIIXY* *	2 2
$[[8,3]]$	***YIIIIYYY*** ***XIIIIXXX***	1 1	***YIIIIYYY*** ***XIIIIXXX***	1 1
$[[9,1]]$	***ZIZYIIIIY****	3	***ZIZYIIIIY**** *ZIZXIIIIX* *	3 3
$[[9,3]]$	***IIIYYIYYI*** ***IIIXXIXXI***	1 1	***IIIYYIYYI*** ***IIIXXIXXI***	1 1
$[[10,1]]$	*IIYZIIIIZY*	2	*XIIZIIZIIX* *YIIZIIZIIY*	3 3
$[[10,2]]$	***IYXIIIIIXY*** *IZYIIIIIYZ*	3 3	***IYXIIIIIXY****	3
$[[11,1]]$	***IYIIZIIZIIY****	3	***IYIIZIIZIIY**** *IXIIZIIZIIX* *	3 3
$[[12,1]]$	***IIIIYIIZZIIY***	3	***IIIIYIIZZIIY*** *IIIIXIIZZIIX*	3 3
$[[12,2]]$	*YIIIIZIIIIYZ*	2	*XZIIIIIIIZXI* *YZIIIIIIIZYI*	3 3
$[[12,3]]$	*(**XIIXIIXIIXII**,* ***IYIIIIYIIYYI**)*	2	*(**XIIXIIXIIXII**,* ***IYIIIIYIIYYI**)* *(YIIYIIYIIYII,* *IXIIIIXIIXXI)*	2 2

Table 3.8: Generators and distances for the best CSSY codes found for the biased XZ channel using hill climbing.

	$k=1$		$k=2$		$k=3$	
n	Generators	d	Generators	d	Generators	d
5	*XXXXX* *IIIYY* *IYIYI* *IIYIY*	1	*XXXIX* *YYIYI* *YIYYI*	1	*XXXXI* *XXIXX*	1
6	*XXXXXX* *IIYYYY* *IYYIYY* *YIYYIY* *IYYYYI*	2	*IXXXXX* *YIYYII* *YIYIYI* *YYIYII*	1	*IXIXXX* *YYIIYI* *YIYIYY*	1
7	*IIIXIXX* *XXXXXIX* *YYIYYYI* *YIIYYIY* *IYIIYII* *YYYIIYY*	2	*XXXXXXX* *YYIIYYI* *YYIYIYI* *IYYYYYY* *YYIYIIY*	2	*XXXXXXX* *YIIYYYI* *YYIIYIY* *YYYYIII*	2
8	*XIXXIIXX* *IXIXXXXI* *IYYIIYIY* *IYYIIIYI* *YIYIYIYY* *IYYYIYYY* *YYYYIIIY*	2	*XXXXIIXX* *XIXXXXXI* *IIIIIYYY* *IYYYIYYI* *YYIIIYII* *IYYIYYYY*	2	*XXIXXXXX* *XXXXIXXX* *YIIIIYYY* *YYIYIIYI* *IIYYYIYY*	2
9	*IIXXIIIXX* *XXIXXIIXX* *IXXXXIXXI* *IIIXXXIIX* *IIYYYYIYY* *IYYIIYIIY* *IYIYYYYIY* *YIIIYYYII*	3	*IXXXXIXIX* *XXIIXXXXI* *XIXIXXXII* *YIIYYIIII* *YYIIIYIYY* *IIYYIYIYI* *YIIYYYYIY*	2	*XXIIXXXIX* *IIXXXXIXX* *YYIYYIYII* *IYYYIYYIY* *IIIYYIIYY* *IIIYIYYII*	2
10	*IXXIXXXIXI* *IIXXIIXIXX* *IXIXIIXXXI* *XIIIXIIXXI* *YYIIIIIIYY* *YIYYIIIIYY* *IYYIIIYYYY* *YIYIIYYIYY* *IIIYYIIIYI*	3	*IXIXIIXXXX* *IXXXXXXIIX* *XIIXXIXXII* *YIIYIYYYIY* *YIYYIYIIIY* *IIYIYIYIIY* *YYYIYIIIIY* *IYIIIYIIYI*	2	*XIIXXXIXIX* *IIXIXXXIXI* *XXIIXIIXXI* *YYYIYIIYIY* *YIIIIIYIYY* *YIYYYYYIII* *IIIIYYIYIY*	2
11	*XIXXIIXIXIX* *IXIIXIXXIXX* *IIIXXXXIXXI* *XXIIXIIXXIX* *IIIIIIYYYII* *YYIIYIYYYYY* *IYYIIIIIIIY* *IYYYYIYYIYY* *YYYIYIYIYYI* *IIYIYYIIIIY*	3	*IIXXIXXIXXX* *XIIXIIXXXII* *IIXIXIXIXXI* *IXXXIIXXIIX* *YIYYIIIYYIY* *YYYIIYYIIIY* *YYIYYIIIIYI* *YIYIIIYIIII* *IYYIYIYIYIY*	3	*IXIIXXXIIIX* *XIIXXIXXIXX* *XIIIIXXXXIX* *XXXIIXXIXXI* *YIYYIYYYIII* *IIYYIYIYYYY* *YYYIYIIYIYI* *YIIIYYIYYYI*	3
12	*XXXIIXIIXXXX* *IXIXXXXXIXXI* *IIXXIIXXXXII* *XXIXXXXIIIII* *XIIXIXXXXXXI* *IIXXIIIIIXIX* *IIYIYIYYYIYY* *YYIIYIYIYIYI* *YIIIYYYYIIII* *IIYYIIYYIIYI* *IIYIYYIYYYII*	3	*XXIXIIXIXXIX* *XIXIIIXXXIXI* *XXXIIXIXXIXX* *IIIXXXIXXIIX* *IIYIIIYYIIYY* *IYYIYIYIYYYI* *IYYYIYIYYYYI* *IYIIIIIYIIYY* *IIYIIYIIIYYY* *YYYYYIIYIIYY*	2	*XIIXIXXIXXII* *IIXXXIXXXIIX* *XXXIIIIXXIII* *XXXIXIXIIXXI* *IYYYYYYYYIII* *YIYIIIYYYYII* *YIIYIYYYIIIY* *YIYYYIIIIIYY* *YIYIIIYIIIYI*	3

Table 3.9: Generators and distances for the best CSSY codes found for the AD channel using hill climbing.

	$k=1$		$k=2$		$k=3$	
n	Generators	d	Generators	d	Generators	d
5	*IIXIX* *IIIXX* *IYYYY* *YIYYY*	1	*XIXXI* *XXXII* *IYYYY*	1	*IIYYI* *YYYIY*	1
6	*XXXXXI* *XXXXIX* *YIIYII* *IYIYII* *YIYYYY*	2	*XXXXXX* *IXIXXI* *IYYYIY* *YIIYYY*	2	*IXXIXI* *YIYYYI* *YYIIYY*	1
7	*XXXXIII* *XXIIXXI* *XIXIIXX* *IYIYIYY* *YYIIYYI* *IIYYYYI*	3	*IXIIIIX* *XXXIXXI* *XXXXIIX* *IYYYYYY* *YYYIYIY*	2	*XXXXXXX* *IIYYYIY* *YYIIYIY* *IYYIIYY*	2
8	*IXIXXIII* *XXXXXXXI* *IXIXIXII* *XXIIXXXX* *YYIIYYII* *YYYIYYYY* *IYIYIIIY*	2	*IXXIIIIX* *XXXIXIIX* *XXIXIXXX* *YYIIYYIY* *YYIYYIIY* *IIYYIYYY*	2	*XIXIXXII* *IXXIIXXX* *XIIXIXXX* *YYYYYYIY* *IIIIIIYY*	2
9	*IIXXXIXXX* *XXXIIIXII* *XIXIXXXIX* *IXIXIIXIX* *YIIIIYYIY* *IYYYIYIII* *YIYIYYIII* *YIIYYYYYI*	3	*IXXXXXIIX* *XXIXIIIXI* *IIXXIIXXX* *XIXXIXXXI* *YYYIYIIIY* *YYIYYYIYI* *IYIYIIYII*	2	*IXXXIXIXX* *IIIXXIXXX* *XIXIXXIXI* *YYYIIYYYI* *YYIYIIYYY* *IIIYYYIII*	2
10	*XIIXIXIXXX* *IXXXXIIIIX* *XIXXXIXIXX* *IXXIIIXXXX* *YYYIIYIIII* *IYIYYYYYIY* *IIYIIIYIYY* *IIYIYIYYYI* *YIYIYYYIII*	3	*XXXIXXIIXI* *XIXXIIXXXI* *IXXIIXIXXX* *IXXXIIIXII* *YYIIYYIYIY* *YYYIIIYIYY* *YYIYYIYIYI* *IYIYYYIIYY*	3	*XXIXIXXIIX* *XXXIXXIIXI* *IIIIIXIXXX* *XIIXXXIIII* *IYIIYYYYYY* *YIIYYYYIYI* *IIYYIYYIIY*	2
11	*XIXIIXIXIII* *XIIXIXIIIXI* *IXXXIIIIXIX* *XXXIXXXXXIIX* *IXIIXXIIXXI* *IYIIYIYIIIY* *IIIYYIIIIYY* *IIYYYYYIIII* *YYIYIYIIYYY* *YYIIIIIYIYY*	3	*XIXXXIIXXXI* *XXXIIXXIXIX* *IIXIIXIXXXX* *XIXXIIXXIIX* *YIIIIYYIYII* *IIYIYYIYIYI* *IYIYIIYYYYY* *IIIYYYYYYIY* *YIYYYYIYYII*	3	*XXXIIXXXIXI* *IXXXXIXXIII* *IXIIXXIXXXX* *IXXIIXIIIIX* *YIYYIYIIIYI* *YIYYIIYYIIY* *YYYIYYIYIYY* *YIIYYIYYYYI*	3
12	*XIIXIIXXXIII* *XXIXIIIIXIIX* *XXIIXXXXIXIX* *IIXIXIXXXIXX* *IIXIXIIIIXIX* *IIIXXIXIXXXX* *YYYIIIYIIYII* *IYIYYYYYYIYY* *YYYYIIIYYYYI* *YIYYIYYIYIIY* *IYYYYYYIIIYI*	3	*XXXXIIIIIIXI* *XXXIXIIIIXIX* *XIIXXXXIIXII* *XXIXXXIIXXXX* *XXXXXIIXXIIX* *IIYIYIIYIYYY* *YYIIIIIYIYIY* *IIIYYYIYYYYI* *IYIYYIYYYYIY* *YYYYIIYIIYII*	3	*XXIIIIIXXXXI* *XIXIXXIIXIXI* *XXIIXXXXXXIIX* *XXIXIXXIIIXI* *YYIYIIIYIIYY* *IYIYYIIYYYII* *IIYYIIIIIYYI* *IYIIYYIIIYIY* *YYYIYIYIIYYI*	3

Table 3.10: Generators and distances for the best linear codes found for the biased XZ channel using hill climbing.

	$k=1$		$k=2$		$k=3$	
n	Generators	d	Generators	d	Generators	d
5	*XXYIY* *ZZXIX* *XYIYX* *ZXIXZ*	3	-	-	-	-
6	-	-	*YIXIYX* *XIZIXZ* *XYZZZY* *ZXYYYX*	2	-	-
7	*ZIZXZZY* *YIYZYYX* *IYYYZII* *IXXXYII* *ZYYIIXI* *YXXIIZI*	3	-	-	*IXZYXXZ* *IZYXZZY* *YYXZXIY* *XXZYZIX*	2
8	-	-	*IYXZYYXI* *IXZYXXZI* *XIYXYYIY* *ZIXZXXIX* *IXYYYXIY* *IZXXXZIX*	3	-	-
9	*ZZIIIXIIY* *YYIIIZIIX* *IXZZZXZZX* *IZYYYZYYZ* *IYIZZIZXX* *IXIYYIYZZ* *IIYYIIXXI* *IIXXIIZZI*	3	-	-	*ZXXXYXXIZ* *YZZZXZZIY* *ZZZXIYXZY* *YYYZIXZYX* *ZYZIIZZXI* *YXYIIYYZI*	3
10	-	-	*XZYXXXYZZX* *ZYXZZZXYYZ* *ZYXYXZZXXX* *YXZXZYYZZZ* *YXIZYYXIYZ* *XZIYXXZIXY* *ZIYYYXYZYI* *YIXXXZXYXI*	3	-	-
11	*YZXIIYYZIZX* *XYZIIXXYIYZ* *YYXZYXIZYYY* *XXZYXZIYXXX* *YIXYYXZXZZY* *XIZXXZYZYYX* *IZZYXIYIYXX* *IYYXZIXIXZZ* *ZXZXIZIIZXZ* *YZYZIYIIYZY*	3	-	-	*XIZZZIYYIXY* *ZIYYYIXXIZX* *XYXIYIYXYIZ* *ZXZIXIXZXIY* *XXZXXIIZYYI* *ZZYZZIIYXXI* *ZIYIYYYYIYY* *YIXIXXXXIXX*	3
12	-	-	*IIIIYYIIIIZZ* *IIIIXXIIIIYY* *YYZYZYIIYIIY* *XXYXYXIIXIIX* *IZZZZZZIXXYY* *IYYYYYYIZZXX* *ZXYIYYXYIZII* *YZXIXXZXIYII* *ZYYXZZIIZXZZ* *YXXZYYIIYZYY*	4	-	-

Table 3.11: Generators and distances for the best linear codes found for the AD channel using hill climbing.

	$k=1$		$k=2$		$k=3$	
n	Generators	d	Generators	d	Generators	d
5	*XIXZZ* *ZIZYY* *IXZXZ* *IZYZY*	3	-	-	-	-
6	-	-	*IXIXYY* *IZIZXX* *YXXIIY* *XZZIIX*	2	-	-
7	*IXYYXYZ* *IZXXZXY* *XXIYYYY* *ZZIXXXX* *XZXYIZY* *ZYZXIYX*	3	-	-	*XZZZIXY* *ZYYYIZX* *IXYZYXY* *IZXYXZX*	2
8	-	-	*IYIYZYXX* *IXIXYXZZ* *ZIIXYXYX* *YIIZXZXZ* *ZZZXXIIZ* *YYYZZIIY*	3	-	-
9	*YXXIXZXZY* *XZZIZYZYX* *ZXYYXYIZZ* *YZXXZXIYY* *IZYYIYXIX* *IYXXIXZIZ* *IIIIYIZYZ* *IIIIXIYXY*	3	-	-	*ZZIXIZIXY* *YYIZIYIZX* *YZIXXXXYX* *XYIZZZZXZ* *IXYXYXXZZ* *IZXZXZZYY*	3
10	-	-	*XYYIIYZZYX* *ZXXIIXYYXZ* *ZXYIYIXXZZ* *YZXIXIZZYY* *XXIIYZZXYY* *ZZIIXYYZXX* *YXXYYXXIIZ* *XZZXXZZIIY*	3	-	-
11	*YIYZYZZYXXY* *XIXYXYYXZZX* *YXIIIYZZIXI* *XZIIIXYYIZI* *XYXZIZYIXIZ* *ZXZYIYXIZIY* *XIZXIXXXZIX* *ZIYZIZZZYIZ* *YXXZXYYIXYX* *XZZYZXXIZXZ*	3	-	-	*XZXYYIIXIZZ* *ZYZXXIIZIYY* *YIIXIZYXXII* *XIIZIYXZZII* *ZIIIXZZYZZY* *YIIIZYYXYYX* *IYXZZZIZXIZ* *IXZYYYIYZIY*	3
12	-	-	*IIIIYXYYXZXX* *IIIIXZXXZYZZ* *IIZIXZZZIZZY* *IIYIZYYYIYYX* *IZIZIXYIXZII* *IYIYIZXIZYII* *ZIYXIXXIIIYI* *YIXZIZZIIIXI* *XIYIYIXXIYZX* *ZIXIXIZZIXYZ*	4	-	-

Chapter 4

Heuristic construction of codeword stabilized codes[1]

Abstract

The family of codeword stabilized codes encompasses the stabilizer codes as well as many of the best known nonadditive codes. However, constructing optimal n-qubit codeword stabilized codes is made difficult by two main factors. The first of these is the exponential growth with n of the number of graphs on which a code can be based. The second is the NP-hardness of the maximum clique search required to construct a code from a given graph. We address the second of these issues through the use of a heuristic clique finding algorithm. This approach has allowed us to find $((9, 97 \leq K \leq 100, 2))$ and $((11, 387 \leq K \leq 416, 2))$ codes, which are larger than any previously known codes. To address the exponential growth of the search space, we demonstrate that graphs that give large codes typically yield clique graphs with a large number of nodes. The number of such nodes can be determined relatively efficiently, and we demonstrate that n-node graphs yielding large clique graphs can be found using a genetic algorithm. This algorithm uses a crossover operation based on spectral bisection that we demonstrate to be superior to more standard crossover operations. Using this genetic algorithm approach, we have found $((13, 18, 4))$ and $((13, 20, 4))$ codes that are larger than any previously known code. We also consider codes for the amplitude damping channel. We demonstrate that for $n \leq 9$, optimal codeword stabilized codes correcting a single amplitude damping error can be found by considering standard form codes that detect one of only three of the 3^n possible equivalent error sets. By combining this error set selection with the genetic algorithm approach, we have found $((11, 68))$ and $((11, 80))$ codes capable of correcting a single amplitude damping error and $((11, 4))$, $((12, 4))$, $((13, 8))$, and $((14, 16))$ codes capable of correcting two amplitude damping errors.

[1]This chapter has been published as Ref. [31]: A. Rigby, J. C. Olivier, and P. D. Jarvis, "Heuristic construction of codeword stabilized codes," *Physical Review A*, vol. 100, no. 6, p. 062303, Dec. 2019, doi.org/10.1103/PhysRevA.100.062303. Only minor typographical and formatting changes have been made.

4.1 Introduction

Quantum codes can be used to protect quantum information against the effects of a noisy channel. An n-qubit code is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$ of dimension K. If $\mathcal{Q}$ can detect any arbitrary error on fewer than d qubits but not some error on d qubits, then $\mathcal{Q}$ is said to have distance d and is called an $((n, K))$ or $((n, K, d))$ code. Equivalently, a code has distance d if it can detect the set $\mathcal{E}$ of all Pauli errors of weight less than d but cannot detect some weight-d Pauli error [85]. A well-understood family of codes is the stabilizer (additive) codes, which are codes defined using an abelian subgroup of the n-qubit Pauli group [15]. However, codes outside of the stabilizer framework, called nonadditive codes, can potentially encode a larger subspace while still detecting the same error set [17, 18, 19, 20, 21, 22]. Codeword stabilized (CWS) codes encompass both the stabilizer codes and many of the best known nonadditive codes [32, 33]. In general, an $((n, K))$ CWS code is defined using an n-qubit stabilizer state, which is a stabilizer code of dimension one, and a set of K n-qubit Pauli operators called word operators [32]. A standard form CWS code $\mathcal{Q}$ is one where the stabilizer state is defined by a simple undirected n-node graph G (that is, it is a graph state) and the word operators are defined by a binary classical code $\mathcal{C} \subseteq \mathrm{GF}(2)^n$ with $|\mathcal{C}| = K$ [32]. For $\mathcal{Q}$ to detect an error set $\mathcal{E}$, $\mathcal{C}$ must detect the classical error set $Cl_G(\mathcal{E}) \subseteq \mathrm{GF}(2)^n$ induced by the graph. An appropriate classical code of maximum size can be found by constructing a clique graph and performing a maximum clique search [33].

The error set that must be detected by an $((n, K, d))$ code $\mathcal{Q}$ is invariant under any permutation of the Pauli matrices X, Y, and Z on any subset of qubits. As a result of this symmetry, we call $((n, K, d))$ codes symmetric codes. This symmetry also means that if $\mathcal{Q}'$ is local Clifford (LC) equivalent to $\mathcal{Q}$ (that is, if $\mathcal{Q}' = U\mathcal{Q}$ for some LC operator U), then $\mathcal{Q}'$ is also an $((n, K, d))$ code. It follows from the LC equivalence of every stabilizer state to a graph state [86, 87, 88] that every CWS code is LC equivalent to one in standard form [32]. It is therefore sufficient to consider only standard form codes when attempting to construct an optimal $((n, K, d))$ CWS code. In fact, it is sufficient to consider only codes based on graph states that are not LC equivalent up to a permutation of qubit labels [33]. This corresponds to considering only graphs that are not isomorphic up to a series of local complementations [86]. For $n \leq 12$, this set of inequivalent graphs, denoted $\mathcal{L}_n$, has been enumerated [34, 35, 36] and, in theory, can be exhaustively searched to construct an optimal code. Such a search of $\mathcal{L}_{10}$ has previously yielded the well-known $((10, 24, 3))$ code [19]. For distance-two codes, searching $\mathcal{L}_n$ quickly becomes prohibitive with increasing n due to the rapidly growing clique graphs and the NP-hardness of finding a maximum clique [37]. To address this, we employ the heuristic Phased Local Search (PLS) clique finding algorithm [89]. Using this approach, we have found $((9, 97 \leq K \leq 100, 2))$ and $((11, 387 \leq K \leq 416, 2))$ codes that are larger than the best known nonadditive codes presented in Refs. [20] and [21], respectively.

The apparent exponential growth of $|\mathcal{L}_n|$ with increasing n means that even if $\mathcal{L}_n$ were enumerated for $n \geq 13$, an exhaustive search would be prohibitive. As such, other search strategies are required for constructing codes. To aid this search, we demonstrate a relationship between

the code size and the order (number of nodes) of the clique graph yielded by a given graph. In particular, we show that the clique graph orders exhibit clustering and that the graphs yielding the best codes tend to belong to the highest clique graph order cluster. This reduces the search to finding graphs that yield large clique graphs, and we show that such graphs can be generated by using a genetic algorithm to search the set of all distinct n-node graphs. This genetic algorithm uses a crossover operation based on spectral bisection, which we show to be significantly more effective than standard single-point, two-point, and uniform crossover operations. Using this genetic algorithm approach, we have found $((13, 18, 4))$ and $((13, 20, 4))$ codes. These codes are larger than an optimal $((13, 16, 4))$ stabilizer code, and to the best of our knowledge, they are the first $d \geq 4$ codes to achieve this (we note that there is a family of $d = 8$ nonadditive codes that are larger than the best known, but not necessarily optimal, stabilizer codes [22]).

For asymmetric codes, the error set $\mathcal{E}$ that they detect is no longer invariant under Pauli matrix permutation. This means that if $\mathcal{Q}$ detects $\mathcal{E}$, then there is no guarantee that an LC-equivalent code $\mathcal{Q}' = U\mathcal{Q}$ also detects $\mathcal{E}$. However, if $\mathcal{Q}$ detects the LC-equivalent error set $U^\dagger \mathcal{E} U$, then $\mathcal{Q}'$ will detect $\mathcal{E}$. As a result, when attempting to construct an optimal $((n, K))$ code CWS code detecting $\mathcal{E}$, it is sufficient to consider standard form codes based on elements of $\mathcal{L}_n$ that detect one of the up to 6^n possible LC-equivalent error sets [90] (the 6^n value stems from there being six possible permutations of the Pauli matrices on each of the n qubits). Such an asymmetric error set arises when constructing codes that correct amplitude damping errors. In this case, a partial symmetry reduces the number of LC-equivalent error sets to 3^n; however, this is still large enough to make an exhaustive search prohibitive for $n \geq 10$. Again, we therefore require different search strategies for constructing codes. We demonstrate that for $n \leq 9$, optimal CWS codes correcting a single amplitude damping error can be found by considering only codes based on nonisomorphic graphs that detect one of three LC-equivalent error sets. By combining this error set selection with the genetic algorithm approach, we have found $((11, 68))$ and $((11, 80))$ codes capable of correcting a single amplitude damping error. These are larger than the best known stabilizer codes detecting the same error set [15]. We have also found $((11, 4))$, $((12, 4))$, $((13, 8))$, and $((14, 16))$ stabilizer codes capable of correcting two amplitude damping errors.

The paper is organized as follows. Section 4.2 gives an introduction to undirected graphs, genetic algorithms, classical codes, and quantum codes. Section 4.3 details our search strategies for symmetric codes and presents the best codes we have found. This is then extended to asymmetric codes for the amplitude damping channel in Sec. 4.4. The paper is concluded in Sec. 4.5.

4.2 Background

4.2.1 Undirected graphs

A simple undirected graph $G = (N, E)$ of order n consists of a set of nodes $N = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E \subseteq \{\{v_i, v_j\} : v_i, v_j \in N, v_i \neq v_j\}$. An edge $e = \{v_i, v_j\} \in E$ is an unordered

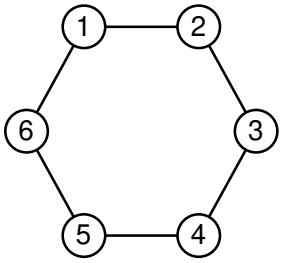

Figure 4.1: A drawing of a cycle graph where the circles correspond to nodes and the lines to edges.

pair that connects the nodes $v_i, v_j \in N$, which are called the endpoints of e. A graph is typically drawn with the nodes depicted as circles that are joined by lines representing the edges. An example of such a drawing is given in Fig. 4.1. G can be represented by the symmetric $n \times n$ adjacency matrix Γ, where

$$\Gamma_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

The neighborhood $\mathcal{N}(v_i) = \{v_j : \{v_i, v_j\} \in E\}$ of some node $v_i \in N$ is the set of nodes to which it is connected. The degree $\deg(v_i) = |\mathcal{N}(v_i)|$ of v_i is the number of nodes to which it is connected. The $n \times n$ degree matrix D has elements

$$D_{ij} = \begin{cases} \deg(v_i) & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

A subgraph $G_S(N_S, E_S)$ of $G = (N, E)$ is a graph with nodes $N_S \subseteq N$ and edges $E_S \subseteq E$. The subgraph induced by a subset of nodes $N_I \subseteq N$ is the graph $G_I = G[N_I] = (N_I, E_I)$, where $E_I = \{\{v_i, v_j\} \in E : v_i, v_j \in N_I\}$ contains all the edges in E that have both endpoints in N_I. A walk is a sequence whose elements alternate between connected nodes and the edges that connect them. For example, $1, \{1,2\}, 2, \{2,3\}, 3, \{3,4\}, 4, \{3,4\}, 3$ is a walk in the graph shown in Fig. 4.1. The length of a walk is the number of edges it contains. A path is a walk containing no repeated nodes or edges, with the exception that the first and last node can be the same, in which case the path is called a cycle. A graph, such as the one shown in Fig. 4.1, where all nodes belong to a single cycle is called a cycle graph. A graph is connected if there is a path between any two of its nodes. A connected component of G is a maximal connected subgraph $G_S(N_S, E_S)$ [maximal in that there is no other connected subgraph $G_T(N_T, E_T)$ where $N_S \subset N_T$].

Two graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ are isomorphic if they differ only up to a relabeling of nodes. Formally, they are isomorphic if there exists an isomorphism from G_1 to G_2, which is a bijection $f : N_1 \to N_2$ such that $\{v_i, v_j\} \in E_1$ if and only if $\{f(v_i), f(v_j)\} \in E_2$. An isomorphism $f : N \to N$ from a graph $G = (N, E)$ to itself is called an automorphism. The set of all all automorphisms of G forms a group $\mathrm{Aut}(G)$ under composition. There are a number of packages, such as NAUTY [91, 92], available for determining the automorphism group of a given graph. We denote the set of all distinct n-node graphs with nodes $N = \{1, 2, \ldots, n\}$ as $\mathcal{D}_n$, the

Table 4.1: The number of distinct, nonisomorphic, and non-LC-isomorphic graphs with $n \leq 12$ nodes.

n	$\lvert\mathcal{D}_n\rvert$	$\lvert\mathcal{G}_n\rvert$	$\lvert\mathcal{L}_n\rvert$
1	2^0	1	1
2	2^1	2	2
3	2^3	4	3
4	2^6	11	6
5	2^{10}	34	11
6	2^{15}	156	26
7	2^{21}	1 044	59
8	2^{28}	12 346	182
9	2^{36}	274 668	675
10	2^{45}	12 005 168	3 990
11	2^{55}	1 018 997 864	45 144
12	2^{66}	165 091 172 592	1 323 363

size of which grows exponentially with $|\mathcal{D}_n| = 2^{\binom{n}{2}}$. $\mathcal{D}_n$ can be partitioned up to isomorphism to give the set $\mathcal{G}_n$; $|\mathcal{G}_n|$ also grows exponentially with n [93] as shown in Table 4.1 for $n \leq 12$. The size of some $g \in \mathcal{G}_n$ with representative $G \in \mathcal{D}_n$ is $n!/|\mathrm{Aut}(G)|$ [93].

The complement $\bar{G} = (N, \bar{E})$ of a graph $G = (N, E)$ has an edge $\{v_i, v_j\} \in \bar{E}$ if and only if $\{v_i, v_j\} \notin E$. A local complementation (LC) at node v_i replaces the induced subgraph $G[\mathcal{N}(v_i)]$ with its complement (while we use LC for both local Clifford and local complementation, its meaning should be clear from the context in which it is used). If two graphs $G_1, G_2 \in \mathcal{D}_n$ differ by a series of local complementations, then we say they are LC equivalent. If a series of local complementations applied to G_1 yields a graph G_2' that is isomorphic to G_2, then we say that G_1 and G_2 are LC isomorphic. Partitioning $\mathcal{D}_n$ up to LC isomorphism gives the set $\mathcal{L}_n$, which has been enumerated for $n \leq 12$ [34, 35, 36] and also seems to grow exponentially with n as shown in Table 4.1. Any two graphs that are isomorphic are necessarily LC isomorphic, and therefore, any element $l \in \mathcal{L}_n$ is the union $l = \cup_i g_i$ of elements $g_i \in \mathcal{G}_n$. These g_i can be determined from any representative of l using Algorithm 5.1 of Ref. [34]. If a subset $A \subseteq \mathcal{D}_n$ contains graphs that are representatives of m different elements of $\mathcal{G}_n$ ($\mathcal{L}_n$), then we say m of the graphs in A are nonisomorphic (non-LC isomorphic).

A graph $G = (N, E)$ is complete if every node is connected to every other node; that is, if $E = \{\{v_i, v_j\} : v_i, v_j \in N, v_i \neq v_j\}$. If an induced subgraph $G[\tilde{N}]$ for some $\tilde{N} \subseteq N$ is complete, then $\tilde{N}$ is called a clique. A clique of maximum size in G is called a maximum clique. Finding a maximum clique in a graph is is NP-hard [37]; however, there are a number of heuristic algorithms that can find large, if not maximum, cliques. One such algorithm is the Phased Local Search (PLS) [89], which performs well compared to other heuristic algorithms in terms of both speed and clique finding ability [94]. The PLS algorithm constructs a clique by initially selecting a node at random. It then iteratively selects nodes to add to the current clique (potentially replacing an existing node in the clique) until a maximum number of selections is reached. To ensure good performance on graphs with varying structures, PLS cycles through multiple different selection methods. The search is repeated for a prescribed number of attempts, after which the largest clique found is returned.

A bipartition of $G = (N, E)$ divides the nodes into two disjoint subsets N_1 and N_2. A bipartition is called a bisection if $|N_1| = |N_2|$ for even $|N|$ or if $||N_1| - |N_2|| = 1$ for odd $|N|$. An optimal bisection is one that minimizes the number of edges connecting nodes in N_1 to those in N_2. Finding such an optimal bisection is NP-hard [95]; however, approximate heuristic approaches are available. One such approach is spectral bisection [96, 97, 98, 99], which is based on the graph's Laplacian matrix $L = D - \Gamma$. L is positive semidefinite and, as such, has real, non-negative eigenvalues. The eigenvector $\boldsymbol{u} = (u_1, \ldots, u_n)$ corresponding to the second smallest eigenvalue is called the Fiedler vector [100]. The Fiedler vector can be used to bisect N, with the indices of the $\lfloor n/2 \rfloor$ smallest components of $\boldsymbol{u}$ dictating the nodes in N_1 and N_2 simply being $N_2 = N \backslash N_1$.

4.2.2 Genetic algorithms

Suppose we wish to determine which element in a set A is optimal in some sense. This can be expressed as finding the $a \in A$ that maximizes a fitness function $f : A \to \mathbb{R}$. The brute-force approach to this problem is to determine the fitness of every element $a \in A$. This is called an exhaustive search and quickly becomes impractical if the search space A is large and/or evaluating the fitness of elements is computationally intensive. In such cases, heuristic search algorithms can be used to find good, but potentially not optimal, elements of A. The simplest such approach is a random search, where fitness is calculated only for the elements in a randomly selected subset $B \subset A$. Another heuristic search strategy is the genetic algorithm, which is inspired by natural evolution [101, 82]. There are many genetic algorithm variants; a simple implementation is as follows. Initially, the child population, which is an N-element subset of A, is randomly generated. This is followed by a calculation of each child's fitness (a child being an element of the child population). The genetic algorithm then iterates for some predetermined number of maximum generations. In each generation, the previous generation's child population becomes the current generation's parent population (whose elements are called parents). A new child population is then formed by selecting two parents at a time and producing two children from them. The parents are selected according to their fitness, with high fitness parents having a higher chance of selection. With probability p_c, the two children will be produced via crossover, which combines attributes of the two parents; otherwise, they will simply be duplicates of their parents. Each child is then subjected to mutation (random alteration) with probability p_m before being added to the child population. Once the child population again contains N children, their fitnesses are calculated and a new generation begins.

Tournament selection is a simple and commonly used method of selecting parents based on their fitness. First, a subset of the parent population is chosen at random, and then the fittest parent within this subset is selected. The size of the subset chosen is called the tournament size; it controls the selection pressure of the genetic algorithm, which is a measure of how dependent selection is on having high fitness. If the tournament size is large, then there is high selection pressure, meaning that the highest-fitness parents tend to be selected. This exploitative approach gives faster convergence; however, the search is more likely to become stuck at a suboptimal

local maximum [102]. Conversely, a small tournament size will lead to greater exploration of the search space at the cost of potentially slow convergence. A common modification to the genetic algorithm is the inclusion of elitist selection, which involves adding some number of the fittest parents to the child population at the start of each generation. This preserves the best elements; however, the increased selection pressure can again increase the probability of convergence to a suboptimal local maximum.

The crossover and mutation operations used depend on how elements of A are represented. A standard representation involves encoding elements as bit strings of fixed length b. A common and simple mutation operation in this case involves flipping any given bit in a child bit string with some probability (this probability is often taken to be $1/b$ [82]). Standard crossover methods include single-point, two-point, and uniform crossover. In single-point crossover, an index $1 \leq i \leq b$ is chosen, and the values beyond this point are exchanged between the two parent bit strings to form two child bit strings. In two-point crossover, two such indices are selected and all values between them are exchanged. In uniform crossover, each individual bit is exchanged between the two parents with some probability p_e.

In some cases, representations other than bit strings are more natural. For example, it may be possible to represent elements as graphs. Crossover becomes more complicated with such a representation. A potential method is presented in Ref. [103] and is as follows. First, the parent graphs P_1 and P_2 are each split into two subgraphs, called fragments, to produce disconnected parents P_{1D} and P_{2D}. To split a parent graph, first an edge $\{v_i, v_j\}$ is chosen at random. In an iterative process, the shortest path between v_i and v_j is determined, and a randomly selected edge in this path is removed (in the first iteration, this will simply be the edge $\{v_i, v_j\}$). This continues until no path exists between v_i and v_j. The connected component containing v_i is the fragment F_1, and the subgraph induced by the remaining nodes is the fragment F_2. In the next step, disconnected children C_{1D} and C_{2D} are formed by exchanging a fragment, say F_1, between each of the parent graphs. The two fragments in each disconnected child are then combined to produce children C_1 and C_2. This combination process involves iteratively selecting a node from each fragment and joining them with an edge. The probability of a node being selected is proportional to the difference in its current degree to its degree in its initial parent graph. This process of adding edges is repeated until all of the nodes in one of the fragments, say F_1, have the same degree as they did in their initial parent graph. If a node v_l in F_2 has degree lower than its initial degree by some amount δ_l, then in a process repeated δ_l times, it will be connected to a randomly selected node in F_1 with 50% probability. As outlined in Ref. [104], the splitting process presented here has some undesirable attributes. Firstly, it tends to produce two fragments with a vastly different number of nodes. Secondly, it often removes a large number of edges from within the larger fragment; these are edges that did not have to be removed to split the parent graph.

4.2.3 Classical codes

A classical channel is a map $\Phi : \mathcal{A}_x \to \mathcal{A}_y$, where $\mathcal{A}_x$ is the set of possible inputs and $\mathcal{A}_y$ is the set of possible outputs. We are concerned with channels where the input and outputs are binary [that is, channels for which $\mathcal{A}_x = \mathcal{A}_y = \mathrm{GF}(2)$]. In this case, the action of the channel can be expressed as

$$\Phi(x) = x + e = y, \tag{4.3}$$

where $x \in \mathrm{GF}(2)$ is the channel input, $y \in \mathrm{GF}(2)$ is the channel output, and $e \in \mathrm{GF}(2)$ is an error (or noise) symbol. A code can be used to protect against the noise introduced by the channel. A length-n binary code is a subset $\mathcal{C} \subseteq \mathrm{GF}(2)^n$ whose elements are called codewords. Codewords are transmitted as n sequential uses of Φ or, equivalently, as a single use of the combined channel Φ^n, which is comprised of n copies of Φ. The action of Φ^n on some input $\boldsymbol{x} \in \mathcal{C}$ is

$$\Phi^n(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{e} = \boldsymbol{y}, \tag{4.4}$$

where $\boldsymbol{y} \in \mathrm{GF}(2)^n$ is the channel output and $\boldsymbol{e} \in \mathrm{GF}(2)^n$ is an error "vector." The weight of an error is the number of nonzero components from which it comprised.

We say that a code $\mathcal{C}$ can detect a set of errors $\mathcal{E} \subseteq \mathrm{GF}(2)^n$ if

$$\boldsymbol{x}_i + \boldsymbol{e} \neq \boldsymbol{x}_j \tag{4.5}$$

for all $\boldsymbol{e} \in \mathcal{E}$ and $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}$, where $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. That is, the errors in $\mathcal{E}$ can be detected if they do not map one codeword to another. Furthermore, we say that $\mathcal{C}$ can correct $\mathcal{E}$ if

$$\boldsymbol{x}_i + \boldsymbol{e}_k \neq \boldsymbol{x}_j + \boldsymbol{e}_l \tag{4.6}$$

for all $\boldsymbol{e}_k, \boldsymbol{e}_l \in \mathcal{E}$ and $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}$, where $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. This condition simply ensures that two codewords cannot be mapped to the same $\boldsymbol{y} \in \mathrm{GF}(2)^n$, in which case the transmitted codeword cannot be inferred with certainty. $\mathcal{C}$ is said to have distance d if it can detect any error of weight less than d but is unable to detect some weight-d error. Note that $\mathcal{C}$ can correct $\mathcal{E}$ if and only if it can detect $\mathcal{E} + \mathcal{E} = \{\boldsymbol{e}_k + \boldsymbol{e}_l : \boldsymbol{e}_k, \boldsymbol{e}_l \in \mathcal{E}\}$, meaning that a distance-d code can correct any error of weight $t = \lfloor (d-1)/2 \rfloor$ or less. A length-n code $\mathcal{C}$ of size $|\mathcal{C}| = K$ and distance d is called an (n, K) or (n, K, d) code. If $\mathcal{C}$ forms a vector space, then it is called linear and has $K = 2^k$. A linear code encodes the state of k bits and is called an $[n, k]$ or $[n, k, d]$ code.

Finding a code $\mathcal{C}$ of maximum size that detects an error set $\mathcal{E}$ can be expressed as a clique finding problem. This is achieved by constructing a graph $G_{\mathcal{E}} = (N_{\mathcal{E}}, E_{\mathcal{E}})$ whose nodes are potential codewords; that is, $N_{\mathcal{E}} = \mathrm{GF}(2)^n$. Two nodes $\boldsymbol{x}_i, \boldsymbol{x}_j \in N_{\mathcal{E}}$ are connected by an edge $\{\boldsymbol{x}_i, \boldsymbol{x}_j\} \in E_{\mathcal{E}}$ if $\boldsymbol{x}_i + \boldsymbol{x}_j \notin \mathcal{E}$ (that is, if there is not an error mapping one to the other). Any clique $\mathcal{C}$ in $G_{\mathcal{E}}$ is a code detecting $\mathcal{E}$, and a code of maximum possible size is a maximum clique in $G_{\mathcal{E}}$. Note that if a code $\mathcal{C}$ detects $\mathcal{E}$, then so does $\mathcal{C}' = \boldsymbol{x} + \mathcal{C}$ for any $\boldsymbol{x} \in \mathcal{C}$. As $\mathbf{0} \in \mathcal{C}'$ and $|\mathcal{C}| = |\mathcal{C}'|$, this means there is always an optimal code (that is, a maximum-size code detecting $\mathcal{E}$) that contains the all-zero codeword. The clique search can be restricted to such codes by taking $N_{\mathcal{E}} = \mathbf{0} \cup [\mathrm{GF}(2)^n \backslash \mathcal{E}]$.

4.2.4 Quantum codes

The action of a quantum channel Φ on a quantum state described by the density operator ρ is

$$\Phi(\rho) = \sum_k A_k \rho A_k^\dagger, \tag{4.7}$$

where the A_k, called Kraus operators, satisfy $\sum_k A_k^\dagger A_k = I$ (the identity operator) [14]. The channel can be interpreted as mapping $\rho \mapsto A_k \rho A_k^\dagger$ (up to normalization) with probability $\mathrm{tr}(A_k \rho A_k^\dagger)$ [6]. If $\rho = |\phi\rangle\langle\phi|$ (that is, if the input state is pure), then this becomes the mapping $|\phi\rangle \mapsto A_k|\phi\rangle$ (up to normalization) with corresponding probability $\langle\phi|A_k^\dagger A_k|\phi\rangle$. In this paper, we are interested in qubit systems; that is, systems where states $|\phi\rangle$ belong to a two-dimensional Hilbert space $\mathcal{H} \cong \mathbb{C}^2$. Similar to the classical case, the noise introduced by a quantum channel can be protected against by employing a code. A quantum (qubit) code of length n is a subspace $\mathcal{Q} \subseteq (\mathbb{C}^2)^{\otimes n}$. Codewords $|\phi\rangle \in \mathcal{Q}$ are transmitted across the combined n-qubit channel $\Phi^{\otimes n}$.

Suppose a code $\mathcal{Q}$ has an orthonormal basis $\mathcal{B} = \{|\phi_1\rangle, \ldots, |\phi_K\rangle\}$, and take $\mathcal{E} = \{E_1, \ldots, E_r\}$ to be the basis for some complex vector space of linear n-qubit operators (called error operators). We say that $\mathcal{Q}$ can detect any error in the span of $\mathcal{E}$ if

$$\langle\phi_i|E|\phi_j\rangle = C_E \delta_{ij} \tag{4.8}$$

for all $E \in \mathcal{E}$ and $|\phi_i\rangle, |\phi_j\rangle \in \mathcal{B}$, where C_E is a scalar that depends only on E [15]. Furthermore, we say that $\mathcal{Q}$ can correct any error in the span of $\mathcal{E}$ if

$$\langle\phi_i|E_k^\dagger E_l|\phi_j\rangle = C_{kl}\delta_{ij} \tag{4.9}$$

for all $E_k, E_l \in \mathcal{E}$ and $|\phi_i\rangle, |\phi_j\rangle \in \mathcal{B}$, where C is an $r \times r$ Hermitian matrix [85]. The weight of an error E is the number of qubits on which it acts. $\mathcal{Q}$ has distance d if it can detect any error of weight less than d but not some weight-d error. Similar to the classical case, a code can correct $\mathcal{E}$ if and only if it can detect $\mathcal{E}^\dagger\mathcal{E} = \{E_k^\dagger E_l : E_k, E_l \in \mathcal{E}\}$, meaning that a distance-d quantum code can also correct any error of weight $t = \lfloor (d-1)/2 \rfloor$ or less. A length-n code of dimension K and distance d is called an $((n, K))$ or $((n, K, d))$ code (the double brackets differentiate from the classical case). A code $\mathcal{Q}$ correcting $\mathcal{E}$ is called nondegenerate if the spaces E_kQ and E_lQ are linearly independent (that is, their intersection is trivial) for any $E_k, E_l \in \mathcal{E}$, where $E_k \neq E_l$. If all such spaces are orthogonal, then $\mathcal{Q}$ is called pure.

The Pauli matrices in the computational $\{|0\rangle, |1\rangle\}$ basis are

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{4.10}$$

X can be viewed as a bit-flip operator as $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Z can be viewed as a phase flip as $Z|0\rangle = |0\rangle$ and $Z|1\rangle = -|1\rangle$. $Y = iXZ$ can be viewed as a combined bit and phase flip. The Pauli matrices are Hermitian, unitary, and anticommute with each other. Furthermore, they form the group

$$\mathcal{P}_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} = \langle X, Y, Z\rangle. \tag{4.11}$$

called the Pauli group. The n-qubit Pauli group $\mathcal{P}_n$ consists of all n-fold tensor product combinations of elements of $\mathcal{P}_1$. For example, $\mathcal{P}_8$ contains the element $I \otimes I \otimes X \otimes I \otimes Y \otimes Z \otimes I \otimes I$, which we can write more compactly as $X_3Y_5Z_6$. The weight $w(g)$ of some $g \in \mathcal{P}_n$ is the number of elements in the tensor product that are not equal to the identity up to phase. The commutation relations of the Pauli matrices mean that elements of $\mathcal{P}_n$ must either commute or anticommute, with two elements anticommuting if their nonidentity components differ in an odd number of places. The Pauli matrices along with the identity form a basis for the complex vector space of all 2×2 matrices. It therefore follows that

$$\mathcal{E}^r = \{E = \sigma_1 \otimes \cdots \otimes \sigma_n : \sigma_i \in \{I, X, Y, Z\} \text{ and } w(E) \leq r\}, \tag{4.12}$$

is a basis for all n-qubit errors of weight less than or equal to r. An equivalent definition is $\mathcal{E}^r = \{E_1 \dots E_r : E_i \in \mathcal{E}^1\}$; that is, $\mathcal{E}_r$ is the set of all r-fold products of elements of $\mathcal{E}_1$, which can be written as $\mathcal{E}^1 = \{I, X_i, Y_i, Z_i\}$, where $1 \leq i \leq n$. It is sometimes convenient to express some $E \in \mathcal{P}_n$ up to phase as $E \propto X^{u_1}Z^{v_1} \otimes \cdots \otimes X^{u_n}Z^{v_n} = X^{\boldsymbol{u}}Z^{\boldsymbol{v}}$, where $\boldsymbol{u} = (u_1, \dots, u_n)$, $\boldsymbol{v} = (v_1, \dots, v_n) \in \mathrm{GF}(2)^n$.

Two n-qubit codes $\mathcal{Q}$ and $\mathcal{Q}'$ are local unitary (LU) equivalent if $\mathcal{Q}' = U\mathcal{Q}$ for some $U \in U(2)^{\otimes n}$. These codes will have the same dimension as if $\mathcal{B} = \{|\phi_1\rangle, \dots, |\phi_K\rangle\}$ is an orthonormal basis for $\mathcal{Q}$, then $\mathcal{B}' = U\mathcal{B} = \{U|\phi_1\rangle, \dots, U|\phi_K\rangle\}$ is an orthonormal basis for $\mathcal{Q}'$. It follows from Eq. (4.8) that $\mathcal{Q}'$ detects the error set $\mathcal{E}$ if and only if $\mathcal{Q}$ detects the LU-equivalent error set $\mathcal{E}' = U^\dagger \mathcal{E} U$. Furthermore, $\mathcal{E}$ is a basis for all errors of weight less than d if and only if $\mathcal{E}'$ is also such a basis. Therefore, $\mathcal{Q}$ and $\mathcal{Q}'$ have the same distance; that is, they are both $((n, K, d))$ codes. If two codes differ by a LU operator and/or permutation of qubit labels, which also has no effect on the size or distance of the code, then they are called equivalent codes. The normalizer of $\mathcal{P}_1$ in $U(2)$ is the single-qubit Clifford group $\mathcal{C}_1 = \{U \in U(2) : U^\dagger \mathcal{P}_1 U = \mathcal{P}_1\}$. The n-qubit local Clifford group $\mathcal{C}_1^n$ is comprised of all possible n-fold tensor products of elements from $\mathcal{C}_1$. Two codes are local Clifford (LC) equivalent if they are LU equivalent for some $U \in \mathcal{C}_1^n \subset U(2)^{\otimes n}$.

Stabilizer codes (also called additive codes) are defined by an abelian subgroup $\mathcal{S} < \mathcal{P}_n$, called the stabilizer, that does not contain $-I$ [15]. The code $\mathcal{Q}$ is the space of states that are fixed by every element $s_i \in \mathcal{S}$; that is,

$$\mathcal{Q} = \{|\phi\rangle \in (\mathbb{C}^2)^{\otimes n} : s_i|\phi\rangle = |\phi\rangle \; \forall \, s_i \in \mathcal{S}\}. \tag{4.13}$$

The requirement that $-I \notin \mathcal{S}$ both means that no $s \in \mathcal{S}$ can have a phase factor of $\pm i$, and that if $s \in \mathcal{S}$, then $-s \notin \mathcal{S}$. If $\mathcal{S}$ is generated by $M = \{M_1, \dots, M_m\} \subset \mathcal{P}_n$, then it is sufficient (and obviously necessary) for $\mathcal{Q}$ to be stabilized by every M_i. Assuming that the set of generators is minimal, it can be shown that $\dim(\mathcal{Q}) = 2^{n-m} = 2^k$ [6]; that is, $\mathcal{Q}$ encodes the state of a k-qubit system. An n-qubit stabilizer code with dimension $K = 2^k$ and distance d is called an $[[n, k]]$ or $[[n, k, d]]$ code.

An n-qubit stabilizer state $|\mathcal{S}\rangle$ is an $[[n, 0, d]]$ code defined by a stabilizer $\mathcal{S}$ with n generators. The distance of a stabilizer state is defined to be equal to the weight of the lowest nonzero weight element in $\mathcal{S}$. A graph state $|G\rangle$ is a stabilizer state defined by a graph $G \in \mathcal{D}_n$. Each node i

corresponds to a qubit and is also associated with a stabilizer generator

$$M_i = X_i Z_{\mathcal{N}(i)} = X_i \prod_{j \in \mathcal{N}(i)} Z_j. \tag{4.14}$$

Each graph state $|G\rangle$ defines a basis $\mathcal{B} = \{Z^{\boldsymbol{w}}|G\rangle : \boldsymbol{w} \in \mathrm{GF}(2)^n\}$ for $(\mathbb{C}^2)^{\otimes n}$ [105]. An error $E = X_i$ maps the graph state $|G\rangle$ to

$$X_i|G\rangle = X_i(X_i Z_{\mathcal{N}(i)})|G\rangle = Z_{\mathcal{N}(i)}|G\rangle = Z^{\boldsymbol{r}_i}|G\rangle, \tag{4.15}$$

where $\boldsymbol{r}_i$ is the ith row of the adjacency matrix for G. That is, an X error applied at node i is equivalent to Z errors being applied at its neighbors; this is called the $X - Z$ rule [106]. It can be shown that every stabilizer state is LC equivalent to a graph state [86, 87, 88]. Two graph states $|G_1\rangle$ and $|G_2\rangle$ are the same up to a relabeling of qubits if and only if their corresponding graphs G_1 and G_2 are isomorphic. Furthermore, $|G_1\rangle$ and $|G_2\rangle$ are LC equivalent if and only if G_1 and G_2 are LC equivalent [86]. Therefore, $|G_1\rangle$ and $|G_2\rangle$ are equivalent (as quantum codes) if G_1 and G_2 are LC isomorphic (the converse does not necessarily hold as two states can be LU equivalent without being LC equivalent [107]).

4.2.5 CWS codes

The family of codeword stabilized (CWS) codes contains all stabilizer codes as well as many of the best known nonadditive codes [32, 33]. An $((n, K))$ CWS code $\mathcal{Q}$ is defined using an n-qubit stabilizer state $|\mathcal{S}\rangle$ and a set of K word operators $\mathcal{W} = \{W_1, \ldots, W_K\} \subset \mathcal{P}_n$. In particular, $\mathcal{Q}$ is the span of the basis codewords $|W_i\rangle = W_i|\mathcal{S}\rangle$. Note that for the $|W_i\rangle$ to actually form a basis, no two word operators can differ only by a stabilizer element; that is, it cannot be the case that $W_i W_j \in \bar{\mathcal{S}} = \cup_{\alpha \in \{\pm 1, \pm i\}} \alpha \mathcal{S}$ for any $W_i \neq W_j$. For a CWS code, the criterion for detecting an error set $\mathcal{E}$ becomes

$$\langle W_i|E|W_j\rangle = \langle \mathcal{S}|W_i^\dagger E W_j|\mathcal{S}\rangle = C_E \delta_{ij} \tag{4.16}$$

for all $E \in \mathcal{E}$ and $W_i, W_j \in \mathcal{W}$. If $\mathcal{E}$ contains only Pauli errors $E \in \mathcal{P}_n$, then

$$\langle \mathcal{S}|W_i^\dagger E W_j|\mathcal{S}\rangle = \begin{cases} 0 & \text{if } W_i^\dagger E W_j \notin \bar{\mathcal{S}}, \\ \alpha & \text{if } W_i^\dagger E W_j \in \alpha \mathcal{S}, \end{cases} \tag{4.17}$$

where $\alpha \in \{\pm 1, \pm i\}$. Therefore, the $i \neq j$ case of Eq. (4.16) holds for some $E \in \mathcal{E}$ if and only if

$$W_i^\dagger E W_j \notin \bar{\mathcal{S}} \tag{4.18}$$

for all $W_i, W_j \in \mathcal{W}$, where $W_i \neq W_j$. Furthermore, the $i = j$ case holds for some $E \in \mathcal{E}$ if and only if either

$$W_i^\dagger E W_i \notin \bar{\mathcal{S}} \tag{4.19}$$

or

$$W_i^\dagger E W_i \in \alpha \mathcal{S} \tag{4.20}$$

for all $W_i \in \mathcal{W}$ and some particular $\alpha \in \{\pm 1, \pm i\}$.

It follows from the LC equivalence of every stabilizer state to a graph state that every CWS code is LC equivalent to one based on a graph state $|G\rangle$ with word operators of the form $W_i = Z^{\boldsymbol{x}_i}$ [32]. Such a code is called a standard form CWS code, and its basis codewords are simply elements of the graph basis defined by G. The set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\} \subseteq \mathrm{GF}(2)^n$ forms a classical binary code $\mathcal{C}$, and without loss of generality, we can take $\boldsymbol{x}_1 = \mathbf{0}$ [32]. It can be shown that if $\mathcal{C}$ is linear, then the CWS code is additive [32], whereas if $\mathcal{C}$ is not linear, then the code may be additive or nonadditive [33] (although if $K \neq 2^k$, then the CWS code must obviously be nonadditive). The effect of an error $E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}}$ on one of the basis codewords $|W_i\rangle = Z^{\boldsymbol{x}_i}|G\rangle$ follows from the $X - Z$ rule, with

$$\begin{aligned} E|W_i\rangle &\propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}} Z^{\boldsymbol{x}_i}|G\rangle \\ &\propto Z^{\boldsymbol{v}} Z^{\boldsymbol{x}_i} X^{\boldsymbol{u}}|G\rangle \\ &= Z^{\boldsymbol{v}} Z^{\boldsymbol{x}_i} Z^{\boldsymbol{u}\Gamma}|G\rangle \\ &= Z^{\boldsymbol{v}} Z^{\boldsymbol{u}\Gamma} Z^{\boldsymbol{x}_i}|G\rangle \\ &= Z^{Cl_G(E)}|W_i\rangle, \end{aligned} \tag{4.21}$$

where Γ is the adjacency matrix for G and

$$Cl_G(E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}}) = \boldsymbol{v} + \boldsymbol{u}\Gamma. \tag{4.22}$$

Therefore, the effect of $E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}}$ is equivalent to that of $E' = Z^{Cl_G(E)}$, where $Cl_G(E) \in \mathrm{GF}(2)^n$ is a classical error induced by the graph. It follows from this equivalence that $\langle W_i|E|W_j\rangle \propto \langle W_i|Z^{Cl_G(E)}|W_j\rangle$, which means that Eq. (4.18) is satisfied when

$$Z^{\boldsymbol{x}_i} Z^{Cl_G(E)} Z^{\boldsymbol{x}_j} \notin \bar{\mathcal{S}}. \tag{4.23}$$

For a graph state, the only stabilizer element with no X component is the identity $I = Z^{\mathbf{0}}$. Equation (4.23) therefore reduces to $\boldsymbol{x}_i + Cl_G(E) \neq \boldsymbol{x}_j$, which is simply the classical error detection criterion of Eq. (4.5). This means that an error E can be detected only if $\mathcal{C}$ detects the classical error $Cl_G(E)$. Following the same reasoning, Eq. (4.19) becomes $\boldsymbol{x}_i + Cl_G(E) \neq \boldsymbol{x}_i$, which reduces to $Cl_G(E) \neq \mathbf{0}$. Equation (4.20) becomes

$$Z^{\boldsymbol{x}_i} E Z^{\boldsymbol{x}_i} \in \alpha\mathcal{S}, \tag{4.24}$$

which reduces to $E \in \alpha\mathcal{S}$ for $\boldsymbol{x}_1 = \mathbf{0}$. If there is some $W_i = Z^{\boldsymbol{x}_i}$ that anticommutes with E, then Eq. (4.24) becomes $E \in -\alpha\mathcal{S}$. This would mean that both $\alpha^{-1}E \in \mathcal{S}$ and $-\alpha^{-1}E \in \mathcal{S}$, from which it follows that $-I \in \mathcal{S}$. This cannot be the case as $\mathcal{S}$ is a stabilizer. Therefore, to satisfy Eq. (4.20), it must be the case that $[Z^{\boldsymbol{x}_i}, E] = 0$ for all $\boldsymbol{x}_i \in \mathcal{C}$. For $E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}}$, this condition is equivalent to requiring $\boldsymbol{x}_i \cdot \boldsymbol{u} = 0$ for all $\boldsymbol{x}_i \in \mathcal{C}$, were $\boldsymbol{a} \cdot \boldsymbol{b} = \sum_i a_i b_i$ is the standard Euclidean inner product. In summary, a standard form code detects $E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}} \in \mathcal{E}$ if

$$\boldsymbol{x}_i + Cl_G(E) \neq \boldsymbol{x}_j \tag{4.25}$$

for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{C}$, where $\boldsymbol{x}_i \neq \boldsymbol{x}_j$, and either

$$Cl_G(E) \neq \mathbf{0} \tag{4.26}$$

or

$$\boldsymbol{x}_i \cdot \boldsymbol{u} = 0 \tag{4.27}$$

for all $\boldsymbol{x}_i \in \mathcal{C}$.

Designing a CWS code $\mathcal{Q}$ for a given graph G and error set $\mathcal{E}$ consists of finding a classical code $\mathcal{C}$ that satisfies Eqs. (4.25)-(4.27) for every $E \in \mathcal{E}$. It is convenient to express this as a clique finding problem as outlined in Ref. [33]. First, the set of classical errors

$$Cl_G(\mathcal{E}) = \{Cl_G(E) : E \in \mathcal{E}\} \tag{4.28}$$

induced by the graph is determined. Also required is the set

$$\begin{aligned} D_G(\mathcal{E}) = \{\boldsymbol{x} \in \mathrm{GF}(2)^n : Cl_G(E) = \mathbf{0} \,\text{and} \\ \boldsymbol{x} \cdot \boldsymbol{u} \neq 0 \,\text{for some}\, E \propto X^{\boldsymbol{u}} Z^{\boldsymbol{v}} \in \mathcal{E}\}. \end{aligned} \tag{4.29}$$

These are elements of $\mathrm{GF}(2)^n$ that cannot be included in the code as they violate Eqs. (4.26) and (4.27). An algorithm for efficiently determining $Cl_G(\mathcal{E})$ and $D_G(\mathcal{E})$ is given in Ref. [33]. A classical code $\mathcal{C}$ satisfying Eqs. (4.25)-(4.27) is a clique in the graph $G_{\mathcal{E}} = (N_{\mathcal{E}}, E_{\mathcal{E}})$ with

$$N_{\mathcal{E}} = \mathbf{0} \cup \{\mathrm{GF}(2)^n \backslash [Cl_G(\mathcal{E}) \cup D_G(\mathcal{E})]\} \tag{4.30}$$

and $E_{\mathcal{E}}$ defined by the classical error set $Cl_G(\mathcal{E})$ as outlined in Sec. 4.2.3. That is, two nodes $\boldsymbol{x}_i, \boldsymbol{x}_j \in N_{\mathcal{E}}$ are connected by an edge $\{\boldsymbol{x}_i, \boldsymbol{x}_j\} \in E_{\mathcal{E}}$ if $\boldsymbol{x}_i + \boldsymbol{x}_j \notin Cl_G(\mathcal{E})$. If $D_G(\mathcal{E}) = \emptyset$, then for all $E \not\propto I \in \mathcal{E}$, it must be the case that $Cl_G(E) \neq \mathbf{0}$, and hence $C_E = 0$ in Eq. (4.16). Therefore, for $E = E_k^\dagger E_l \in \mathcal{E}$ where $E_k E_l \in \mathcal{P}_n$ and $E_k \not\propto E_l$, it follows that $\langle W_i | E_k^\dagger E_l | W_j \rangle = 0$. That is, $\mathcal{Q}$ is pure if $D_G(\mathcal{E}) = \emptyset$ [33, 108].

4.2.6 Code bounds

A simple, but relatively loose, upper bound on the dimension K of an n-qubit code of distance d is given by the quantum singleton bound [85]

$$K \leq 2^{n-2(d-1)}. \tag{4.31}$$

A tighter limit on code size is given by the linear programming bound [41]. An $((n, K, d))$ code can exist only if there are homogeneous polynomials $A(x, y)$, $B(x, y)$, and $S(x, y)$ of degree n such that

$$A(1, 0) = 1, \tag{4.32}$$

$$B(x, y) = KA\left(\frac{x + 3y}{2}, \frac{x - y}{2}\right), \tag{4.33}$$

$$S(x, y) = KA\left(\frac{x + 3y}{2}, \frac{y - x}{2}\right), \tag{4.34}$$

$$B(1, y) - A(1, y) = O(y^d), \tag{4.35}$$

$$A(x, y) \geq 0, \tag{4.36}$$

$$B(x, y) - A(x, y) \geq 0, \tag{4.37}$$

Table 4.2: Bounds on the maximum k of an $[[n, k, d]]$ stabilizer code for $1 \leq n \leq 15$ and $2 \leq d \leq 5$.

$n\backslash d$	2	3	4	5
1	–	–	–	–
2	0	–	–	–
3	0	–	–	–
4	2	–	–	–
5	2	1	–	–
6	4	1^{C}	0	–
7	4	1	$-^{\mathrm{A}}$	–
8	6	3	0	–
9	6	3	0	–
10	8	4	2	–
11	8	5	2	1
12	10	6	4	1^{C}
13	10	7	4^{B}	1
14	12	8	6	$2-3$
15	12	9	6^{A}	3^{A}

$$S(x, y) \geq 0. \tag{4.38}$$

Here, $C(x, y) \geq 0$ means that the coefficients of the polynomial C are non-negative, and $O(y^d)$ is a polynomial in y with no terms of degree less than d. A pure $((n, K, d))$ code can exist only if Eqs. (4.32)-(4.38) can be satisfied along with

$$A(1, y) = 1 + O(y^d). \tag{4.39}$$

The linear programming bound is monotonic [109], meaning that if the constraints can be satisfied for some K, then they can be satisfied for all lower code dimensions too. This monotonicity holds even if K is allowed to be a real number (rather than just an integer). Following Ref. [110], we define the real number $K(n, d)$ as the largest $K > 1$ for which Eqs. (4.32)-(4.38) can be satisfied. The purity conjecture of Ref. [41] states that if the linear programming constraints hold for $K = K(n, d)$, then $A(1, y) = 1 + O(y^d)$. The content of this conjecture is simply that the linear programming bound for pure codes is the same as for potentially impure codes. This conjecture has been verified to hold for $n \leq 100$ [110].

For stabilizer codes, bounds on maximum k are given in Table 4.2 for $1 \leq n \leq 15$ and $2 \leq d \leq 5$. All lower bounds are given by the best known stabilizer codes (these codes can be found at Ref. [111]). The unmarked upper bounds are given by the linear programming bound for $K = 2^k$ (determined using YALMIP [112]). If the lower and upper bounds coincide, then a single value is given; otherwise, they are separated by a dash. In the cases marked "A," the $[[7, 0, 4]]$, $[[15, 7, 4]]$, and $[[15, 4, 5]]$ codes that do not violate the linear programming bound are excluded by arguments given in Sec. 7 of Ref. [41]. In the case marked "B," the $[[13, 5, 4]]$ code that does not violate the linear programming bound is excluded by the argument of Ref. [113]. The entries marked "C" indicate cases where a code meeting the bound must be impure (also outlined in Sec. 7 of Ref. [41]). An extended version of Table 4.2 for $n \leq 256$ is available at Ref. [111].

Table 4.3 gives the bounds on maximum K for a potentially nonadditive $((n, K, d))$ code where

Table 4.3: Bounds on the maximum size K of an $((n, K, d))$ code for $1 \leq n \leq 15$ and $2 \leq d \leq 5$.

$n\backslash d$	2	3	4	5
1	–	–	–	–
2	1	–	–	–
3	1	–	–	–
4	4	–	–	–
5	[A]6	2	–	–
6	16	2	1	–
7	[A]24 – 26	2 – 3	0 – 1	–
8	64	8 – 9	1	–
9	[A]96 – 112	[C]12 – 13	1	–
10	256	[D]24	4 – 5	–
11	[B]386 – 460	32 – 53	4 – 7	2
12	1 024	64 – 89	16 – 20	2
13	[B]1 586 – 1 877	128 – 204	16 – 40	2 – 3
14	4 096	256 – 324	64 – 102	4 – 10
15	[B]6 476 – 7 606	512 – 580	64 – 150	8 – 18

$1 \leq n \leq 15$ and $2 \leq d \leq 5$. All upper bounds are from the linear programming bound. The lower bounds marked "A" are from the family of nonadditive $((2\alpha + 1, 3 \times 2^{2\alpha-3}, 2))$ codes of Ref. [20]. Those marked "B" are from the family of $((4\alpha + 2\beta + 3, M_{\alpha\beta}, 2))$ codes of Ref. [21], where $\beta \in \{0, 1\}$ and

$$M_{\alpha\beta} = \sum_{i=0}^{\alpha} \binom{4\alpha + 2\beta + 3}{2i + \beta}. \tag{4.40}$$

The lower bounds marked "C" and "D" correspond to the $((9, 12, 3))$ and $((10, 24, 3))$ codes of Refs. [18] and [19], respectively. All other lower bounds are given by the best known stabilizer codes.

4.3 Symmetric codes

An $((n, K, d))$ code must detect the set $\mathcal{E}^{d-1}$ as defined in Eq. (4.12). Note that $\mathcal{E}^1$, and hence $\mathcal{E}^{d-1}$ more generally, is invariant under any permutation of the Pauli matrices X, Y, and Z on any subset of qubits. As a result of this symmetry, we call $((n, K, d))$ codes symmetric codes. Furthermore, as outlined in Sec. 4.2.4, this symmetry means that if some code $\mathcal{Q}$ detects $\mathcal{E}^{d-1}$, then so does any equivalent code $\mathcal{Q}'$. It is therefore sufficient to consider only standard form codes when attempting to construct an optimal symmetric CWS code. Furthermore, we need only consider standard form codes based on representatives from different elements of $\mathcal{L}_n$. However, as outlined in Sec. 4.2.1, the size of $\mathcal{L}_n$ appears to grow exponentially, and it has only been enumerated for $n \leq 12$. Furthermore, constructing an optimal classical code for a given graph by finding a maximum clique is NP-hard as mentioned in Sec. 4.2.1. In this section, we explore methods of code construction that address these two obstacles.

Table 4.4: The fraction of elements of $\mathcal{L}_n$, $\mathcal{G}_n$, and $\mathcal{D}_n$ that yield optimal $K = 2^{n-2}$ codes for even $n \leq 10$ and $d = 2$. The values given for $n = 8$ and 10 are lower bounds.

n	$\mathcal{L}_n$	$\mathcal{G}_n$	$\mathcal{D}_n$
2	0.500	0.500	0.500
4	0.500	0.636	0.641
6	0.539	0.763	0.833
8	0.643	0.909	0.938
10	0.815	0.977	0.981

4.3.1 Distance-two codes

First we consider distance-two codes of even length. As outlined in Tables 4.2 and 4.3, there are even-length stabilizer codes with $k = n - 2$ that saturate the singleton bound for $n \leq 14$. In fact, there are stabilizer codes that saturate the bound for all even n [20]. Despite this, there is still some insight to be gained from constructing CWS codes with these parameters. For $n \leq 10$, it is feasible to exhaustively search $\mathcal{L}_n$ (that is, to construct a code based on a representative of each element of $\mathcal{L}_n$). Using the code size distribution over $\mathcal{L}_n$, it is possible to determine the distributions over $\mathcal{G}_n$ and $\mathcal{D}_n$ by counting the number of nonisomorphic and distinct graphs, respectively, in each element of $\mathcal{L}_n$ (see Sec. 4.2.1). As an example, the code size distributions for $n = 6$ are shown in Fig. 4.2. It can be seen that over 50%, 75%, and 80% of elements of $\mathcal{L}_6$, $\mathcal{G}_6$, and $\mathcal{D}_6$, respectively, yield optimal $K = 16$ codes. The fraction of elements of $\mathcal{L}_n$, $\mathcal{G}_n$, and $\mathcal{D}_n$ that yield optimal codes for even $2 \leq n \leq 10$ is shown in Table 4.4. For $2 \leq n \leq 6$, the clique graphs generated are small enough for maximum cliques to be found using the exact algorithm of Ref. [114]. For $n \geq 8$, we have resorted to using the approximate PLS algorithm due to the larger clique graphs. We have allowed the PLS algorithm 100 attempts, each of which used a maximum of 1 000 selections (these are the default PLS parameters that we have employed). As a result of having used an approximate clique finding algorithm, the values given in the $n = 8$ and 10 rows of Table 4.4 are a lower bounds. It can be seen that in each case, the fraction of elements in $\mathcal{D}_n$ yielding an optimal code is greater than that of $\mathcal{G}_n$, which in turn, is greater than that of $\mathcal{L}_n$. Furthermore, increasing n increases the fraction of optimal codes in all cases. In particular, by $n = 10$, over 98% of distinct graphs yield a code with an optimal $K = 256$. This trend suggests that for larger n, we are highly likely to find an optimal code even if we use a randomly selected graph. This goes some way to explaining the results of Ref. [115], where cycle graphs were shown to give optimal codes for even $n \leq 12$.

The case of odd n is somewhat more interesting. Here, as shown in Ref. [20], the linear programming bound reduces to

$$K \leq 2^{n-2}\left(1 - \frac{1}{n-1}\right). \tag{4.41}$$

Stabilizer codes cannot saturate this bound and are restricted to $k \leq n - 3$. Again, we can construct codes based on an exhaustive search of $\mathcal{L}_n$ for $n \leq 11$. For $n = 3$, a single element of $\mathcal{L}_3$ yields an optimal $K = 1$ code. Similarly, a single element of $\mathcal{L}_5$ yields a code with $K = 6$, which matches the size of the optimal code given in Refs. [17, 20]. For $n = 7$, there is more of a spread in the code sizes as shown in Fig. 4.3. It can be seen that a large number of graphs

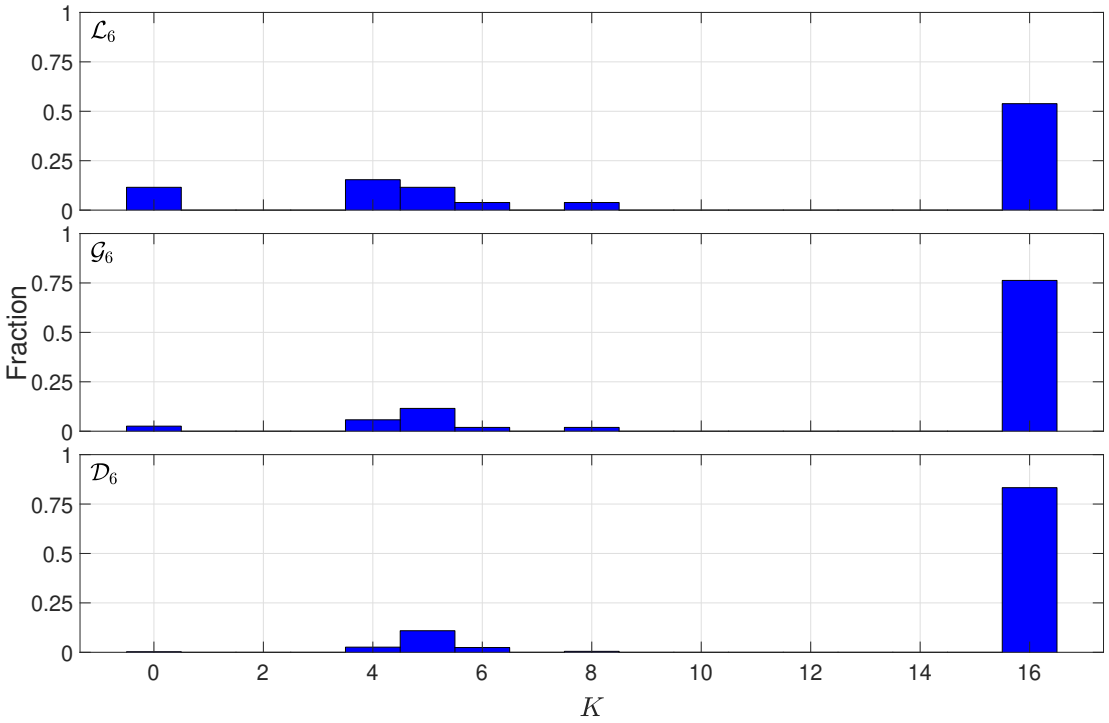

Figure 4.2: Code size distributions for non-LC-isomorphic, nonisomorphic, and distinct graphs in the case of $n = 6$ and $d = 2$.

yield codes with $K = 16$ or 22, which match the size of an optimal stabilizer code and the code of Ref. [21], respectively. Furthermore, there are seven elements of $\mathcal{L}_7$ that yield codes with $K = 24$, which match the size of the code of Ref. [20]. No graphs yield codes with $K = 25$ or 26 despite such codes not being excluded by the linear programming bound.

For $n = 9$, an exhaustive search of $\mathcal{L}_9$ is still feasible; however, we have done so using the PLS clique finder, and as such, there may exist larger CWS codes than the ones reported here. Similar to the $n = 7$ case, the majority of graphs gave codes with $K = 64$, 93, or 96, which match the size of an optimal stabilizer code, the code of Ref. [21], and the code of Ref. [20], respectively. However, we have also found seven elements of $\mathcal{L}_9$ that yield codes with $K \geq 97$. To increase the likelihood that we have found maximum-size codes for these seven graphs, we have repeated the clique search for each of them using 10 000 attempts. This has resulted in one $K = 97$ code, two $K = 98$ codes, and four $K = 100$ codes. Representatives of the elements of $\mathcal{L}_n$ that yielded these codes are shown in Fig. 4.4. Note that we do not label the nodes as isomorphic graphs yield equivalent codes. Given below each of the drawings is the graph in graph6 format (see Ref. [92] for details). A classical code for each of these graphs is given in the Supplemental Material [116] (this is the case for all codes presented in this paper). While these $K \geq 97$ codes are larger than any previously known codes, they do not saturate the linear programming bound of $K = 112$.

For $n = 11$, we have performed an exhaustive search of $\mathcal{L}_{11}$ with an increased 10 000 PLS selections to account for the larger cliques. Here, we have mostly obtained codes with $K = 256$, 384, or 386, which match the size of an optimal stabilizer code, the code of Ref. [20], and the code of Ref. [21], respectively. We have also found 413 elements of $\mathcal{L}_{11}$ that yield codes with

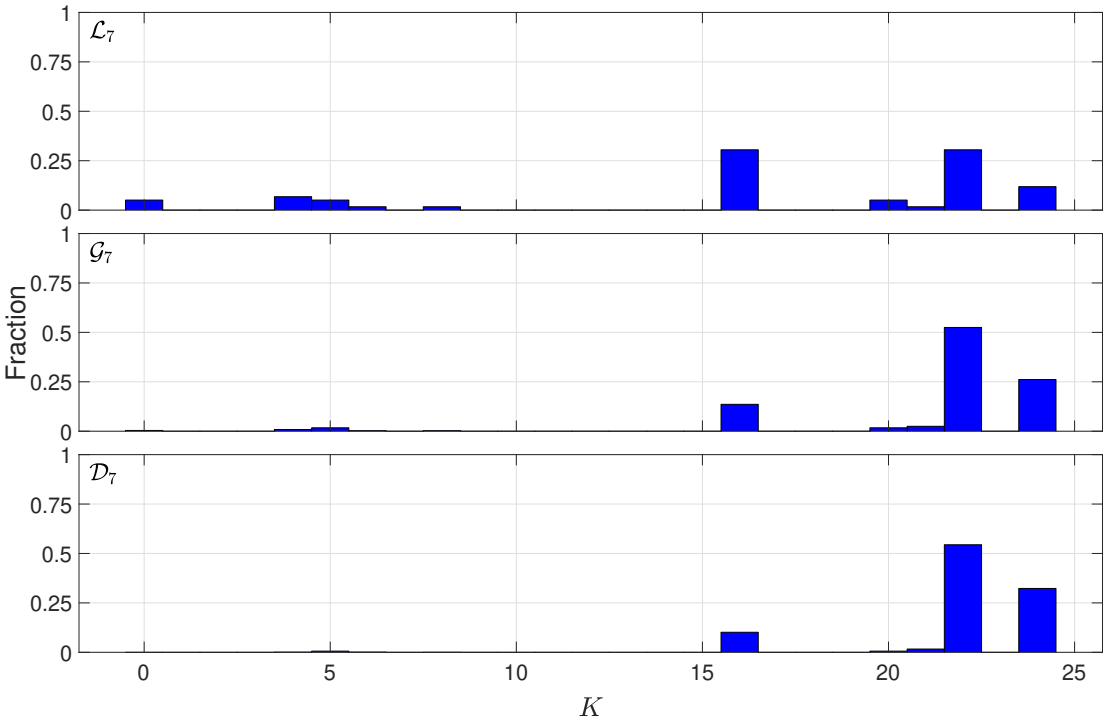

Figure 4.3: Code size distributions for non-LC-isomorphic, nonisomorphic, and distinct graphs in the case of $n = 7$ and $d = 2$.

Table 4.5: Number of elements N_K of $\mathcal{L}_{11}$ that gave codes of given size K with $d = 2$.

K	387	388	389	390	391	392	398	400	402	404	406	408	416
N_K	51	11	1	1	2	54	2	207	1	74	1	6	2

$K \geq 387$. As for the $n = 9$ case, we have repeated the clique search for these graphs using 10 000 attempts. The resulting code size distribution is given in Table 4.5. Representatives of elements of $\mathcal{L}_{11}$ that yield codes with $K \geq 406$ are shown in Fig. 4.5 (the remaining graphs are included in the Supplemental Material [116]). Again, while these are the largest codes known, they do not saturate the linear programming bound of $K = 460$. As $\mathcal{L}_n$ has not been enumerated for $n \geq 13$, we cannot continue this exhaustive search procedure for higher n. Any (nonexhaustive) search of $\mathcal{G}_n$ or $\mathcal{D}_n$ is also impractical for $n \geq 13$ due to the large clique graphs produced, which both makes the clique search slow and reduces the likelihood that the clique found is of maximum size.

Figure 4.6 shows the relationship between code size and clique graph order $|N_\mathcal{E}|$ for $4 \leq n \leq 11$. It can be seen that the data are clustered by clique graph order; furthermore, in each case, the graphs yielding the largest codes belong to the highest-$|N_\mathcal{E}|$ cluster. This clustering behavior can be explained by considering Eq. (4.30), which gives

$$|N_\mathcal{E}| = 2^n + 1 - |Cl_G(\mathcal{E})| - |D_G(\mathcal{E})| + |Cl_G(\mathcal{E}) \cap D_G(\mathcal{E})|. \tag{4.42}$$

It follows from Eq. (4.29) that $\mathrm{GF}(2)^n \backslash D_G(\mathcal{E})$ is the annihilator of $\mathcal{E}' = \{E \in \mathcal{E} : Cl_G(E) = 0\}$ and is therefore a subspace of $\mathrm{GF}(2)^n$. If $\dim[\mathrm{GF}(2)^n \backslash D_G(\mathcal{E})] = r \leq n$, then $|D_G(\mathcal{E})| = 2^n - 2^r$, which gives $|N_\mathcal{E}| = 2^r + 1 - |Cl_G(\mathcal{E})| + |Cl_G(\mathcal{E}) \cap D_G(\mathcal{E})|$. The clusters therefore correspond to different values of r. The codes in the highest-$|N_\mathcal{E}|$ cluster are pure as they have $D_G(\mathcal{E}) = \emptyset$. That this cluster contains codes of maximum size is not entirely surprising in light of the purity

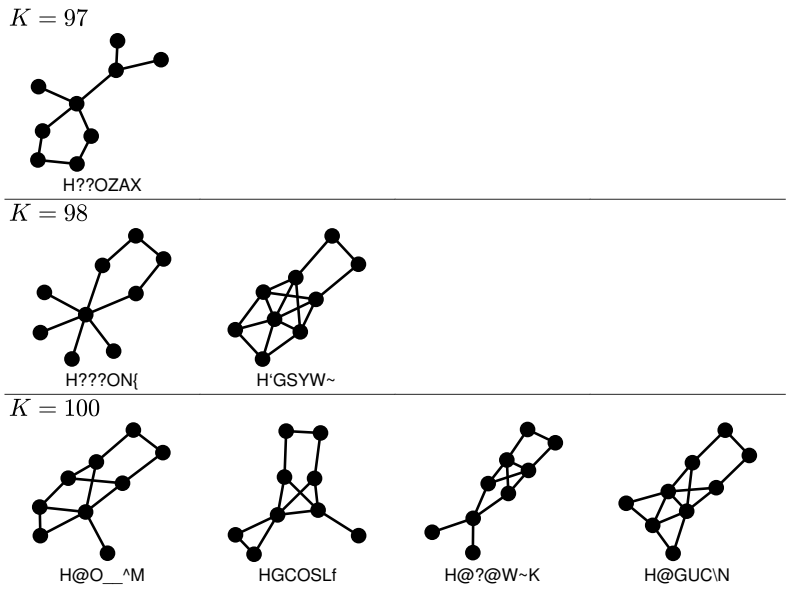

Figure 4.4: Non-LC-isomorphic graphs that yield $((9, 97 \leq K \leq 100, 2))$ codes.

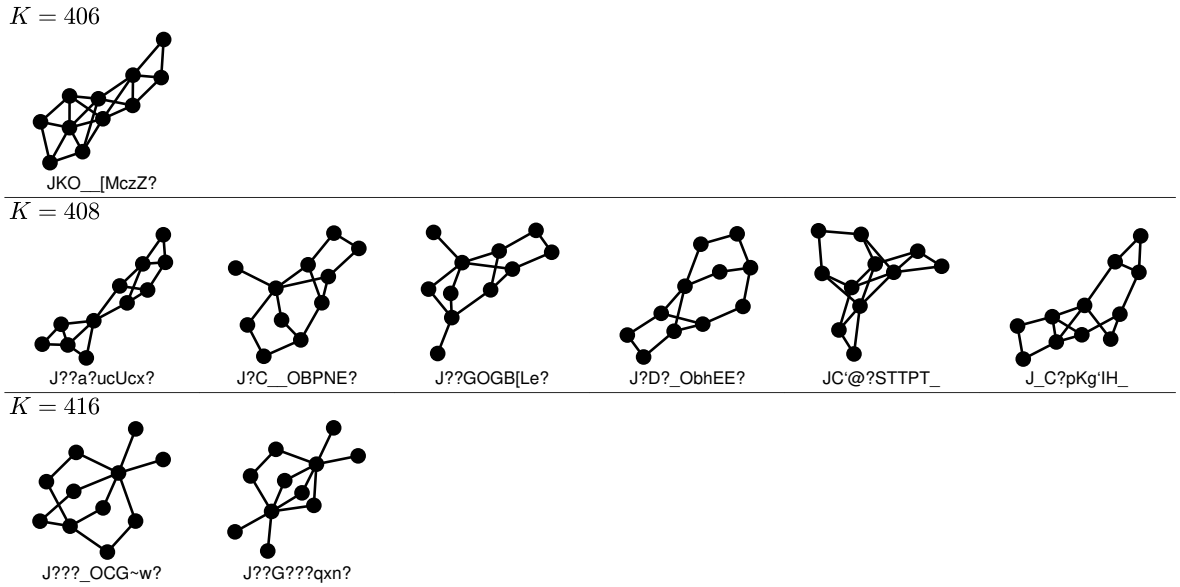

Figure 4.5: Non-LC-isomorphic graphs that yield $((11, 406 \leq K \leq 416, 2))$ codes.

conjecture outlined in Sec. 4.2.6.

4.3.2 Distance-three codes

Distance-three codes are of practical interest as they allow for the correction of an arbitrary single-qubit error. For $n \leq 11$, we can exhaustively search $\mathcal{L}_n$ in the same way as we have for the distance-two codes of the previous section. There are one and two elements of $\mathcal{L}_5$ and $\mathcal{L}_6$, respectively, that give optimal $K = 2$ codes (note that all $K = 2$ CWS codes are additive [33]). Similarly, there are 18 elements of $\mathcal{L}_7$ that yield $K = 2$ codes. As has been previously shown in Ref. [33], although the linear programming bound does not exclude them, there are no $((7, 3, 3))$ CWS codes. There are six elements of $\mathcal{L}_8$ that give $K = 8$ codes. No elements yield a $K = 9$ code despite such a code not being excluded by the linear programming bound. There are eight

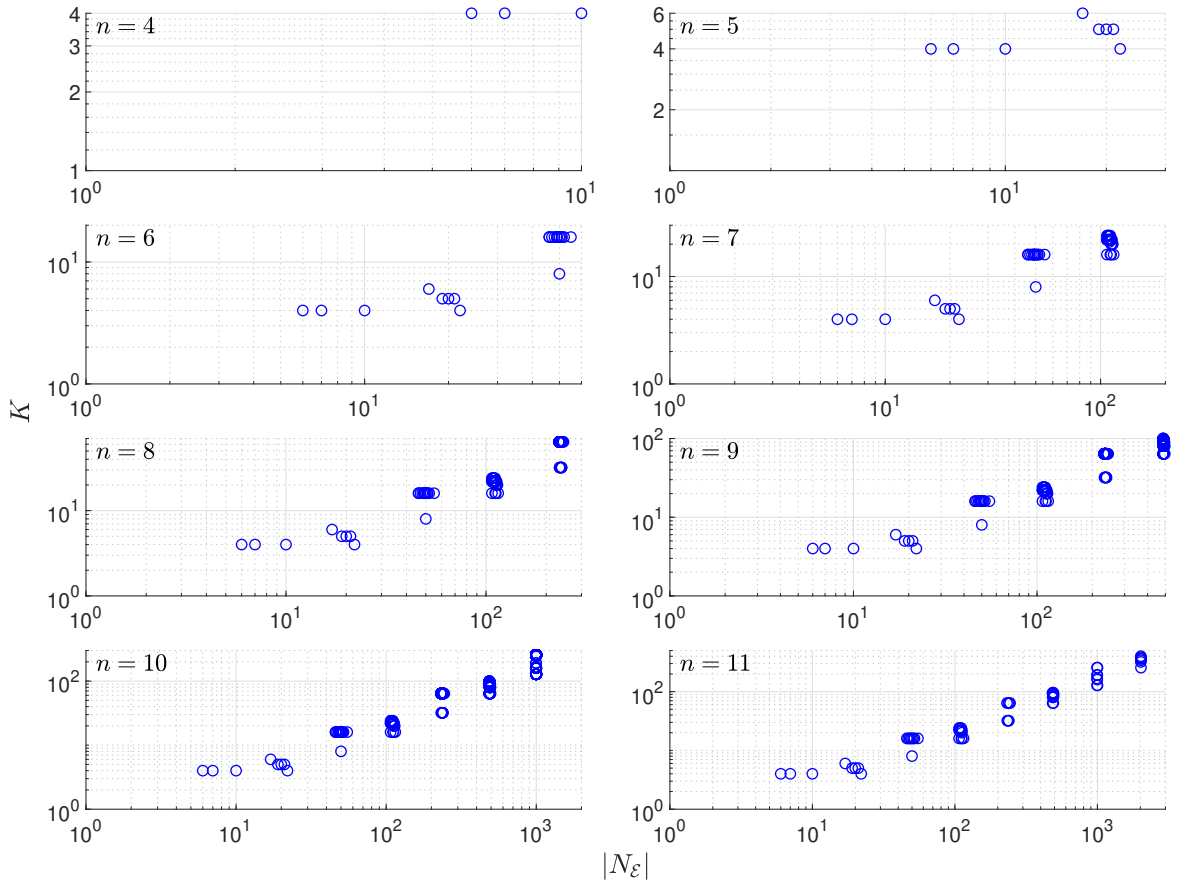

Figure 4.6: Code size versus clique graph order for codes with $4 \leq n \leq 11$ and $d = 2$.

elements of $\mathcal{L}_9$ that yield $K = 12$ codes, which match the size of the code presented in Ref. [18]. Again, no elements yield a $K = 13$ code despite such a code not being excluded by the linear programming bound. An exhaustive search of $\mathcal{L}_{10}$ has previously been performed in Ref. [19], where it was shown that a single element yields an optimal $K = 24$ code. We have exhaustively searched $\mathcal{L}_{11}$ using the PLS clique finder. This has yielded 13 709 $K = 32$ codes, which match the size of an optimal stabilizer code. No larger codes were found, which is somewhat surprising given that the linear programming bound is $K = 53$.

Figure 4.7 shows the relationship between code size and clique graph order for distance-three codes with $8 \leq n \leq 11$. It can be seen that there is greater spread within the clusters compared to the distance-two case of Fig. 4.6. According to Eq. (4.42), this can be attributed to an increased variance in the size of $Cl_G(\mathcal{E})$. Despite this increased variation, the graphs yielding the best codes still belong to the highest-$|N_\mathcal{E}|$ cluster in all four cases. Importantly, the best codes are not necessarily given by the graphs with the highest clique graph order within this cluster. For example, in the $n = 10$ case, the highest clique graph order cluster contains graphs with $613 \leq |N_\mathcal{E}| \leq 739$, while the graph yielding the $K = 24$ code only has $|N_\mathcal{E}| = 679$.

For $n = 12$, the size of $\mathcal{L}_{12}$ makes an exhaustive search somewhat prohibitive. We can reduce the search space somewhat by considering the distribution of clique graph orders as shown in Fig. 4.8. Note that by using Eq. (4.42), $|N_\mathcal{E}|$ can be computed without actually constructing the clique graph. Our previous observations regarding the relationship between code size and

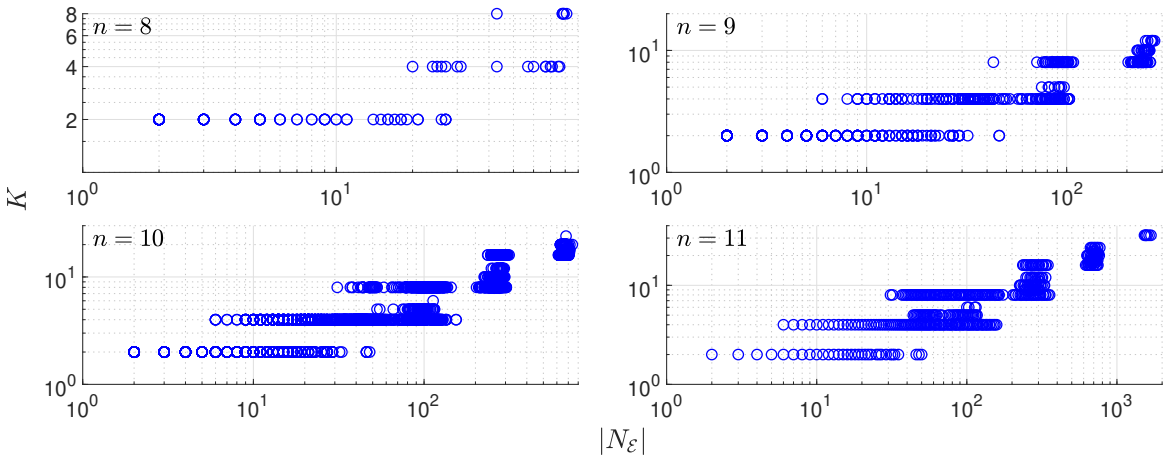

Figure 4.7: Code size versus clique graph order for codes with $8 \leq n \leq 11$ and $d = 3$.

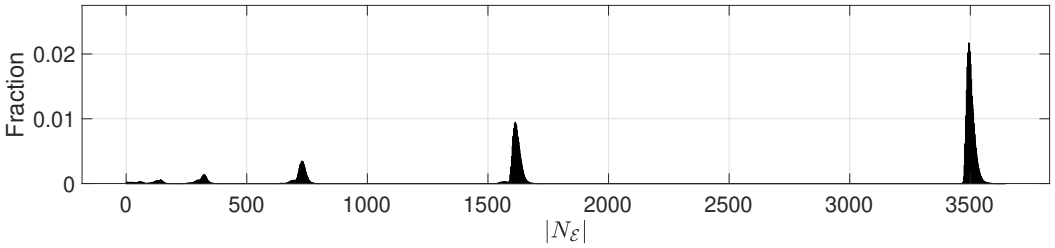

Figure 4.8: Clique graph order distribution over $\mathcal{L}_{12}$ for codes with with $d = 3$.

clique graph order suggest that graphs yielding the best codes are highly likely to be found in the $|N_\mathcal{E}| > 3\,000$ cluster. We have randomly selected 50 000 of the 663 039 elements of $\mathcal{L}_{12}$ in this cluster and constructed a code for each using the PLS clique finder. This has yielded 6 325 codes with $K = 64$, which match the size of an optimal stabilizer code. No larger codes were found despite the linear programming bound not excluding codes with up to $K = 89$. We have not pursued searches for $n \geq 13$ codes as while the clique graphs produced are smaller than in the $d = 2$ case, they are still large enough for maximum clique searches to be unreliable.

4.3.3 Distance-four codes

For $d = 4$, we are able to perform exhaustive searches of $\mathcal{L}_n$ for $n \leq 12$. There are one, five, and eight elements of $\mathcal{L}_6$, $\mathcal{L}_8$, and $\mathcal{L}_9$, respectively, that yield optimal $K = 1$ codes. As expected, no elements of $\mathcal{L}_7$ give a nontrivial code (note that a $K = 1$ CWS code is a stabilizer state and hence pure; such $[[n, 0, d]]$ codes have previously been classified in [35]). There are 10 and 3 060 elements of $\mathcal{L}_{10}$ and $\mathcal{L}_{11}$, respectively, that give $K = 4$ codes, which match the size of an optimal stabilizer code. No elements yield larger codes despite the linear programming bound not excluding codes with up to $K = 5$ and 7, respectively. Unlike the $d = 3$ case, an exhaustive search of $\mathcal{L}_{12}$ is feasible for $d = 4$ due to the smaller clique graphs. However, the clique graphs are still large enough that we have resorted to using the PLS clique finding algorithm. This search has yielded 1 482 codes with $K = 16$, which match the size of an optimal stabilizer code. No larger codes were found despite the linear programming bound not excluding codes with up to $K = 20$. The smaller clique graph sizes in the $d = 4$ case also make searching for codes with

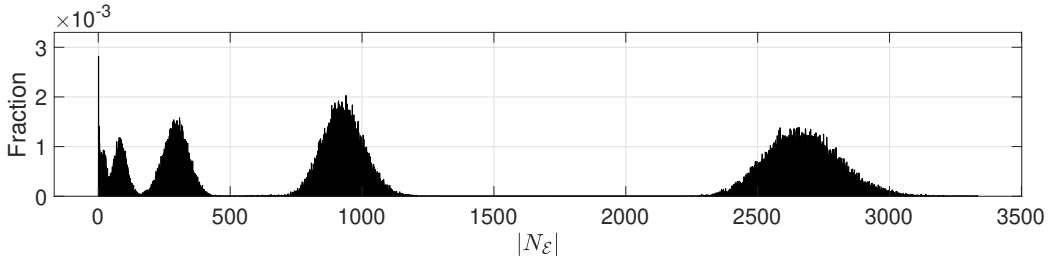

Figure 4.9: Clique graph order distribution over $\mathcal{D}_{13}$ for codes with with $d = 4$.

$n = 13$ and 14 feasible. For $n = 13$, we have randomly selected 100 000 graphs from $\mathcal{D}_{13}$ to estimate the clique graph size distribution as shown in Fig. 4.9. 41 458 of these graphs belong to the $|N_{\mathcal{E}}| > 2\,000$ cluster. Of these, one yielded a $K = 18$ code, which is larger than an optimal $K = 16$ stabilizer code.

To find more $n = 13$ codes with $K > 16$, we want a more reliable way of generating graphs that yield a large clique graph. That is, we wish to search $\mathcal{D}_n$ for graphs yielding a large clique graph in a way that is more efficient than a random search. We have found a genetic algorithm to be effective in this respect. There are a number of ways we could implement mutation and crossover in this algorithm. For mutation, we first select two nodes in the child graph at random. If these two nodes are not connected by an edge, then one is added; otherwise, if they are connected by an edge, then it is removed. If we represent the parent graphs as bit strings, then we can use standard single-point, two-point, or uniform crossover. One way to achieve this is to convert the upper triangular component of a parent adjacency matrix to a bit string row by row. Alternatively, we can use a graph-based approach. However, the method of Ref. [103] outlined in Sec. 4.2.2 is not appropriate for searching $\mathcal{D}_n$ as it is not guaranteed to produce child graphs with n nodes. Furthermore, as previously mentioned, it tends to remove an unnecessarily large number of edges when splitting the parent graphs into two fragments. To address these issues, we propose splitting the parent graphs using a spectral bisection. In particular, the nodes of a parent graph P are bisected into the sets N_1 and N_2, which define the fragments $F_1 = P[N_1]$ and $F_2 = P[N_2]$. A fragment is then exchanged between each parent to form two disconnected children that are then connected following the method of Ref. [103]. An example of this procedure on two $n = 10$ graphs is shown in Fig. 4.10.

We have run 100 genetic algorithm instances using each of the potential crossover methods to compare their performance. In each instance, we have used a population size of $N = 20$, 100 generations, a crossover probability of $p_c = 0.9$, a mutation probability of $p_m = 0.1$, and a tournament size of 10. We have also incorporated elitist selection, with the fittest two parent graphs (that is, the two that yield the largest clique graphs) being added to the child population at the start of each generation. The average order of the highest-order clique graph yielded in each generation is shown in Fig. 4.11. It can be seen that single-point, two-point, and uniform crossover (with $p_e = 0.5$) all exhibit similar performance. However, their performance is also matched by random crossover, where the two children are simply selected at random from $\mathcal{D}_n$ with no input from the parents. As such, the increase in fitness with successive generations when using these crossover methods is simply due to the selection pressure of the genetic algorithm.

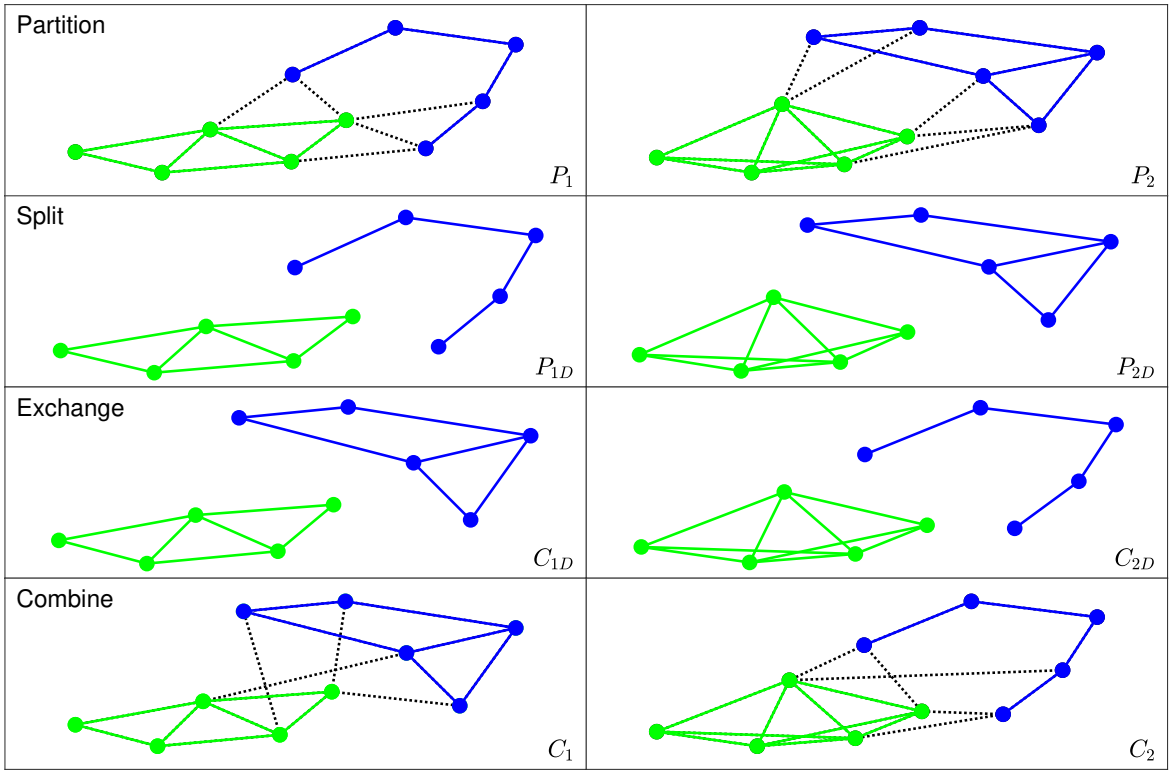

Figure 4.10: Spectral crossover example for $n = 10$ graphs. Each parent graph is split into two fragments according to a spectral bisection. These fragments are then exchanged and combined to form two child graphs.

It can also be seen that spectral crossover gives significantly better performance than all other methods. We have also tested the effect of population size when using spectral crossover. In particular, we have tested population sizes of $N = 10$ and 40 in addition to the previously considered $N = 20$ case. We have used a tournament size of half the population size in each case and left all other parameters unchanged. It can be seen in Fig. 4.11 that, as expected, increasing the population size increases the average maximum fitness. With clique graph order only serving as an indicator of code size, it is not essential for the genetic algorithm to find graphs that yield the absolute largest clique graphs. In fact, as was seen in the $n = 10$, $d = 3$ case, focusing solely on such graphs may mean that we miss the best code(s). With this in mind, we have found using 50 generations and a population size of $N = 10$ to be a good compromise. Using a modest population size and number of generations is also favorable from a run time perspective as determining $|N_{\mathcal{E}}|$ becomes more computationally expensive with increasing code length and/or distance (both of which serve to increase the size of the error set).

The genetic algorithm we have outlined is quite exploitative. To make our search more explorative, we run a large number of genetic algorithm instances, with a code being constructed from the fittest graph found by each instance. For $n = 13$, we have run 50 000 such instances, of which 352 yielded a $K = 18$ code and a further 175 gave a $K = 20$ code. The graphs that yielded codes with $K = 18$ and 20 belong to 35 and 25 different elements of $\mathcal{L}_{13}$, respectively. A representative from each of these elements is shown in Figs. 4.12 and 4.13. Note that the graphs shown are not necessarily the exact ones found using the genetic algorithm; they are LC-equivalent graphs that can be drawn clearly using the force-directed layout method of Ref.

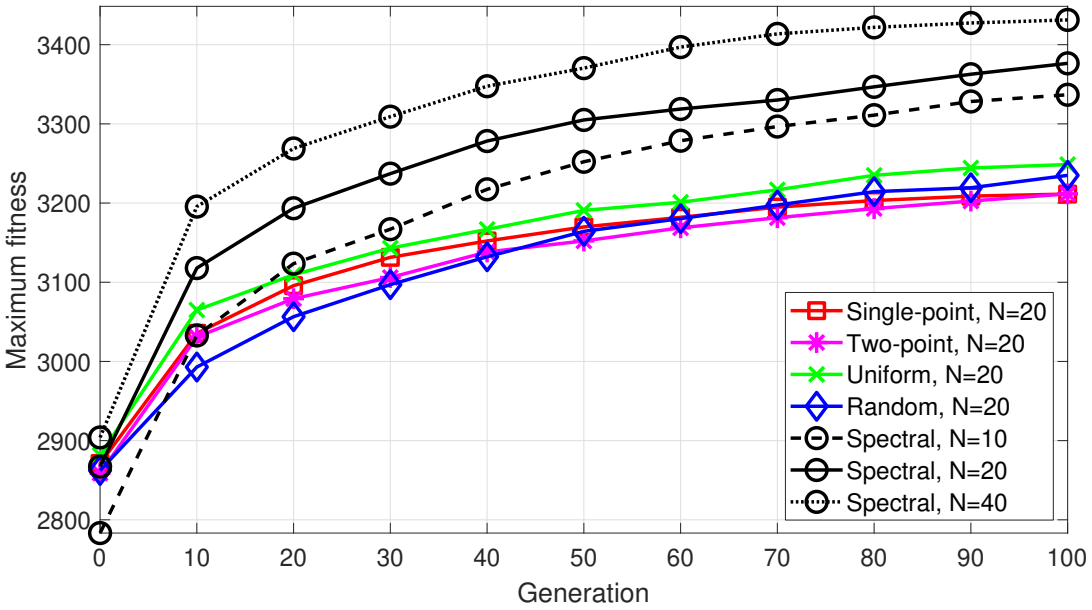

Figure 4.11: Comparison of crossover methods for $n = 13$, $d = 4$ codes. The vertical axis shows the fitness (the clique graph order $|N_{\mathcal{E}}|$) of the highest-fitness element of the child population averaged over 100 genetic algorithm instances.

[117]. While these $K = 18$ and 20 codes are larger than any previously known codes, they do not saturate the linear programming bound of $K = 40$. We have also run 50 000 instances of the genetic algorithm for $n = 14$. 65 of these instances have yielded $K = 64$ codes, which match the size of an optimal stabilizer code. We have not found any codes with $K > 64$ despite the linear programming bound not excluding codes with up to $K = 102$.

4.3.4 Distance-five codes

For $d = 5$, one and five elements of $\mathcal{L}_{11}$ and $\mathcal{L}_{12}$, respectively, yield optimal $K = 2$ codes. For $13 \leq n \leq 15$ we have run 50 000 genetic algorithm instances. 46 978 instances yielded a $K = 2$ code for $n = 13$, 452 instances yielded a $K = 4$ code for $n = 14$, and 14 instances yielded a $K = 8$ code for $n = 15$. No larger codes were found despite the linear programming bound being $K = 3$, 10, and 18, respectively. Note that the existence of a $((13, 3, 5))$ CWS code has already been excluded in Ref. [33] by the same argument that excluded the $((7, 3, 3))$ code.

4.4 Asymmetric codes

A channel of physical interest is the amplitude damping channel

$$\rho \to A_0 \rho A_0^\dagger + A_1 \rho A_1^\dagger, \tag{4.43}$$

where

$$A_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}, \; A_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}. \tag{4.44}$$

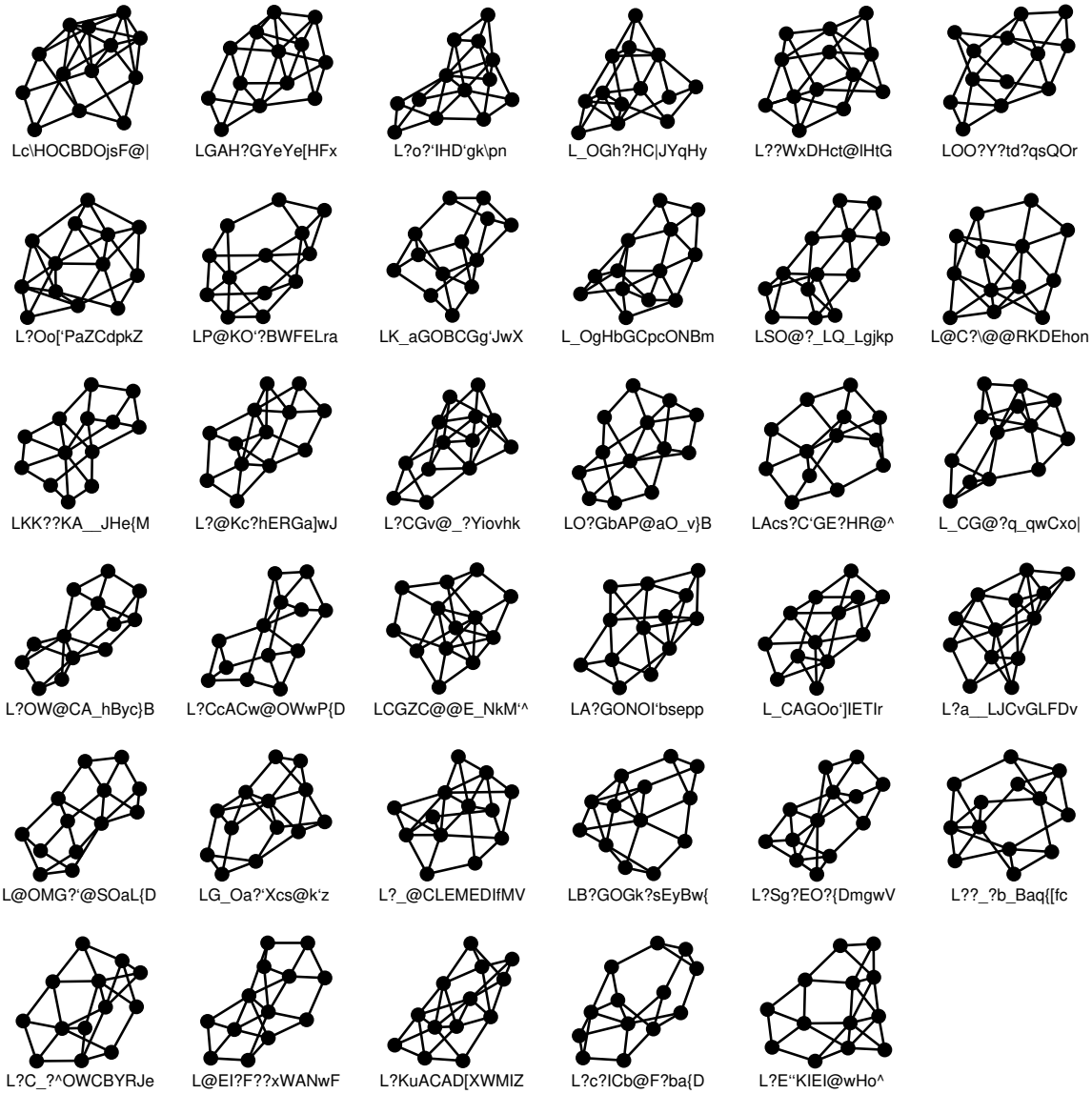

Figure 4.12: Non-LC-isomorphic graphs that yield $((13, 18, 4))$ codes.

It can be shown [118, 15, 119] that a sufficient condition for correcting a single amplitude damping error is the ability to detect

$$\mathcal{E}^{\{1\}} = \{I, X_i, Y_i, Z_i, X_iX_j, X_iY_j, Y_iY_j\}, \tag{4.45}$$

where $1 \leq i, j \leq n$. This is not a necessary condition for correcting an amplitude damping error. In fact, a code detecting $\mathcal{E}^{\{1\}}$ can also correct a single $A_1^\dagger$ error [90]. A code can correct t amplitude damping errors if it can detect $\mathcal{E}^{\{t\}}$, which is comprised of all t-fold combinations of elements from $\mathcal{E}^{\{1\}}$. $\mathcal{E}^{\{t\}}$ is a subset of $\mathcal{E}^{2t}$, which is the set of errors that must be detected to guarantee the ability to correct an arbitrary weight-t error. As a result, there is potential for constructing codes correcting t amplitude damping errors that are larger than those correcting t arbitrary errors. For example, the stabilizer codes presented in Ref. [15] detect $\mathcal{E}^{\{1\}}$ and have the parameters given in Table 4.6 (these values are taken from Ref. [120]). In all but the $n = 8$ case, these codes are larger than the size of an optimal $d = 3$ stabilizer code as given in Table 4.2. An exhaustive search for CWS codes detecting $\mathcal{E}^{\{1\}}$ has been performed in Ref. [90] for

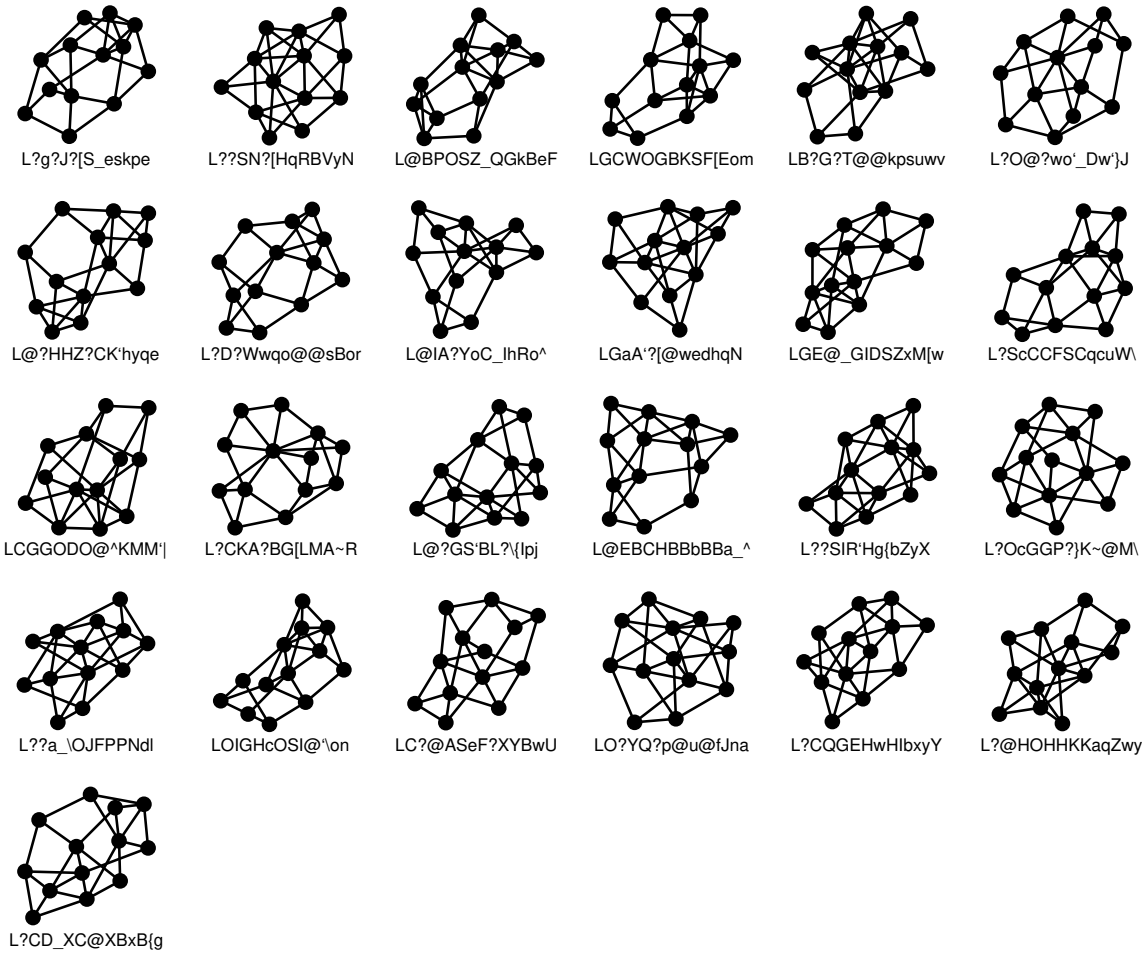

Figure 4.13: Non-LC-isomorphic graphs that yield $((13, 20, 4))$ codes.

Table 4.6: Size (K) of stabilizer codes presented in Ref. [15] and CWS codes presented in Ref. [90] that detect $\mathcal{E}^{\{1\}}$.

n	4	5	6	7	8	9	10	11	12	13	14	15
Stabilizer	1	2	4	8	8	16	32	64	128	256	512	1 024
CWS	–	2	4	8	10	20	–	–	–	–	–	–

$5 \leq n \leq 9$. The size of these codes is also given in Table 4.6, where they can be seen to be larger than the stabilizer codes for $n = 8$ and 9. Other nonadditive codes have also been constructed that can correct a single amplitude damping error [121, 120]; however, they are not directly comparable as they do so in a way that does not guarantee the detection of $\mathcal{E}^{\{1\}}$ (that is, they cannot correct an $A_1^\dagger$ error).

$\mathcal{E}^{\{t\}}$ is not invariant under all possible Pauli matrix permutations. As such, two LC-equivalent CWS codes need not correct the same number of amplitude damping errors. This means that considering standard form codes based on different elements of $\mathcal{L}_n$ no longer constitutes an exhaustive search of all CWS codes. However, as suggested in Ref. [90], a search of $\mathcal{L}_n$ can be made exhaustive by performing it for every LC-equivalent error set of the form $U^\dagger \mathcal{E}^{\{t\}} U$. These sets are versions of $\mathcal{E}^{\{t\}}$ with X, Y, and Z errors permuted on some subset of qubits. If $\mathcal{E}^{\{t\}}$ exhibited no symmetries under such permutations, then there would be 6^n such sets. However, as $\mathcal{E}^{\{t\}}$ is invariant under the permutation $X \leftrightarrow Y$ on any subset of qubits, this number is

Table 4.7: Number of elements of $\mathcal{G}_n$ that yield optimal $((n, K))$ CWS codes for the LC-equivalent error sets $\mathcal{E}^{\{1\}}$, $\mathcal{E}^{\{1\}}_{XZ}$, and $\mathcal{E}^{\{1\}}_{YZ}$. The values given for $n = 9$ are lower bounds.

$((n, K))$	$\mathcal{E}^{\{1\}}$	$\mathcal{E}^{\{1\}}_{XZ}$	$\mathcal{E}^{\{1\}}_{YZ}$
$((5, 2))$	5	9	3
$((6, 4))$	11	16	0
$((7, 8))$	114	157	181
$((8, 10))$	0	4	36
$((9, 20))$	0	6	44

reduced to 3^n. Unfortunately, an exhaustive search is not practical for codes with $n \geq 10$ as even for $n = 10$, there are $3^{10}|\mathcal{L}_{10}| = 235\,605\,510$ cases to test. In this section, we build on our code construction methods to address this increase in the size of the search space.

4.4.1 Single amplitude damping error

To construct new codes for the amplitude damping channel with $n \geq 10$, we first consider $n \leq 9$ to determine what types of codes match the bounds provided in Ref. [90]. Initially, we restrict consideration to standard form codes that detect $\mathcal{E}^{\{1\}}$. As $\mathcal{E}^{\{1\}}$ (and $\mathcal{E}^{\{t\}}$ more generally) is invariant under a permutation of qubit labels, it is sufficient to consider one representative from each element of $\mathcal{G}_n$. The first column of Table 4.7 shows the number of elements of $\mathcal{G}_n$ for $5 \leq n \leq 9$ that yield optimal standard form CWS codes. Note that the value given for $n = 9$ is a lower bound as we have used the PLS clique finder in this case. It can be seen that while we are able to construct optimal codes for $5 \leq n \leq 7$, we are unable to do so for $n = 8$ and 9. To remedy this, we consider the LC-equivalent error sets

$$\mathcal{E}^{\{1\}}_{XZ} = \{I, X_i, Y_i, Z_i, Z_iZ_j, Z_iY_j, Y_iY_j\}, \tag{4.46}$$

$$\mathcal{E}^{\{1\}}_{YZ} = \{I, X_i, Y_i, Z_i, X_iX_j, X_iZ_j, Z_iZ_j\}. \tag{4.47}$$

These are versions of $\mathcal{E}^{\{1\}}$ with the permutations $X \leftrightarrow Z$ and $Y \leftrightarrow Z$, respectively, on every qubit. More generally, we define $\mathcal{E}^{\{t\}}_{XZ}$ and $\mathcal{E}^{\{t\}}_{YZ}$ to be versions of $\mathcal{E}^{\{t\}}$ with the permutations $X \leftrightarrow Z$ and $Y \leftrightarrow Z$, respectively, on every qubit. Columns two and three of Table 4.7 show that exhaustive searches of $\mathcal{G}_n$ using the error sets $\mathcal{E}^{\{1\}}_{XZ}$ and $\mathcal{E}^{\{1\}}_{YZ}$ yield optimal codes for $n = 8$ and 9.

For $n = 10$, the size of $\mathcal{G}_{10}$ combined with the sizes of the clique graphs generated makes an exhaustive search impractical. However, we can still determine the distribution of clique graph sizes over $\mathcal{G}_n$ for the three error sets $\mathcal{E}^{\{1\}}$, $\mathcal{E}^{\{1\}}_{XZ}$, and $\mathcal{E}^{\{1\}}_{YZ}$ as shown in Fig. 4.14. For each of the three error sets, 50 000 graphs in the $|N_{\mathcal{E}}| > 600$ cluster have been selected. In each case, all 50 000 graphs yielded $K = 32$ codes, which match the size of the stabilizer code presented in Ref. [15].

For $n = 11$, an exhaustive search of $\mathcal{G}_{11}$ is impractical, even to simply determine clique graph sizes. We have therefore run 50 000 instances of our genetic algorithm for each of the three error

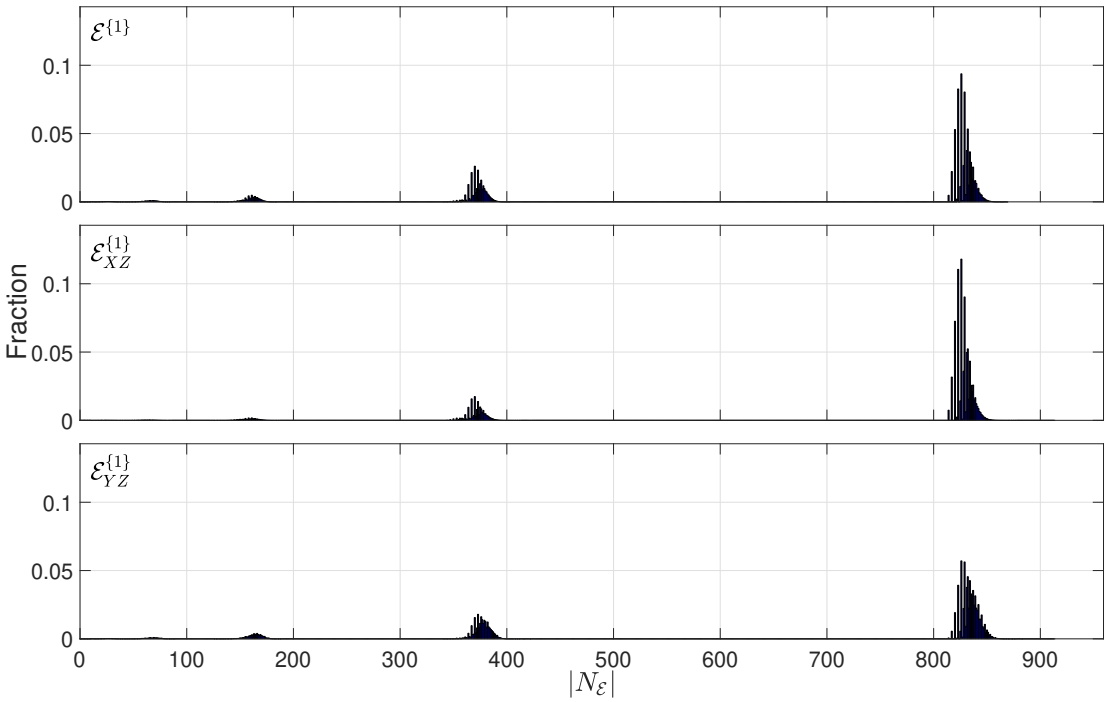

Figure 4.14: Distribution of clique graph order over $\mathcal{G}_{10}$ for the error sets $\mathcal{E}^{\{1\}}$, $\mathcal{E}^{\{1\}}_{XZ}$, and $\mathcal{E}^{\{1\}}_{YZ}$.

sets $\mathcal{E}^{\{1\}}$, $\mathcal{E}^{\{1\}}_{XZ}$, and $\mathcal{E}^{\{1\}}_{YZ}$. For $\mathcal{E}^{\{1\}}$, this has yielded a $K = 64$ code in every case. These codes match the size of the stabilizer code presented in Ref. [15]. For $\mathcal{E}^{\{1\}}_{YZ}$, 1 818 instances yielded codes with $K = 68$, which are larger than the best known stabilizer codes. 28 of these graphs are nonisomorphic and are shown in Fig. 4.15 (a simple circular node layout is used here as we do not have the freedom of picking an LC-isomorphic graph that can be drawn clearly using the force-directed layout method). For $\mathcal{E}^{\{1\}}_{XZ}$, only nine instances yielded codes with $K = 68$; however, there were also 71 instances that yielded codes with $K = 80$. Of these, two of the $K = 68$ graphs are nonisomorphic and two of the $K = 80$ graphs are nonisomorphic; these graphs are also shown in Fig. 4.15. For $n = 12$, applying the same genetic algorithm approach has yielded codes with $K = 128$, which match the size of the stabilizer code presented in Ref. [15]. In particular, of the 50 000 instances run for each error set, 21 535 gave a $K = 128$ code for $\mathcal{E}^{\{1\}}$, 34 906 gave a $K = 128$ code for $\mathcal{E}^{\{1\}}_{XZ}$, and 41 002 gave a $K = 128$ code for $\mathcal{E}^{\{1\}}_{YZ}$.

4.4.2 Two amplitude damping errors

As determined by exhaustive search in Ref. [90], there are no nontrivial CWS codes capable of detecting the error set $\mathcal{E}^{\{2\}}$ with $n \leq 8$. For $n = 9$, the largest CWS code that can detect $\mathcal{E}^{\{2\}}$ has $K = 2$. Interestingly, an exhaustive search of $\mathcal{G}_9$ fails to yield any $K = 2$ codes detecting $\mathcal{E}^{\{2\}}$. However, there are seven elements of $\mathcal{G}_9$ that yield $K = 2$ codes detecting $\mathcal{E}^{\{2\}}_{XZ}$ and 12 elements that yield $K = 2$ codes detecting $\mathcal{E}^{\{2\}}_{YZ}$. For $n = 10$, there are 32 elements of $\mathcal{G}_{10}$ that yield a $K = 2$ code detecting $\mathcal{E}^{\{2\}}$, 309 that yield a $K = 2$ code detecting $\mathcal{E}^{\{2\}}_{XZ}$, and 1 327 that yield a $K = 2$ code detecting $\mathcal{E}^{\{2\}}_{YZ}$. There are no larger standard form $n = 10$ codes detecting $\mathcal{E}^{\{2\}}$, $\mathcal{E}^{\{2\}}_{XZ}$, or $\mathcal{E}^{\{2\}}_{YZ}$.

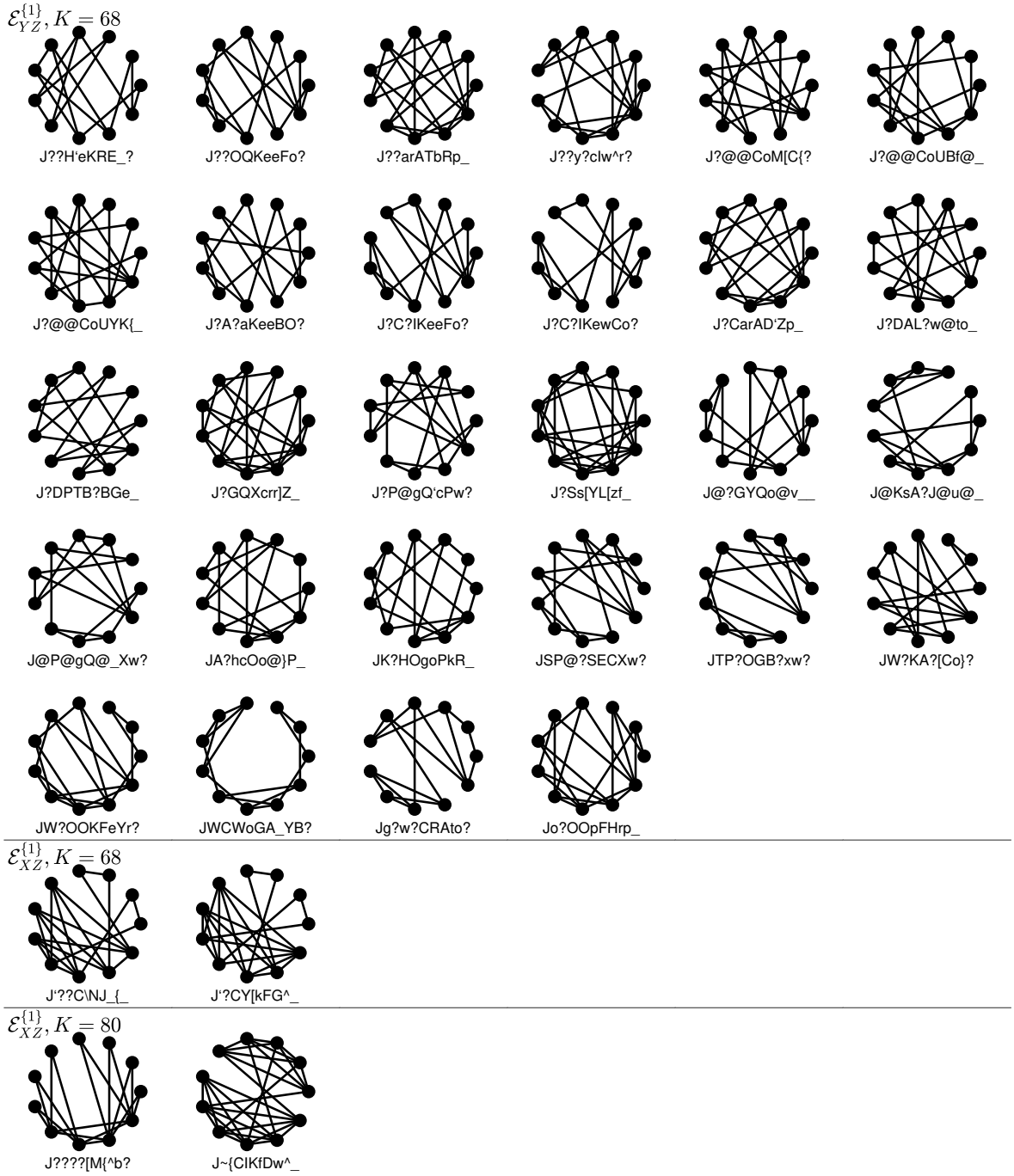

Figure 4.15: Nonisomorphic graphs yielding $((11, 68))$ codes detecting $\mathcal{E}_{YZ}^{\{1\}}$, $((11, 68))$ codes detecting $\mathcal{E}_{XZ}^{\{1\}}$, and $((11, 80))$ codes detecting $\mathcal{E}_{XZ}^{\{1\}}$.

Table 4.8: The number of genetic algorithm instances out of the 50 000 run that yielded an $((n, K))$ code detecting the given error set.

$((n, K))$	$\mathcal{E}^{\{2\}}$	$\mathcal{E}^{\{2\}}_{XZ}$	$\mathcal{E}^{\{2\}}_{YZ}$
$((11, 4))$	2	14	0
$((12, 4))$	45 912	36 275	43 225
$((13, 8))$	38 475	33 163	44 151
$((14, 16))$	3 467	5 840	13 148

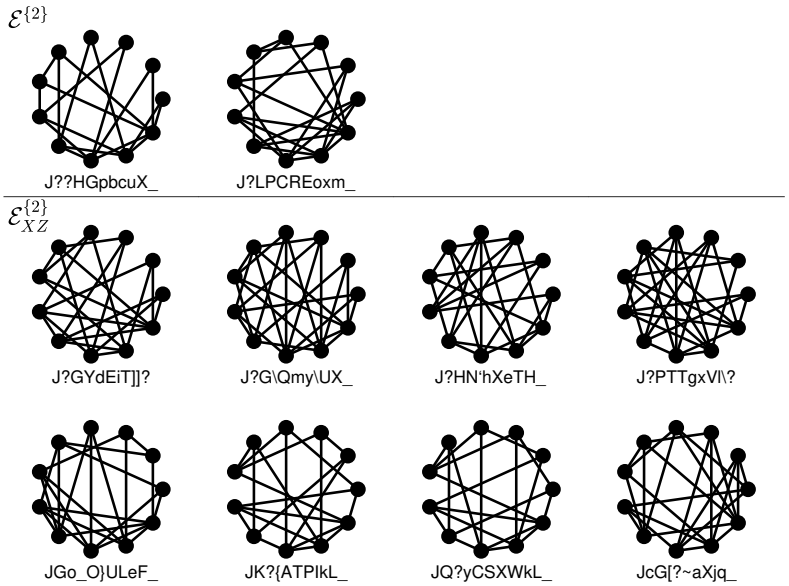

Figure 4.16: Nonisomorphic graphs yielding $((11, 4))$ codes detecting either $\mathcal{E}^{\{2\}}$ or $\mathcal{E}^{\{2\}}_{XZ}$.

As in the single-error-correcting case, any exhaustive search of $\mathcal{G}_n$ for $n \geq 11$ is impractical. For $11 \leq n \leq 14$, we have run 50 000 instances of the genetic algorithm outlined in Sec. 4.3.3 for each of the three error sets $\mathcal{E}^{\{2\}}$, $\mathcal{E}^{\{2\}}_{XZ}$, and $\mathcal{E}^{\{2\}}_{YZ}$. The best codes found have $K = 4$ for $n = 11$ and 12, $K = 8$ for $n = 13$, and $K = 16$ for $n = 14$. The number of genetic algorithm instances yielding codes with these parameters is shown in Table 4.8. Note that nearly all of these codes are stabilizer codes. For $n \leq 13$, we have used an exact clique finder, whereas for $n = 14$, we have used PLS. The $n = 11$ codes are interesting due to how difficult they are to find. The two graphs found for $\mathcal{E}^{\{2\}}$ are nonisomorphic, and eight of those found for $\mathcal{E}^{\{2\}}_{XZ}$ are nonisomorphic. These graphs are shown in Fig. 4.16. It is easy to find graphs giving codes with $K = 4$ codes for $n = 12$, $K = 8$ for $n = 13$, and $K = 16$ for $n = 14$ (they can be found quickly even with a simple random search). However, to the best of our knowledge, no stabilizer codes with these parameters have been previously published. Furthermore, they are all larger than an optimal $d = 5$ stabilizer code that can correct two arbitrary errors. As such, we include graphs yielding codes of these sizes for $\mathcal{E}^{\{2\}}$, $\mathcal{E}^{\{2\}}_{XZ}$, and $\mathcal{E}^{\{2\}}_{YZ}$ in Fig. 4.17.

4.5 Conclusion

We have demonstrated the effectiveness of a number of heuristic approaches to the construction of CWS codes. We have shown that using an approximate maximum clique finding algorithm

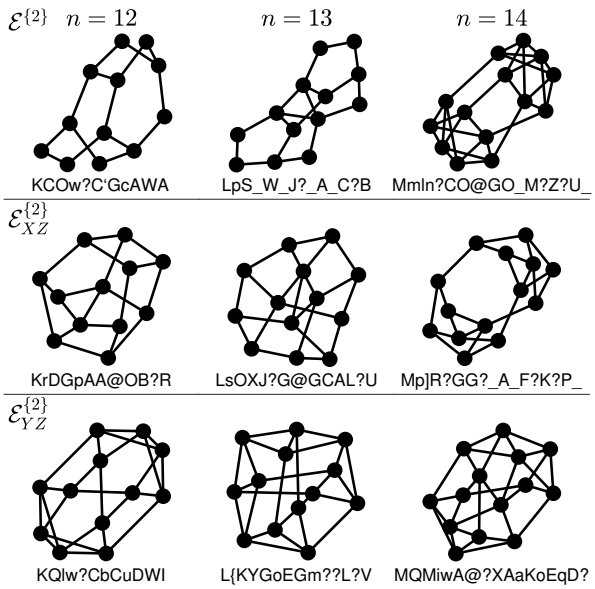

Figure 4.17: Graphs yielding $((12, 4))$, $((13, 8))$, or $((14, 16))$ codes detecting one of $\mathcal{E}^{\{2\}}$, $\mathcal{E}^{\{2\}}_{YZ}$, or $\mathcal{E}^{\{2\}}_{XZ}$.

makes finding larger codes practical. In particular, this has allowed us to find $((9, 97 \leq K \leq 100, 2))$ and $((11, 387 \leq K \leq 416, 2))$ codes that are larger than the best known nonadditive codes. We have demonstrated a clustering of clique graph orders and shown a relationship between clique graph order and code size. Furthermore, we have shown that graphs yielding large clique graphs can be found using a genetic algorithm with a crossover operation based on spectral bisection. This search strategy has yielded $((13, 18, 4))$ and $((13, 20, 4))$ codes, which are larger than any previously known code. Finally, we have shown that good codes correcting amplitude damping errors can be found by considering standard form codes that detect one of only three of the 3^n possible LC-equivalent error sets. Coupling this with the genetic algorithm approach, we have found $((11, 68))$ and $((11, 80))$ codes capable of correcting a single amplitude damping error. We have also found $((11, 4))$, $((12, 4))$, $((13, 8))$, and $((14, 16))$ stabilizer codes capable of correcting two amplitude damping errors.

Chapter 5

Conclusion

This thesis has investigated new heuristic methods for the design of quantum codes and their decoders. As previously noted, this investigation has been comprised of three main components, corresponding the the papers located in Chapters 2 to 4. Here, a high-level review of the main results is provided that aims to complement the more technical conclusions located in each chapter. Future research directions are then detailed that build on these results.

5.1 Summary

One of the reasons that quantum low-density parity-check (QLDPC) codes are attractive is that they allow the use of low-complexity belief propagation decoders [23]. However, the performance of these decoders is typically somewhat less impressive than their classical counterparts. This can be attributed, at least in part, to the large number of unavoidable short cycles in a QLDPC code's factor graph as well as the failure of belief propagation to address the degenerate nature of quantum errors [25]. In Chapter 2, modified belief propagation decoders were developed with the aim of mitigating these issues and hence providing improved decoding performance. While several decoders were presented, central among them is the augmented decoder, a conceptually simple decoder that has previously been proposed in the context of classical low-density parity-check (LDPC) codes [45]. If decoding fails (in a detectable way), then the augmented decoder iteratively reattempts decoding using a modified version of the code's factor graph. This heuristic modification involves simply duplicating a randomly selected subset of the graph's check nodes. Across a range of different QLDPC codes, it was demonstrated that the augmented decoder exhibits a significantly reduced error rate at the cost of a typically negligible complexity increase; furthermore, in each instance, the augmented decoder either outperformed or performed similarly to other modified decoders proposed in literature [25, 44, 42].

For a number of quantum channels of physical interest, phase-flip errors occur far more frequently than bit-flip errors [27, 28]. Analyzing the performance of stabilizer codes on these asymmetric channels is complicated by the #P-completeness of decoding [16], and unlike codes for the

depolarizing channel, distance is not a useful proxy for decoding error rate. In Chapter 3, new methods were developed for constructing good codes for asymmetric channels that address these issues. It was first shown that the error rate of an optimal stabilizer decoder can be approximated by considering only a small fraction the possible errors caused by the channel. This approximation was then used to identify a number of cyclic stabilizer codes that perform well on two different asymmetric channels, generalizing the result of Ref. [29]. Also demonstrated was a heuristic for assessing the performance of a stabilizer code based on the error rate of an associated binary classical code, which is several orders of magnitude faster to calculate and can itself be approximated using a limited error set. Furthermore, it was shown that this heuristic can be used as the basis for a hill-climbing search [a classic (meta)heuristic algorithm] that aims to optimize the performance of this classical code. Such searches yielded a number of highly performant stabilizer codes satisfying various structure constraints.

The family of codeword stabilized (CWS) codes [32, 33], which generalize the stabilizer codes, provide perhaps the most promising framework for the design of nonadditive codes. However, there are two major obstacles in constructing optimal CWS codes. The first of these is the NP-hardness of the required clique search [37], and the second is the exponential growth with code length in the number of inequivalent graphs on which a code can be based [34, 35, 36]. In Chapter 4, new methods for constructing CWS codes were developed that address these two obstacles. In particular, it was shown that the complexity of the clique search can be mitigated, at least in part, through the use of a heuristic clique finding algorithm. Using this approach, a number of distance-two codes were found that are larger than any previously known codes. To deal with the exponential growth of the search space, a similar approach was taken to that of Chapter 3 by first demonstrating a heuristic that can be used to assess graphs. In particular, it was shown that those graphs yielding large codes typically yield clique graphs with a large number of nodes. Motivated by this, a genetic algorithm [another classic (meta)heuristic algorithm] was developed that searched for graphs yielding large clique graphs. This genetic algorithm also built on the theme of heuristic graph modification introduced in Chapter 2 by making use of a crossover algorithm based on spectral bisection, which was shown to outperform more standard crossover operations. Using this search in tandem with the heuristic clique search, a number of distance-four codes were found that are larger than any previously known codes. Also continued was Chapter 3's theme of asymmetric code construction by showing that the search strategy used could be extended to codes correcting amplitude damping errors. This yielded a number of best known codes correcting either one or two amplitude damping errors.

In conclusion, this thesis has made a number of contributions in the field of quantum code and decoder design, and at a higher level, the nature of the methods used has demonstrated the effectiveness of heuristic approaches to complex problems in quantum error correction.

5.2 Future research

The augmented decoders used in Chapter 2 modified the factor graph by selecting check nodes for duplication at random. While the simplicity of this approach is attractive, it may be the

case that decoding performance can be improved by altering the selection method. For example, it may be possible to use data from previous failed decoding attempts to inform the selection of repeated check nodes for the next attempt. Alternatively, it may be more fruitful to perform offline optimization of the set of modified factor graphs using a training set of errors, which would involve a heuristic search similar to those employed in Chapters 3 and 4.

Also in Chapter 2, it was explained that decoding with a modified factor graph is equivalent to employing a modified belief propagation algorithm on the standard factor graph. In particular, the marginal probability estimation and error to check message calculations were modified according to Eqs. 2.65 and 2.66, respectively. There are a number of ways in which these modifications could be generalized; for example, allowing $r(i)$ to take on values other than zero or one would give control over the amount of feedback and/or message amplification. It would be interesting to investigate whether such alterations yield improved decoding performance.

While effective, the hill-climbing search used in Chapter 3 is quite simple. There is potential for more complex search algorithms to be able to find good codes faster, or to perhaps even find better codes. For example, it may be beneficial to use a population based search that is more similar to the genetic algorithm employed in Chapter 4. However, such a search would likely still rely solely on mutation as it is difficult to envisage a meaningful crossover operation that preserves the Abelian nature of the parent stabilizers.

The methods developed in Chapter 3 could be altered to permit searches for QLDPC codes. This would involve replacing the decoding error rate calculation with a Monte Carlo–based estimation of a belief propagation decoder's error rate. In the case of classical LDPC codes, a similar approach based on a genetic algorithm has recently yielded promising results [122].

The standard form CWS codes considered in Chapter 4 are the "graph codes" of Ref. [106], which generalize naturally from two-dimensional qubits to d-dimensional qudits. These qudit graph codes are based on multigraphs where any two nodes can be connected by up to $d-1$ edges. By modifying the proposed genetic algorithm to search the space of such multigraphs, which would involve a generalization of the spectral crossover operation, it may be possible to find best known nonadditive codes on qudits.

Bibliography

[1] P. W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, pp. 124–134, doi.org/10.1109/SFCS.1994.365700.

[2] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, doi.org/10.1137/S0097539795293172.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, Jul. 1996, pp. 212–219, doi.org/10.1145/237814.237866.

[4] C. H. Bennett and S. J. Wiesner, "Communication via One- and Two-Particle Operators on Einstein-Podolsky-Rosen States," *Physical Review Letters*, vol. 69, no. 20, pp. 2881–2884, Nov. 1992, doi.org/10.1103/PhysRevLett.69.2881.

[5] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels," *Physical Review Letters*, vol. 70, no. 13, pp. 1895–1899, Mar. 1993, doi.org/10.1103/PhysRevLett.70.1895.

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge, UK: Cambridge University Press, 2011, doi.org/10.1017/CBO9780511976667.

[7] J. L. Park, "The Concept of Transition in Quantum Mechanics," *Foundations of Physics*, vol. 1, no. 1, pp. 23–33, Mar. 1970, doi.org/10.1007/BF00708652.

[8] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, Oct. 1982, doi.org/10.1038/299802a0.

[9] D. Dieks, "Communication by EPR devices," *Physics Letters A*, vol. 92, no. 6, pp. 271–272, Nov. 1982, doi.org/10.1016/0375-9601(82)90084-6.

[10] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical Review A*, vol. 52, no. 4, p. R2493, Oct. 1995, doi.org/10.1103/PhysRevA.52.R2493.

[11] J. Preskill, "Reliable quantum computers," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 385–410, Jan. 1998, doi.org/10.1098/rspa.1998.0167.

[12] J. Preskill, "Lecture notes for Physics 219: Quantum Computation," theory.caltech.edu/people/preskill/ph229.

[13] D. A. Lidar and T. A. Brun, *Quantum Error Correction*. Cambridge, UK: Cambridge University Press, 2013, doi.org/10.1017/CBO9781139034807.

[14] K. Kraus, *States, Effects and Operations: Fundamental Notions of Quantum Theory*. Berlin, Germany: Springer-Verlag, 1983, doi.org/10.1007/3-540-12732-1.

[15] D. E. Gottesman, "Stabilizer Codes and Quantum Error Correction," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, USA, 1997, arxiv.org/abs/quant-ph/9705052.

[16] P. Iyer and D. Poulin, "Hardness of Decoding Quantum Stabilizer Codes," *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 5209–5223, Sep. 2015, doi.org/10.1109/TIT.2015.2422294.

[17] E. M. Rains, R. H. Hardin, P. W. Shor, and N. J. A. Sloane, "A Nonadditive Quantum Code," *Physical Review Letters*, vol. 79, no. 5, pp. 953–954, Mar. 1997, doi.org/10.1103/PhysRevLett.79.953.

[18] S. Yu, Q. Chen, C. H. Lai, and C. H. Oh, "Nonadditive Quantum Error-Correcting Code," *Physical Review Letters*, vol. 101, no. 9, p. 090501, Aug. 2008, doi.org/10.1103/PhysRevLett.101.090501.

[19] S. Yu, Q. Chen, and C. H. Oh, "Graphical Quantum Error-Correcting Codes," Sep. 2007, arxiv.org/abs/0709.1780.

[20] E. M. Rains, "Quantum Codes of Minimum Distance Two," *IEEE Transactions on Information theory*, vol. 45, no. 1, pp. 266–271, Jan. 1999, doi.org/10.1109/18.746807.

[21] J. A. Smolin, G. Smith, and S. Wehner, "Simple Family of Nonadditive Quantum Codes," *Physical Review Letters*, vol. 99, no. 13, p. 130505, Sep. 2007, doi.org/10.1103/PhysRevLett.99.130505.

[22] M. Grassl and M. Rötteler, "Quantum Goethals-Preparata Codes," in *2008 IEEE International Symposium on Information Theory*, Jul. 2008, pp. 300–304, doi.org/10.1109/ISIT.2008.4594996.

[23] D. J. C. 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, Sep. 2004, doi.org/10.1109/TIT.2004.834737.

[24] J. A. McGowan and R. C. Williamson, "Loop Removal from LDPC Codes," in *Proceedings 2003 IEEE Information Theory Workshop*, Mar. 2003, pp. 230–233, doi.org/10.1109/ITW.2003.1216737.

[25] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Information & Computation*, vol. 8, no. 10, pp. 987–1000, Nov. 2008, doi.org/10.26421/QIC8.10.

[26] A. Rigby, J. C. Olivier, and P. D. Jarvis, "Modified belief propagation decoders for quantum low-density parity-check codes," *Physical Review A*, vol. 100, no. 1, p. 012330, Jul. 2019, doi.org/10.1103/PhysRevA.100.012330.

[27] Z. W. E. Evans, A. M. Stephens, J. H. Cole, and L. C. L. Hollenberg, "Error correction optimisation in the presence of X/Z asymmetry," Sep. 2007, arxiv.org/abs/0709.3875.

[28] L. Ioffe and M. Mézard, "Asymmetric quantum error-correcting codes," *Physical Review A*, vol. 75, no. 3, p. 032345, Mar. 2007, doi.org/10.1103/PhysRevA.75.032345.

[29] A. Robertson, C. Granade, S. D. Bartlett, and S. T. Flammia, "Tailored Codes for Small Quantum Memories," *Physical Review Applied*, vol. 8, no. 6, p. 064004, Dec. 2017, doi.org/10.1103/PhysRevApplied.8.064004.

[30] 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. 050505, Jan. 2018, doi.org/10.1103/PhysRevLett.120.050505.

[31] A. Rigby, J. C. Olivier, and P. D. Jarvis, "Heuristic construction of codeword stabilized codes," *Physical Review A*, vol. 100, no. 6, p. 062303, Dec. 2019, doi.org/10.1103/PhysRevA.100.062303.

[32] A. Cross, G. Smith, J. A. Smolin, and B. Zeng, "Codeword Stabilized Quantum Codes," *IEEE Transactions on Information Theory*, vol. 55, no. 1, pp. 433–438, Jan. 2009, doi.org/10.1109/TIT.2008.2008136.

[33] I. Chuang, A. Cross, G. Smith, J. Smolin, and B. Zeng, "Codeword stabilized quantum codes: Algorithm and structure," *Journal of Mathematical Physics*, vol. 50, no. 4, p. 042109, Apr. 2009, doi.org/10.1063/1.3086833.

[34] L. E. Danielsen, "On Self-Dual Quantum Codes, Graphs, and Boolean Functions," Master's thesis, The University of Bergen, Bergen, Norway, 2005, arxiv.org/abs/quant-ph/0503236.

[35] L. E. Danielsen and M. G. Parker, "On the classification of all self-dual additive codes over GF(4) of length up to 12," *Journal of Combinatorial Theory, Series A*, vol. 113, no. 7, pp. 1351–1367, Oct. 2006, doi.org/10.1016/j.jcta.2005.12.004.

[36] L. E. Danielsen, "Database of Self-Dual Quantum Codes," ii.uib.no/~larsed/vncorbits.

[37] R. M. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations*. Boston, MA, USA: Springer, 1972, pp. 85–103, doi.org/10.1007/978-1-4684-2001-2_9.

[38] A. Rigby, J. C. Olivier, and P. D. Jarvis, "Optimizing short stabilizer codes for asymmetric channels," *Physical Review A*, vol. 101, no. 3, p. 032326, Mar. 2020, doi.org/10.1103/PhysRevA.101.032326.

[39] R. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962, doi.org/10.1109/TIT.1962.1057683.

[40] D. J. C. MacKay, "Good Error-Correcting Codes Based on Very Sparse Matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999, doi.org/10.1109/18.748992.

[41] A. R. Calderbank, E. M. Rains, P. M. Shor, and N. J. A. Sloane, "Quantum Error Correction Via Codes Over GF(4)," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1369–1387, Jul. 1998, doi.org/10.1109/18.681315.

[42] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen Years of Quantum LDPC Coding and Improved Decoding Strategies," *IEEE Access*, vol. 3, pp. 2492–2519, Nov. 2015, doi.org/10.1109/ACCESS.2015.2503267.

[43] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, "Quantum Error Correction and Orthogonal Geometry," *Physical Review Letters*, vol. 78, no. 3, pp. 405–408, Jan. 1997, doi.org/10.1103/PhysRevLett.78.405.

[44] Y.-J. Wang, B. C. Sanders, B.-M. Bai, and X.-M. Wang, "Enhanced Feedback Iterative Decoding of Sparse Quantum Codes," *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1231–1241, Feb. 2012, doi.org/10.1109/TIT.2011.2169534.

[45] A. Rigby, J. C. Olivier, H. C. Myburgh, C. Xiao, and B. P. Salmon, "Augmented decoders for LDPC codes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2018, no. 1, p. 189, Aug. 2018, doi.org/10.1186/s13638-018-1203-5.

[46] N. Delfosse and J.-P. Tillich, "A decoding algorithm for CSS codes using the X/Z correlations," in *2014 IEEE International Symposium on Information Theory*, Jun. 2014, pp. 1071–1075, doi.org/10.1109/ISIT.2014.6874997.

[47] E. Berlekamp, R. McEliece, and H. Van Tilborg, "On the Inherent Intractability of Certain Coding Problems," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, May 1978, doi.org/10.1109/TIT.1978.1055873.

[48] R. W. Hamming, "Error Detecting and Error Correcting Codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, Apr. 1950, doi.org/10.1002/j.1538-7305.1950.tb00463.x.

[49] T. Etzion, A. Trachtenberg, and A. Vardy, "Which Codes Have Cycle-Free Tanner Graphs?" *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 2173–2181, Sep. 1999, doi.org/10.1109/18.782170.

[50] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge, UK: Cambridge University Press, 2008, doi.org/10.1017/CBO9780511791338.

[51] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge, UK: Cambridge University Press, 2003, inference.org.uk/mackay/itila.

[52] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, no. 2, pp. 1098–1105, Aug. 1996, doi.org/10.1103/PhysRevA.54.1098.

[53] A. Steane, "Multiple-particle interference and quantum error correction," *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 452, no. 1954, pp. 2551–2577, Nov. 1996, doi.org/10.1098/rspa.1996.0136.

[54] I. B. Djordjevic, "Quantum LDPC Codes from Balanced Incomplete Block Designs," *IEEE Communications Letters*, vol. 12, no. 5, pp. 389–391, May 2008, doi.org/10.1109/LCOMM.2008.080083.

[55] R. C. Bose, "On the construction of balanced incomplete block designs," *Annals of Eugenics*, vol. 9, no. 4, pp. 353–399, Dec. 1939, doi.org/10.1111/j.1469-1809.1939.tb02219.x.

[56] M. Hagiwara and H. Imai, "Quantum Quasi-Cyclic LDPC Codes," in *2007 IEEE International Symposium on Information Theory*, Jun. 2010, doi.org/10.1109/ISIT.2007.4557323.

[57] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Construction of Quantum LDPC Codes From Classical Row-Circulant QC-LDPCs," *IEEE Communications Letters*, vol. 20, no. 1, pp. 9–12, Jan. 2016, doi.org/10.1109/LCOMM.2015.2494020.

[58] P. Tan and J. Li, "Efficient Quantum Stabilizer Codes: LDPC and LDPC-Convolutional Constructions," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 476–491, Jan. 2010, doi.org/10.1109/TIT.2009.2034794.

[59] M. C. Davey, "Error-correction using low-density parity-check codes," Ph.D. dissertation, University of Cambridge, Cambridge, UK, 2000, inference.org.uk/mcdavey/papers/davey_phd.html.

[60] P. K. Sarvepalli, A. Klappenecker, and M. Rötteler, "Asymmetric quantum codes: constructions, bounds and performance," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 465, no. 2105, pp. 1645–1672, Mar. 2009, doi.org/10.1098/rspa.2008.0439.

[61] S. A. Aly, "Asymmetric Quantum BCH Codes," in *2008 International Conference on Computer Engineering & Systems*, Nov. 2008, pp. 157–162, doi.org/10.1109/ICCES.2008.4772987.

[62] M. F. Ezerman, S. Jitman, S. Ling, and D. V. Pasechnik, "CSS-Like Constructions of Asymmetric Quantum Codes," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6732–6754, Oct. 2013, doi.org/10.1109/TIT.2013.2272575.

[63] L. Wang, K. Feng, S. Ling, and C. Xing, "Asymmetric Quantum Codes: Characterization and Constructions," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2938–2945, Jun. 2010, doi.org/10.1109/TIT.2010.2046221.

[64] G. G. La Guardia, "Asymmetric quantum codes: new codes from old," *Quantum Information Processing*, vol. 12, no. 8, pp. 2771–2790, Mar. 2013, doi.org/10.1007/s11128-013-0562-4.

[65] K. Guenda and T. A. Gulliver, "Symmetric and Asymmetric Quantum Codes," *International Journal of Quantum Information*, vol. 11, no. 05, p. 1350047, Sep. 2013, doi.org/10.1142/S0219749913500470.

[66] W. C. Huffman, "Additive cyclic codes over $\mathbb{F}_4$," *Advances in Mathematics of Communications*, vol. 1, no. 4, pp. 427–459, Nov. 2007, doi.org/10.3934/amc.2007.1.427.

[67] W. C. Huffman, "Additive cyclic codes over $\mathbb{F}_4$," *Advances in Mathematics of Communications*, vol. 2, no. 3, pp. 309–343, Aug. 2008, doi.org/10.3934/amc.2008.2.309.

[68] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, Netherlands: Elsevier, 1977, sciencedirect.com/bookseries/north-holland-mathematical-library/vol/16.

[69] A. Luo, "Good Additive Cyclic Quantum Error-Correcting Codes," Master's thesis, Concordia University, Montreal, Canada, 2004, spectrum.library.concordia.ca/8063.

[70] J. Napp and J. Preskill, "Optimal Bacon-Shor Codes," *Quantum Information & Computation*, vol. 13, no. 5&6, pp. 490–510, May 2013, doi.org/10.26421/QIC13.3-4.

[71] D. P. DiVincenzo, D. W. Leung, and B. M. Terhal, "Quantum Data Hiding," *IEEE Transactions on Information Theory*, vol. 48, no. 3, pp. 580–598, Aug. 2002, doi.org/10.1109/18.985948.

[72] J. Emerson, R. Alicki, and K. Życzkowski, "Scalable noise estimation with random unitary operators," *Journal of Optics B: Quantum and Semiclassical Optics*, vol. 7, no. 10, pp. S347–S352, Sep. 2005, doi.org/10.1088/1464-4266/7/10/021.

[73] C. Dankert, R. Cleve, J. Emerson, and E. Livine, "Exact and approximate unitary 2-designs and their application to fidelity estimation," *Physical Review A*, vol. 80, no. 1, p. 012304, Jul. 2009, doi.org/10.1103/PhysRevA.80.012304.

[74] M.-H. Hsieh and F. Le Gall, "NP-hardness of decoding quantum error-correction codes," *Physical Review A*, vol. 83, no. 5, p. 052331, May 2011, doi.org/10.1103/PhysRevA.83.052331.

[75] 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*, Oct. 2012, pp. 208–211, ieeexplore.ieee.org/document/6400919.

[76] K.-Y. Kuo and C.-C. Lu, "On the Hardnesses of Several Quantum Decoding Problems," Jun. 2013, arxiv.org/abs/1306.5173.

[77] F. Gaitan, *Quantum Error Correction and Fault-Tolerant Quantum Computing*. Boca Raton, FL, USA: CRC Press, 2008, doi.org/10.1201/b15868.

[78] I. Djordjevic, *Quantum Information Processing and Quantum Error Correction: An Engineering Approach*. Oxford, UK: Elsevier, 2012, doi.org/10.1016/C2010-0-66917-3.

[79] A. M. Steane, "Simple quantum error-correcting codes," *Physical Review A*, vol. 54, no. 6, pp. 4741–4751, Dec. 1996, doi.org/10.1103/PhysRevA.54.4741.

[80] R. A. Brualdi, *Introductory Combinatorics*, 5th ed. Upper Saddle River, NJ, USA: Pearson Education, 2010.

[81] G. E. Andrews and K. Eriksson, *Integer Partitions*. Cambridge, UK: Cambridge University Press, 2004, doi.org/10.1017/CBO9781139167239.

[82] S. Luke, *Essentials of Metaheuristics*. Morrisville, NC: Lulu, 2013, cs.gmu.edu/ sean/book/metaheuristics.

[83] A. M. Steane, "Error Correcting Codes in Quantum Theory," *Physical Review Letters*, vol. 77, no. 5, pp. 793–797, Jul. 1996, doi.org/10.1103/PhysRevLett.77.793.

[84] The Sage Developers, *SageMath, the Sage Mathematics Software System (Version 8.7)*, 2019, sagemath.org.

[85] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," *Physical Review A*, vol. 55, no. 2, pp. 900–911, Feb. 1997, doi.org/10.1103/PhysRevA.55.900.

[86] M. Van den Nest, J. Dehaene, and B. De Moor, "Graphical description of the action of local Clifford transformations on graph states," *Physical Review A*, vol. 69, no. 2, p. 022316, Feb. 2004, doi.org/10.1103/PhysRevA.69.022316.

[87] M. Grassl, A. Klappenecker, and M. Rötteler, "Graphs, Quadratic Forms, and Quantum Codes," in *Proceedings IEEE International Symposium on Information Theory*, Jun. 2002, p. 45, doi.org/10.1109/ISIT.2002.1023317.

[88] D. Schlingemann, "Stabilizer codes can be realized as graph codes," *Quantum Information & Computation*, vol. 2, no. 4, pp. 307–323, Jun. 2002, doi.org/10.26421/QIC2.4.

[89] W. Pullan, "Phased local search for the maximum clique problem," *Journal of Combinatorial Optimization*, vol. 12, no. 3, pp. 303–323, Aug. 2006, doi.org/10.1007/s10878-006-9635-y.

[90] T. Jackson, M. Grassl, and B. Zeng, "Codeword Stabilized Quantum Codes for Asymmetric Channels," in *2016 IEEE International Symposium on Information Theory*, Jul. 2016, pp. 2264–2268, doi.org/10.1109/ISIT.2016.7541702.

[91] B. D. McKay and A. Piperno, "Practical graph isomorphism, II," *Journal of Symbolic Computation*, vol. 60, no. 0, pp. 94–112, Jan. 2014, doi.org/10.1016/j.jsc.2013.09.003.

[92] B. D. McKay and A. Piperno, *nauty and Traces User's Guide (Version 2.6)*, 2017, users.cecs.anu.edu.au/~bdm/nauty.

[93] F. Harary and E. M. Palmer, *Graphical Enumeration*. New York, NY, USA: Elsevier, 1973, doi.org/10.1016/C2013-0-10826-4.

[94] Q. Wu and J.-K. Hao, "A review on algorithms for maximum clique problems," *European Journal of Operational Research*, vol. 242, no. 3, pp. 693–709, May 2015, doi.org/10.1016/j.ejor.2014.09.064.

[95] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W.H. Freeman and Company, 1979.

[96] K. M. Hall, "An r-Dimensional Quadratic Placement Algorithm," *Management Science*, vol. 17, no. 3, pp. 219–229, Nov. 1970, doi.org/10.1287/mnsc.17.3.219.

[97] W. E. Donath and A. J. Hoffman, "Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices," *IBM Technical Disclosure Bulletin*, vol. 15, no. 3, pp. 938–944, 1972.

[98] M. Fiedler, "A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory," *Czechoslovak Mathematical Journal*, vol. 25, no. 4, pp. 619–633, 1975, dml.cz/dmlcz/101357.

[99] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 430–452, Jul. 1990, doi.org/10.1137/0611030.

[100] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973, dml.cz/dmlcz/101168.

[101] D. Whitley, "A genetic algorithm tutorial," *Statistics and Computing*, vol. 4, no. 2, pp. 65–85, Jun. 1994, doi.org/10.1007/BF00175354.

[102] S. Legg, M. Hutter, and A. Kumar, "Tournament versus Fitness Uniform Selection," in *Proceedings of the 2004 Congress on Evolutionary Computation*, Jun. 2004, pp. 2144–2151, doi.org/10.1109/CEC.2004.1331162.

[103] A. Globus, J. Lawton, and T. Wipke, "Automatic molecular design using evolutionary techniques," *Nanotechnology*, vol. 10, no. 3, pp. 290–299, Sep. 1999, doi.org/10.1088/0957-4484/10/3/312.

[104] S. Stone, B. Pillmore, and W. Cyre, "Crossover and mutation in genetic algorithms using graph-encoded chromosomes," 2004, gpbib.cs.ucl.ac.uk/gecco2004/prof185.html.

[105] M. Hein, W. Dür, J. Eisert, R. Raussendorf, M. Nest, and H.-J. Briegel, "Entanglement in Graph States and its Applications," in *Quantum Computers, Algorithms and Chaos*. Amsterdam, Netherlands: IOS Press, 2005, pp. 115–218, doi.org/10.3254/978-1-61499-018-5-115.

[106] S. Y. Looi, L. Yu, V. Gheorghiu, and R. B. Griffiths, "Quantum-error-correcting codes using qudit graph states," *Physical Review A*, vol. 78, no. 4, p. 042303, Oct. 2008, doi.org//10.1103/PhysRevA.78.042303.

[107] Z. Ji, J. Chen, Z. Wei, and M. Ying, "The LU-LC conjecture is false," Sep. 2007, arxiv.org/abs/0709.1266.

[108] Y. Li, I. Dumer, M. Grassl, and L. P. Pryadko, "Structured error recovery for code-word-stabilized quantum codes," *Physical Review A*, vol. 81, no. 5, p. 052337, May 2010, doi.org/10.1103/PhysRevA.81.052337.

[109] E. M. Rains, "Monotonicity of the Quantum Linear Programming Bound," *IEEE Transactions on Information Theory*, vol. 45, no. 7, pp. 2489–2492, Nov. 1999, doi.org/10.1109/18.796387.

[110] G. Nebe, E. M. Rains, and N. J. A. Sloane, *Self-Dual Codes and Invariant Theory*. Berlin, Germany: Springer, 2006, doi.org/10.1007/3-540-30731-1.

[111] M. Grassl, "Bounds on the minimum distance of linear codes and quantum codes," codetables.de/.

[112] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *2004 IEEE International Conference on Robotics and Automation*, Sep. 2004, pp. 284–289, doi.org/10.1109/CACSD.2004.1393890.

[113] J. Bierbrauer, R. Fears, S. Marcugini, and F. Pambianco, "The Nonexistence of a [[13, 5, 4]]-Quantum Stabilizer Code," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4788–4793, Jul. 2011, doi.org/10.1109/TIT.2011.2146430.

[114] J. Konc and D. Janezic, "An improved branch and bound algorithm for the maximum clique problem," *MATCH Communications in Mathematical and in Computer Chemistry*, vol. 58, no. 3, pp. 569–590, 2007, match.pmf.kg.ac.rs/content58n3.htm.

[115] W.-T. Yen and L.-Y. Hsu, "Optimal Nonadditive Quantum Error-Detecting Code," Jan. 2009, arxiv.org/abs/0901.1353.

[116] See Supplemental Material at link.aps.org/supplemental/10.1103/PhysRevA.100.062303 for the graph and associated classical code for all the CWS codes.

[117] T. M. J. Fruchterman and E. M. Reingold, "Graph Drawing by Force-directed Placement," *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, Nov. 1991, doi.org/10.1002/spe.4380211102.

[118] M. Grassl, Z. Wei, Z.-Q. Yin, and B. Zeng, "Quantum Error-Correcting Codes for Amplitude Damping," in *2014 IEEE International Symposium on Information Theory*, Jun. 2014, pp. 906–910, doi.org/10.1109/ISIT.2014.6874964.

[119] A. S. Fletcher, P. W. Shor, and M. Z. Win, "Channel-Adapted Quantum Error Correction for the Amplitude Damping Channel," *IEEE Transactions on Information Theory*, vol. 54, no. 12, pp. 5705–5718, Dec. 2008, doi.org/10.1109/TIT.2008.2006458.

[120] P. W. Shor, G. Smith, J. A. Smolin, and B. Zeng, "High Performance Single-Error-Correcting Quantum Codes for Amplitude Damping," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 7180–7188, Oct. 2011, doi.org/10.1109/TIT.2011.2165149.

[121] R. Lang and P. W. Shor, "Nonadditive Quantum Error Correcting Codes Adapted to the Ampltitude Damping Channel," Dec. 2007, arxiv.org/abs/0712.2586.

[122] A. Elkelesh, M. Ebada, S. Cammerer, L. Schmalen, and S. Ten Brink, "Decoder-in-the-Loop: Genetic Optimization-Based LDPC Code Design," *IEEE Access*, vol. 7, pp. 141 161–141 170, Sep. 2019, doi.org/10.1109/ACCESS.2019.2942999.