Tailoring surface codes

Improvements in quantum error correction
with biased noise

David Kingsley Tuckett

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

School of Physics
Faculty of Science
The University of Sydney

2020

Abstract

For quantum computers to reach their full potential will require error correction. We study the surface code, one of the most promising quantum error correcting codes, in the context of predominantly dephasing (Z-biased) noise, as found in many quantum architectures.

We find that the surface code is highly resilient to Y-biased noise, and tailor it to Z-biased noise, whilst retaining its practical features. We demonstrate ultrahigh thresholds for the tailored surface code: $\sim 39\%$ with a realistic bias of $\eta = 100$, and $\sim 50\%$ with pure Z noise, far exceeding known thresholds for the standard surface code: $\sim 11\%$ with pure Z noise, and $\sim 19\%$ with depolarizing noise. Furthermore, we provide strong evidence that the threshold of the tailored surface code tracks the hashing bound for all biases.

We reveal the hidden structure of the tailored surface code with pure Z noise that is responsible for these ultrahigh thresholds. As a consequence, we prove that its threshold with pure Z noise is 50%, and we show that its distance to Z errors, and the number of failure modes, can be tuned by modifying its boundary. For codes with appropriately modified boundaries, the distance to Z errors is $O(n)$ compared to $O(\sqrt{n})$ for square codes, where n is the number of physical qubits. We demonstrate that these characteristics yield a significant improvement in logical error rate with pure Z and Z-biased noise.

Finally, we introduce an efficient approach to decoding that exploits code symmetries with respect to a given noise model, and extends readily to the fault-tolerant context, where measurements are unreliable. We use this approach to define a decoder for the tailored surface code with Z-biased noise. Although the decoder is suboptimal, we observe exceptionally high fault-tolerant thresholds of $\sim 5\%$ with bias $\eta = 100$ and exceeding 6% with pure Z noise.

Our results open up many avenues of research and, recent developments in bias-preserving gates, highlight their direct relevance to experiment.

Statement of contribution

This thesis consists of an introduction, three body chapters and a conclusion. I am wholly responsible for the introduction and conclusion. The body chapters are based on research papers with minor formatting and typographical corrections, as well as some additions indicated below. The research papers are all published in peer-reviewed journals. I am the lead author on all of the research papers and my contribution to each paper is stated below.

Chapter 2 Ultrahigh error threshold for surface codes with biased noise
https://doi.org/10.1103/PhysRevLett.120.050505
https://arxiv.org/abs/1708.08474

Journal Physical Review Letters **120**, 050505 (2018)
© 2018 American Physical Society

Authors David K. Tuckett, Stephen D. Bartlett, and Steven T. Flammia

Contribution This chapter contains the full letter with an expanded numerics section to integrate the supplemental material published alongside the letter. The initial idea for the project arose from discussions between all authors. The research was undertaken by me, with supervision from Stephen and Steve. The numerics presented in the paper were performed by me. All authors contributed to drafting and editing the paper.

Chapter 3 Tailoring surface codes for highly biased noise
https://doi.org/10.1103/PhysRevX.9.041031
https://arxiv.org/abs/1812.08186

Journal Physical Review X **9**, 041031 (2019)
🅭 Creative Commons Attribution 4.0 International

Authors David K. Tuckett, Andrew S. Darmawan, Christopher T. Chubb, Sergey Bravyi, Stephen D. Bartlett, Steven T. Flammia

Contribution This chapter contains the full paper, with the addition of Figure 3.11 to illustrate the key structural theorem. The initial idea for the project arose from discussions between Stephen, Steve, Sergey and me. The research was primarily undertaken by me, with supervision from Stephen and Steve. The numerics presented in the paper were performed by me. The analytics presented in the paper were initiated by me and further developed by Sergey, Chris and me. The decoder adaptation, presented in Section 3.6, was initiated by Andrew and further developed by Andrew and me. I drafted the paper, with contributions from Sergey (Section 3.3.2) and Andrew (Section 3.6), and all authors contributed to editing.

Chapter 4 Fault-tolerant thresholds for the surface code in excess of 5% under biased noise
https://doi.org/10.1103/PhysRevLett.124.130501
https://arxiv.org/abs/1907.02554

Journal Physical Review Letters **124**, 130501 (2020)
© 2020 American Physical Society

Authors David K. Tuckett, Stephen D. Bartlett, Steven T. Flammia, Benjamin J. Brown

Contribution This chapter contains the full letter with the addition of the supplemental material, published alongside the letter, detailing the decoder implementation. The initial idea for the project is due to Ben and builds directly on the work presented in Chapter 3. The research was primarily undertaken by me, with supervision from Ben and advice from Stephen and Steve. The numerics presented in the paper were performed by me. The decoder concept was initially proposed by Ben and further developed by me. Ben drafted the paper and all authors contributed to editing. I wrote the additional section detailing the decoder implementation.

This is to certify that, as supervisor for the candidature upon which this thesis is based, I can confirm that the authorship contribution statements above are correct.

Signature:

Date:

Supervisor name: Stephen Bartlett

Statement of originality

This is to certify that to the best of my knowledge, save as expressly acknowledged, the content of this thesis is my own work. This thesis has not previously been submitted for a degree at this or any other university.

Signature:

Date:

Student name: David Kingsley Tuckett

Acknowledgements

I would like to express my sincere thanks to my supervisor Stephen Bartlett and co-supervisor Steve Flammia for their guidance, support, encouragement and especially their enthusiasm.

I am very grateful to my co-authors for many fascinating and fruitful discussions: Ben Brown, Chris Chubb, Andrew Darmawan, Sergey Bravyi, and my supervisors. I would also like to thank everyone in the Sydney Quantum Information Theory group for creating such a friendly, amusing and inspiring workplace.

Finally, I would like to thank my family. I thank my parents and brother for their love and support over the years. Above all, to my wife Yvonne and my daughters Annie and Charlotte: Thank you for absolutely everything, especially for coming on this amazing journey with me.

Contents

1 Introduction

1.1 Overview

Quantum computers have the potential to solve certain problems that could never be solved, in a reasonable time, using classical computers [1]. Large-scale reliable quantum computers could bring significant societal benefits in many areas ranging from the development of new medicines and more energy efficient processes to enhanced machine learning [2, 3]. Recently, Google performed a sampling calculation in about 200 seconds, using an experimental 53-qubit quantum computer, that they estimate would take 10 000 years using a state-of-the-art classical supercomputer [4]. In response to this claim, IBM estimated the experiment could be classically simulated in 2.5 days [5]. Although the chosen problem was rather contrived and the classical simulation estimates vary wildly, the result reveals some of the potential of quantum computers. In a popular review [6] of Google's result, Scott Aaronson points out that quantum computing is in its infancy; referring to how the transistor revolutionized classical computing, he states: "We don't yet have the quantum computing version of the transistor - that would be quantum error correction".

The power of quantum computing relies on features of quantum systems such as superposition, entanglement and interference. For quantum computers to realize their full potential, the fragile quantum systems that they manipulate must be protected from environmental noise. With the theoretical discovery of quantum error correction [7] and the threshold theorem [8–12], this became a realistic, albeit very challenging, possibility. The threshold theorem states that it is possible to perform an arbitrarily long quantum computation provided the physical error rate is below some constant threshold. The threshold, which denotes the physical error rate below which the logical error rate can be made arbitrarily small by increasing the code size, depends on both the noise characteristics of the physical system and the choice of an appropriate quantum error correcting code and decoder. The challenge is, therefore, both to implement physical qubits and gates with very low error rates, and to find and tailor practical codes and decoders to achieve high thresholds with realistic noise models.

The surface code [13, 14], an example of a topological stabilizer code, is one of the most studied quantum error correcting codes [15, 16]. Several characteristics place it among the most promising candidates to realize the first generation of scalable quantum computers [17]. It meets the constraints of many quantum computing architectures, having a two-dimensional layout and requiring only local stabilizer measurements, whilst achieving some of the highest thresholds of any known code [16]. Noise affecting quantum systems can be modeled as a probability distribution over Pauli X, Y and Z operators. The most studied noise models are bit-flip (X) noise, phase-flip (Z) noise, and depolarizing noise, where X, Y and Z errors are equally likely, on single qubits. Studies considering these noise models have demonstrated that the surface code achieves high thresholds in both the code-capacity context [18–23] of reliable measurements and the fault-tolerant context [22–25] of unreliable measurements.

In many quantum architectures, such as certain superconducting qubits [26], quantum dots [27], and trapped ions [28], among others, the noise is biased towards dephasing, meaning that Z errors occur much more frequently than other errors. There has been significant interest in adapting quantum codes to biased noise [26, 29–40]. Of particular interest are approaches that can be applied to topological codes. Some of these approaches have demonstrated a significant increase in threshold [26, 33, 34], at the cost of increasing the overhead by concatenating repetition codes with another, possibly topological code. Others have demonstrated performance improvements with biased noise by modifying the size and shape of stabilizer measurements [35–40].

In this thesis, we reveal inherent features of the standard surface code that make it particularly resilient to Y errors and Y-biased noise; a noise model that, to my knowledge had not previously been studied on the surface code. By applying a small modification, which maintains the size and locality of stabilizer measurements, we tailor the surface code to make it particularly resilient to Z errors. Using the tailored surface code, we demonstrate that ultrahigh code-capacity thresholds can be achieved with experimentally prevalent Z-biased noise. We further show how features of the tailored surface code can be leveraged to increase the distance to Z errors quadratically, with no increase in overhead, and demonstrate a significant reduction in logical error rate with Z-biased noise. Finally, we define an efficient decoder that exploits symmetries of the tailored code and Z-biased noise model to demonstrate exceptionally high thresholds in the fault-tolerant context, which is of direct relevance to experiments.

1.2 Thesis structure

In the remainder of this chapter, we introduce the foundation of technical concepts upon which this thesis is built, along with a consistent notation. Chapters 2, 3 and 4 are self-contained papers, which present our research results. Finally in Chapter 5, we summarize our results and suggest possible extensions for future work.

There is a strong narrative relating the research papers, with each paper building on the results of the preceding paper. This narrative is supported by preambles to the research chapters, which are reproduced here for convenience:

Chapter 2 We consider surface code thresholds in an idealized context with reliable measurements. Previous studies of surface code thresholds had focused on pure X (bit-flip) or Z (phase-flip) noise and depolarizing noise. We discover that the threshold of the surface code with pure Y noise is much higher. Tailoring the code, with a simple modification, we demonstrate that ultrahigh thresholds can be achieved with the dephasing or Z-biased noise that is prevalent in many experimental setups.

Chapter 3 We reveal the structure of the tailored surface code with pure Z noise that is responsible for the ultrahigh thresholds discovered in Chapter 2. Furthermore, we show that, by modifying the code boundary, this structure can be leveraged to tune the distance and number of failure modes of the code with respect to pure Z noise. Using codes with such modified boundaries, we demonstrate a significant reduction in logical failure rate with Z-biased noise.

Chapter 4 We introduce an efficient approach to decoding that exploits symmetries of a code with respect to a dominant noise model. Applying this approach, with the insights from Chapter 3, we define a decoder for the tailored surface code with Z-biased noise. We use this decoder to extend the ultrahigh threshold results of Chapter 2 to the fault-tolerant context, i.e. the regime directly relevant to experiments where measurements are unreliable.

1.3 Quantum information

In this section, we introduce the basic concepts and notation of quantum information used throughout this thesis. The concepts presented here are covered in more depth in Ref. [1] and, from a computer science perspective, in Ref. [41]. We start by describing quantum states, before defining the ubiquitous Pauli operators, and finally discussing the measurement and evolution concepts that are most relevant to quantum error correction.

1.3.1 States

The basic unit of classical information is the *bit*, where a bit can have one of two values, conventionally represented by 0 and 1. In a digital computer, the value of a bit might be physically manifested as a signal voltage that lies in one of two bands. The quantum analogue to the classical bit is the *qubit*, which is short for quantum bit. We start by describing pure states of an isolated qubit, then pure states of an isolated system of qubits, and finally mixed states of qubits that interact with their environment.

Single qubit states

A qubit represents a two-level quantum system. Examples of two-level quantum systems are the horizontal / vertical polarization of a photon, or the spin-up / spin-down states of an electron. The two-levels are conventionally represented as $|0\rangle$ and $|1\rangle$, known as *kets* in Dirac notation [42]. However, unlike classical bits which either take the value 0 or 1, the state of a qubit can be a *superposition* of $|0\rangle$ and $|1\rangle$. An isolated qubit is described as being in a *pure* state, and a general pure qubit state is written as:

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

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$, referred to as the *normalization* condition. Technically the set of vectors $\{e^{i\theta}\,|\psi\rangle\}$, for all θ, defines a one-dimensional subspace or *ray* that represents the state; however, the *global phase*, $e^{i\theta}$, is of no physical consequence and we are free to choose $|\psi\rangle$ to represent the state.

The statespace of a qubit is a two-dimensional complex Hilbert space $\mathcal{H}_1$, i.e. the complex vector space $\mathbb{C}^2$ over $\mathbb{C}$ equipped with an inner product, defined below; in this thesis, Hilbert space always refers to a finite-dimensional complex inner product space. The kets $|0\rangle$ and $|1\rangle$ form an orthonormal basis, known as the computational basis, for the qubit statespace

$$\mathcal{H}_1 = \operatorname{span}\{|0\rangle\,, |1\rangle\}. \tag{1.2}$$

These basis kets are written in vector notation as

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1.3}$$

The state $|\psi\rangle$ can then be written in vector notation as

$$|\psi\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{1.4}$$

The dual vectors to $|0\rangle$ and $|1\rangle$ are written as $\langle 0|$ and $\langle 1|$, known as *bras*, and the dual vector to $|\psi\rangle$ is defined as

$$\langle\psi| = \alpha^* \langle 0| + \beta^* \langle 1|, \tag{1.5}$$

and can be written in vector notation as

$$\langle\psi| = \alpha^* \begin{pmatrix} 1 & 0 \end{pmatrix} + \beta^* \begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha^* & \beta^* \end{pmatrix}. \tag{1.6}$$

The inner product between any two states $|\psi\rangle$ and $|\phi\rangle$ is defined to be $\langle\psi|\phi\rangle = \langle\psi|\,|\phi\rangle$. With this notation, the orthonormality of $|0\rangle$ and $|1\rangle$ is expressed as $1 = \langle 0|0\rangle = \langle 1|1\rangle$, $0 = \langle 0|1\rangle = \langle 1|0\rangle$, and the normality of $|\psi\rangle$, see Eqn. 1.1, is expressed succinctly as $\langle\psi|\psi\rangle = 1$.

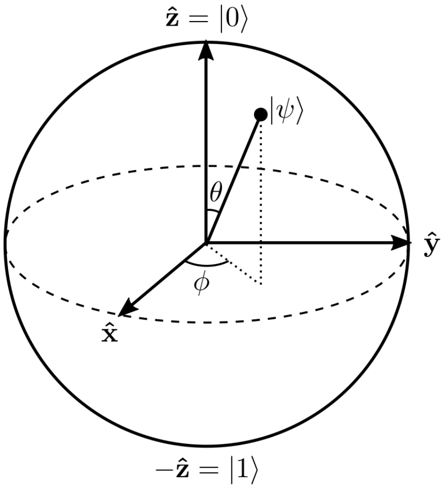


Figure 1.1: Bloch sphere

A general pure qubit state can be represented geometrically as a point on the surface of the Bloch sphere [1], see Fig. 1.1. Since the global phase of a quantum state is of no physical consequence and the state must be normalized, a pure qubit state can be parameterized by two real numbers θ and ϕ as

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{1.7}$$

where $0 \leq \theta \leq \pi/2$ and $0 \leq \phi < 2\pi$. The *relative phase*, $e^{i\phi}$, is of physical consequence, with different values corresponding to different states. The computational basis states $|0\rangle$ and $|1\rangle$ are represented by points at the North pole (positive z axis) and South pole (negative z axis), respectively. It is convenient to introduce the standard notation $|+\rangle$ and $|-\rangle$ for the states corresponding to the surface points on the positive and negative x axes, respectively. These states also form a orthonormal basis for the qubit statespace, and can be written as a superposition of the computational basis states as

$$|+\rangle = \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle \quad \text{and} \quad |-\rangle = \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle. \tag{1.8}$$

Similarly we use the notation $|{+i}\rangle$ and $|{-i}\rangle$ for the states corresponding to the surface points on the positive and negative y axes, respectively, and these, written as a superposition of the computation basis states, are

$$|{+i}\rangle = \tfrac{1}{\sqrt{2}}\,|0\rangle + i\tfrac{1}{\sqrt{2}}\,|1\rangle \quad \text{and} \quad |{-i}\rangle = \tfrac{1}{\sqrt{2}}\,|0\rangle - i\tfrac{1}{\sqrt{2}}\,|1\rangle\,. \tag{1.9}$$

The symmetry of the Bloch sphere makes is clear that any pure qubit state can be expressed as a superposition of basis states. For example, the computational basis states can be written in terms of $|+\rangle$ and $|-\rangle$ as

$$|0\rangle = \tfrac{1}{\sqrt{2}}\,|+\rangle + \tfrac{1}{\sqrt{2}}\,|-\rangle \quad \text{and} \quad |1\rangle = \tfrac{1}{\sqrt{2}}\,|+\rangle - \tfrac{1}{\sqrt{2}}\,|-\rangle\,. \tag{1.10}$$

Multiple qubit states

So far we have described pure states of an isolated qubit. We now describe pure states of an isolated system of multiple qubits. As stated, the statespace of a single qubit is the Hilbert space $\mathcal{H}_1$. We can combine Hilbert spaces to construct a larger Hilbert space using the *tensor product*. In general, if we have two Hilbert spaces $\mathcal{V}$ and $\mathcal{W}$, with orthonormal bases $\{|v\rangle\}$ and $\{|w\rangle\}$, then the tensor product of $\mathcal{V}$ and $\mathcal{W}$, written as $\mathcal{V} \otimes \mathcal{W}$, has an orthonormal basis $\{|v\rangle \otimes |w\rangle\}$, such that $\dim(\mathcal{V} \otimes \mathcal{W}) = \dim \mathcal{V} \times \dim \mathcal{W}$. For arbitrary vectors $|v\rangle, |v'\rangle \in \mathcal{V}$, $|w\rangle, |w'\rangle \in \mathcal{W}$ and scalar $c \in \mathbb{C}$, the tensor product has the following important properties:

$$c(|v\rangle \otimes |w\rangle) = (c\,|v\rangle) \otimes |w\rangle = |v\rangle \otimes (c\,|w\rangle) \tag{1.11}$$

$$(|v\rangle + |v'\rangle) \otimes |w\rangle = |v\rangle \otimes |w\rangle + |v'\rangle \otimes |w\rangle \tag{1.12}$$

$$|v\rangle \otimes (|w\rangle + |w'\rangle) = |v\rangle \otimes |w\rangle + |v\rangle \otimes |w'\rangle\,. \tag{1.13}$$

For a system of n-qubits, the statespace is given by the higher-dimensional Hilbert space $\mathcal{H}_n$, which is the n-fold tensor product of $\mathcal{H}_1$, i.e. $\mathcal{H}_n$ is the complex inner product space $\mathbb{C}^{2^n}$ over $\mathbb{C}$. In terms of the standard computational basis for a system of n-qubits, we have

$$\mathcal{H}_n = \operatorname{span}\{|a_1\, a_2\, \ldots\, a_n\rangle \mid a_j \in \mathbb{Z}_2\} = \operatorname{span}\{|a_1\rangle \otimes |a_2\rangle \otimes \ldots \otimes |a_n\rangle \mid a_j \in \mathbb{Z}_2\}. \tag{1.14}$$

We will often denote the n-qubit Hilbert space by $\mathcal{H}$, leaving the n implicit.

An example should clarify the notation. Consider the arbitrary single qubit states $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ and $|\phi\rangle = \gamma\,|0\rangle + \delta\,|1\rangle$. The tensor product of $|\psi\rangle$ with $|\phi\rangle$ is

$$\begin{aligned} |\psi\rangle \otimes |\phi\rangle &= (\alpha\,|0\rangle + \beta\,|1\rangle) \otimes (\gamma\,|0\rangle + \delta\,|1\rangle) \\ &= \alpha\gamma(|0\rangle \otimes |0\rangle) + \alpha\delta(|0\rangle \otimes |1\rangle) + \beta\gamma(|1\rangle \otimes |0\rangle) + \beta\delta(|1\rangle \otimes |1\rangle) \\ &= \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle \end{aligned} \tag{1.15}$$

The tensor product when applied in vector notation is called the Kronecker product. Writing $|\psi\rangle \otimes |\phi\rangle$ in vector notation, see Eqn. 1.4, gives the definition of the Kronecker product, for a pair of two-dimensional vectors, as

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix}\alpha \\ \beta\end{pmatrix} \otimes \begin{pmatrix}\gamma \\ \delta\end{pmatrix} = \begin{pmatrix}\alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta\end{pmatrix}. \tag{1.16}$$

The two-qubit state $|\psi\rangle \otimes |\phi\rangle$ is an example of a product state, where both qubits are individually in well-defined pure states. Beyond product states, a multiple qubit system can also exist in an *entangled* state. For a system in an entangled state, the system is in a well-defined pure state but individual qubits, or subsystems of qubits, are not in well-defined pure states. An example of such an entangled state, known as a Bell state [1], is

$$\left|\phi^+\right\rangle = \tfrac{1}{\sqrt{2}}\,|00\rangle + \tfrac{1}{\sqrt{2}}\,|11\rangle \tag{1.17}$$

The entangled state $|\phi^+\rangle$ cannot be expressed as the tensor product of separate pure states for the two individual qubits. Entangled states can be defined for any multiple qubit system and, in Section 1.5, we will see that they are an essential ingredient in quantum error correction.

Mixed states

So far we have only considered isolated qubits or systems of qubits. Such isolated systems are described as being in a pure state, which, as we have seen, can be represented by a unit vector, written as a ket. We now consider *open* systems, i.e. systems that interact with their environment. For example, consider a system, subject to noise from the environment, whose state may have been altered with some probability. Without full knowledge of the interaction, we must describe such as system as being in a *mixed* state, i.e. a probabilistic mixture of pure states.

The standard representation for a mixed state is called the *density operator*, which can be expressed as a convex combination of pure states $|\psi_j\rangle$ each with probability p_j, written as

$$\rho = \sum_j p_j\,|\psi_j\rangle\langle\psi_j| \tag{1.18}$$

where $p_j \in \mathbb{R}$, $p_j \geq 0$ and $\sum p_j = 1$, ensuring convexity. The density operator is a sum of outer products of vectors and so can be represented as a matrix; it is interchangeably called the *density matrix*. Defined in this way, the density matrix is positive semi-definite and has trace equal to one, i.e. $\rho \geq 0$ and $\mathrm{tr}(\rho) = 1$.

Geometrically, a mixed state of a single qubit can be represented as a point inside the Bloch sphere, see Fig. 1.1. It is important to note that a mixed state is not the same as a pure superposition state, as is apparent in the measurement statistics, see Section 1.3.3. A pure state can also be represented by a density operator as the trivial sum $\rho = |\psi\rangle\langle\psi|$. We can distinguish between pure and mixed states by noting that $\mathrm{tr}(\rho^2) \leq 1$ holds in general, and equality holds if and only if ρ represents a pure state.

There are different perspectives on mixed states. On the one hand, a mixed state can be prepared as a statistical ensemble of pure states; however, the decomposition of a mixed state into a sum of pure states is not unique, so there is no way to physically distinguish between different preparations of a mixed state represented by a given density operator. On the other hand, if we consider a composite quantum system in an entangled pure state, then the state of a subsystem is in general a mixed state; the so-called *reduced density matrix* representing the subsystem is derived by performing a partial trace over the rest of the composite system. Given the different perspectives, we define the density operator, without reference to statistical ensembles of pure states, as a positive operator with trace equal to one.

Random errors map a computationally useful pure state to a noisy mixed state and, in Section 1.5, we will see how quantum error correction aims to return the mixed state to the error-free pure state, as far as is possible.

1.3.2 Pauli operators

Quantum operators, which can be represented by matrices, play an essential role in describing the measurement and evolution of quantum states. The *Pauli operators* [1] (also called *Pauli matrices*), in particular, are ubiquitous in the field of quantum information. Here we focus on the mathematical properties of the Pauli operators as a reference for subsequent subsections on measurement and evolution, see Sections 1.3.3 and 1.3.4.

It is useful to first define the single-qubit *identity operator* (sometimes called the zeroth Pauli operator)

$$I = |0\rangle\langle 0| + |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{1.19}$$

The identity operator has the important property that any non-zero vector is an eigenvector with eigenvalue 1, such that its action on any state is trivial.

The single-qubit Pauli operators in Dirac and matrix notation are

$$X = |0\rangle\langle 1| + |1\rangle\langle 0| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1.20}$$

$$Y = -i\,|0\rangle\langle 1| + i\,|1\rangle\langle 0| = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{1.21}$$

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1.22}$$

Each of the Pauli operators has eigenvalues +1 and −1. The corresponding eigenvectors are $|+\rangle$ and $|-\rangle$ for X, $|{+i}\rangle$ and $|{-i}\rangle$ for Y, and $|0\rangle$ and $|1\rangle$ for Z, as defined in Eqns 1.8, 1.9 and 1.3. Therefore, the spectral decompositions of the Pauli operators are

$$X = |+\rangle\langle +| - |-\rangle\langle -| \tag{1.23}$$

$$Y = |{+i}\rangle\langle {+i}| - |{-i}\rangle\langle {-i}| \tag{1.24}$$

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1|\,. \tag{1.25}$$

The Pauli operators are Hermitian, unitary, and mutually anticommute

$$X = X^\dagger, \quad Y = Y^\dagger, \quad Z = Z^\dagger \tag{1.26}$$

$$XX^\dagger = YY^\dagger = ZZ^\dagger = I \tag{1.27}$$

$$XY = -YX, \quad YZ = -ZY, \quad ZX = -XZ \tag{1.28}$$

and are related as

$$X = -iYZ, \quad Y = -iZX, \quad Z = -iXY. \tag{1.29}$$

The Pauli operators can be generalized to n-qubits as the n-fold tensor product of single-qubit Pauli operators. The n-qubit Pauli group is defined as

$$\mathcal{P}_n = \{cf_1 \otimes f_2 \otimes \ldots \otimes f_n \mid f_j \in \{I, X, Y, Z\}, c \in \{\pm 1, \pm i\}\}\,. \tag{1.30}$$

We will often denote the n-qubit Pauli group by $\mathcal{P}$, leaving the n implicit. The Pauli group $\mathcal{P}$ provides a basis for linear operators on the n-qubit Hilbert space $\mathcal{H}$. In Section 1.5, we will see that errors on a quantum system can be modeled as the action of elements from this group, and that quantum error correcting codes can be defined by measurements described by elements from this group.

1.3.3 Measurement

Measurement of a quantum system does not in general reveal a pre-existing property of the system but rather induces the system to make a random transition, with probability determined by the pre-measurement state and choice of measurement, to a post-measurement state that is consistent with the measurement. We start by briefly introducing general measurements before looking at the special case of projective measurements and finally stabilizer measurements, which are used extensively in quantum error correcting codes.

General measurements

A *general measurement* is described by a collection of measurement operators $\{M_m\}$ where there is one operator M_m for each possible outcome m. For a system in a pure state $|\psi\rangle$, the probability of a given outcome m is

$$p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle\,, \tag{1.31}$$

and the post-measurement state of the system is

$$\frac{M_m |\psi\rangle}{\sqrt{p(m)}}. \tag{1.32}$$

The following completeness condition on $\{M_m\}$ ensures that the probabilities sum to 1:

$$\sum_m M_m^\dagger M_m = I. \tag{1.33}$$

For a system in a mixed state ρ, Eqn. 1.31, giving the probability of outcome m, becomes

$$p(m) = \mathrm{tr}(M_m^\dagger M_m \rho), \tag{1.34}$$

and Eqn. 1.32, giving the post-measurement state, becomes

$$\frac{M_m \rho M_m^\dagger}{\sqrt{p(m)}}. \tag{1.35}$$

Projective measurements

Projective measurements are a special case of general measurements. In this special case, we require that the measurement operators $\{M_m\}$ be orthogonal projectors, i.e. writing Π_λ in place of M_m, we require that $\Pi_\lambda = \Pi_\lambda^\dagger$ and $\Pi_j \Pi_k = \delta_{j,k}\Pi_j$. The projective measurement is then described by a Hermitian operator M called an *observable* with spectral decomposition

$$M = \sum_\lambda \lambda \Pi_\lambda \tag{1.36}$$

where Π_λ projects onto the eigenstate (or eigenspace for degenerate eigenvalues) of M with eigenvalue λ. For a system in a pure state $|\psi\rangle$, the probability of a given outcome λ is

$$p(\lambda) = \langle\psi| \Pi_\lambda |\psi\rangle \tag{1.37}$$

and the post-measurement state of the system is

$$\frac{\Pi_\lambda \left|\psi\right\rangle}{\sqrt{p(\lambda)}}. \tag{1.38}$$

This statement of outcome probability and post-measurement state is known as the Born rule [43].

We illustrate the Born rule for the simple case of a projective measurement on a single qubit in the computational basis. Recall the general pure qubit state, Eqn. 1.1, in Dirac and vector notation

$$\left|\psi\right\rangle = \alpha \left|0\right\rangle + \beta \left|1\right\rangle = \begin{pmatrix}\alpha \\ \beta\end{pmatrix} \tag{1.39}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. The coefficients α and β are referred to as *probability amplitudes*, for reasons that will become clear. The observable associated with measurement in the computational basis, where the possible post-measurement states are $\left|0\right\rangle$ and $\left|1\right\rangle$, is the Pauli Z operator, see Eqn. 1.22. Recall that the Z operator has eigenvalues $+1$ and -1 and corresponding eigenstates $\left|0\right\rangle$ and $\left|1\right\rangle$, i.e. $Z\left|0\right\rangle = \left|0\right\rangle$ and $Z\left|1\right\rangle = -\left|1\right\rangle$. Since the eigenvalues are non-degenerate, the projection operators are simply $\Pi_{+1} = \left|0\right\rangle\!\left\langle 0\right|$ and $\Pi_{-1} = \left|1\right\rangle\!\left\langle 1\right|$ for eigenvalues $+1$ and -1, respectively. The Born rule then gives the probabilities of outcome eigenvalues $+1$ and -1 as

$$p(+1) = \left\langle\psi\right| \Pi_{+1} \left|\psi\right\rangle = \left\langle\psi|0\right\rangle \left\langle 0|\psi\right\rangle = |\left\langle 0|\psi\right\rangle|^2 = |\begin{pmatrix}1 & 0\end{pmatrix} \begin{pmatrix}\alpha \\ \beta\end{pmatrix}|^2 = |\alpha|^2 \tag{1.40}$$

and

$$p(-1) = \left\langle\psi\right| \Pi_{-1} \left|\psi\right\rangle = \left\langle\psi|1\right\rangle \left\langle 1|\psi\right\rangle = |\left\langle 1|\psi\right\rangle|^2 = |\begin{pmatrix}0 & 1\end{pmatrix} \begin{pmatrix}\alpha \\ \beta\end{pmatrix}|^2 = |\beta|^2, \tag{1.41}$$

with post-measurement states $\left|0\right\rangle$ and $\left|1\right\rangle$, respectively. This example illustrates projective measurement in the computational basis, also known as the Z basis. Similar measurements can be made in any basis. In this thesis, we also consider measurements in the X and Y bases as described by the Pauli X and Y operators, respectively.

Stabilizer measurements

The quantum error correcting codes discussed in thesis make extensive use of binary projective measurements onto subspaces, called *stabilizer measurements*. Stabilizer measurements are covered by the formalism of projective measurements given in the previous subsection. We illustrate them here with an example.

Consider the four-dimensional Hilbert space $\mathcal{H}_2$ of a two-qubit system

$$\mathcal{H}_2 = \operatorname{span}\{\left|00\right\rangle, \left|01\right\rangle, \left|10\right\rangle, \left|11\right\rangle\}. \tag{1.42}$$

We can define two orthogonal subspaces of $\mathcal{H}_2$ as

$$\mathcal{V}_0 = \operatorname{span}\{\left|00\right\rangle, \left|11\right\rangle\} \quad \text{and} \quad \mathcal{V}_1 = \operatorname{span}\{\left|01\right\rangle, \left|10\right\rangle\}. \tag{1.43}$$

The observable $ZZ = Z \otimes Z$ has eigenspaces $\mathcal{V}_0$ and $\mathcal{V}_1$ with eigenvalues $+1$ and -1, respectively. To see this, consider a general state $\left|\psi\right\rangle$ in the subspace $\mathcal{V}_0$

$$\left|\psi\right\rangle = \alpha \left|00\right\rangle + \beta \left|11\right\rangle \tag{1.44}$$

then we have

$$ZZ\,|\psi\rangle = \alpha(Z\,|0\rangle \otimes Z\,|0\rangle) + \beta(Z\,|1\rangle \otimes Z\,|1\rangle) = \alpha\,|00\rangle + \beta\,|11\rangle = |\psi\rangle\,. \tag{1.45}$$

Conversely, consider a general state $|\phi\rangle$ in the subspace $\mathcal{V}_1$

$$|\phi\rangle = \gamma\,|01\rangle + \delta\,|10\rangle \tag{1.46}$$

then we have

$$ZZ\,|\phi\rangle = \gamma(Z\,|0\rangle \otimes Z\,|1\rangle) + \delta(Z\,|1\rangle \otimes Z\,|0\rangle) = -\gamma\,|01\rangle - \delta\,|10\rangle = -\,|\phi\rangle\,. \tag{1.47}$$

In summary, ZZ is written in terms of orthogonal projectors, see Eqn. 1.36, as

$$ZZ = +\mathit{\Pi}_+ - \mathit{\Pi}_- = +(|00\rangle\langle 00| + |11\rangle\langle 11|) - (|01\rangle\langle 01| + |10\rangle\langle 10|), \tag{1.48}$$

and measuring ZZ projects a state in $\mathcal{H}_2$ onto one of the subspaces $\mathcal{V}_0$ or $\mathcal{V}_1$ with measurement outcome $+1$ or -1, respectively, allowing us to determine which subspace.

With this example, we can already see how to construct a basic error detection code against single bit-flip errors, i.e. an error that flips a qubit between 0 and 1, see Section 1.3.4. The idea is that we encode a general single-qubit state, $\alpha\,|0\rangle + \beta\,|1\rangle$ into two qubits as $|\psi\rangle$, given by Eqn. 1.44, in the $\mathcal{V}_0$ subspace. Suppose $|\psi\rangle$ suffers a single bit-flip error with some probability. Such an error would rotate $|\psi\rangle$ into the the $\mathcal{V}_1$ subspace. Therefore, by performing a ZZ measurement, we can detect whether such an error has occurred. Crucially, if an error has not occurred, the logical state $|\psi\rangle$ is unaffected by the measurement; the coefficients α and β are preserved and we say the state is stabilized. Although this encoding allows us to detect such an error, it does not give us enough information to know how to correct the error. In Section 1.5, we will see how to construct codes that correct any single-qubit error.

1.3.4 Evolution

In order to perform useful quantum computation, or even maintain an error-free state using quantum error correction, we need to manipulate or evolve the quantum system representing our information. We start by describing the evolution of an isolated system, known as unitary evolution, before describing the evolution of open systems such as those subject to noise from their environment.

Unitary evolution

The evolution of an isolated system, represented by a pure state, is described by *unitary* operators. A unitary operator U satisfies the relation

$$UU^\dagger = U^\dagger U = I. \tag{1.49}$$

We denote the group of unitary operators by $\mathcal{U}$. The evolution of a pure state $|\psi\rangle$ by $U \in \mathcal{U}$ is written $U\,|\psi\rangle$. Similarly, the unitary evolution of a mixed state ρ is written $U\rho U^\dagger$. Unitary operators are also often referred to as *gates* by analogy to classical logic gates.

The Pauli operators, see Section 1.3.2, are important examples of single-qubit unitary operators or gates. Referring back to the Bloch sphere, see Fig. 1.1, representation of a single-qubit state, the effect of the Pauli X, Y, and Z operators is to rotate the qubit π radians around the x, y and z axes, respectively. In terms of the computational basis states, the effect is

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle \tag{1.50}$$
$$Y|0\rangle = i|1\rangle, \quad Y|1\rangle = -i|0\rangle \tag{1.51}$$
$$Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle. \tag{1.52}$$

We say that X induces a *bit-flip*, Z induces a *phase-flip*, and Y is equivalent to applying Z then X with a phase of i.

Another important group of operators or gates belong to the *Clifford* group. The Clifford group is defined as the normalizer of the Pauli group $\mathcal{P}$, see Eqn. 1.30

$$\mathcal{C} = \{U \in \mathcal{U} \mid UPU^\dagger \in \mathcal{P} \;\forall\; P \in \mathcal{P}\}. \tag{1.53}$$

In this thesis, Clifford group always refers to this definition, although Clifford groups actually form a hierarchy of which this is the second level and the Pauli group is the first. The size of the Clifford group grows quickly in the number of qubits but fortunately it is generated by just three operators: the Hadamard gate H, the phase gate S, and the two-qubit controlled-not gate C_X

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}, \quad S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}, \quad C_X = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}. \tag{1.54}$$

The Hadamard gate is Hermitian and conjugates X to Z and vice versa, i.e. $HXH = Z$ and $HZH = X$. The phase gate is the square-root of Z, i.e. $S^2 = Z$. The controlled-not gate applies the identity I or X to the second qubit if the first qubit is in the state $|0\rangle$ or $|1\rangle$, respectively, i.e. $C_X = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$.

An interesting result, known as the Gottesman-Knill theorem [44], is that quantum circuits consisting of only preparation and measurement in the computational basis together with Clifford gates can be simulated efficiently on a classical computer. Nevertheless, such quantum circuits are of particular interest to us since they are sufficient for implementing quantum error correction, as we will see in Section 1.5. For completeness, we note that in order to achieve universal quantum computation, the Clifford gates must be supplemented with one non-Clifford gate such as the T-gate

$$T = \begin{pmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{pmatrix}, \tag{1.55}$$

which is the square-root of S, i.e. $T^2 = S$. One of the criteria in selecting a practical quantum error correcting code is the ease with which a universal gate set can be implemented on the encoded qubit.

Error channels

The evolution of an open system that interacts with its environment, described by a mixed state, is not in general unitary. It is always possible to describe the evolution by a unitary operation on the combined Hilbert space of the open system and an ancillary system; this is known as *purification* and is described by the Stinespring dilation theorem [45]. However, it is often simpler to consider the evolution of the open system directly. This evolution is described as a channel and represented by a linear map $\mathcal{E}$ between operators. We recall, from Section 1.3.1, that an open system is represented by a density matrix ρ, which is a positive operator with trace equal to one. A map from density operators to density operators is completely-positive (CP), i.e. it preserves positivity $\mathcal{E}(\rho) \geq 0$, and trace-preserving (TP), i.e. $\operatorname{tr}(\mathcal{E}(\rho)) = \operatorname{tr}(\rho)$. (In this thesis we do not consider trace-decreasing maps that represent error processes involving qubit loss; such errors are, in any case, generally easier to correct since the location of the loss is usually known.) The error channels considered in this thesis can therefore be represented by CPTP maps.

There are several equivalent representations of CPTP maps. We use the *operator-sum representation* [46] in this thesis. In the operator-sum representation, a CP map $\mathcal{E}$ can be written as

$$\mathcal{E}(\rho) = \sum_j K_j \rho K_j^\dagger. \tag{1.56}$$

Additionally we require that the K_j, known as *Kraus operators*, satisfy the completeness relation

$$\sum_j K_j^\dagger K_j = 1 \tag{1.57}$$

to ensure trace-preservation.

In this thesis, we consider a range of noise models, including the most commonly studied error models: bit-flip, phase-flip and depolarizing noise. In addition, we consider biased-depolarizing noise, where rotations about one axis occur much more frequently than about other axes, see Fig. 1.1. As stated in Section 1.1, dephasing noise, rotation about the z axis, is dominant in many quantum architectures. We make the simplifying assumption that errors are applied to each physical qubit according to the same probability distribution, known as an *independent and identically-distributed* (i.i.d.) noise model. We now give definitions of the various single-qubit error channels discussed in this thesis. In each case, the initial state of the qubit is represented by ρ. The qubit state is modified with some probability p and unchanged with probability $(1 - p)$.

The *bit-flip channel* applies a Pauli X operator, i.e. a π radian rotation about the x axis (see Fig. 1.1), with probability p, and is defined as

$$\mathcal{E}(\rho) = (1 - p)\rho + X\rho X. \tag{1.58}$$

The *phase-flip channel* applies a Pauli Z operator, i.e. a π radian rotation about the z axis, with probability p, and is defined as

$$\mathcal{E}(\rho) = (1 - p)\rho + Z\rho Z. \tag{1.59}$$

The *depolarizing channel* applies a Pauli X, Y or Z operator each with probability $p/3$, and is defined as

$$\mathcal{E}(\rho) = (1 - p)\rho + \frac{p}{3}\left(X\rho X + Y\rho Y + Z\rho Z\right). \tag{1.60}$$

This definition, in terms of Pauli operators, differs slightly from the formal definition $\mathcal{E}(\rho) = pI/2 + (1 - p)\rho$. However it is so commonly used in simulations of quantum error correction that we use it here for ease of comparison with prior results.

The *biased-depolarizing channel* is a variation on the depolarizing channel where rotations about one axis dominate. We model it by introducing a bias towards one of the Pauli operators.

Let p_X, p_Y and p_Z be the probabilities for Pauli X, Y and Z errors, respectively, such that $p = p_X + p_Y + p_Z$ is the probability of any single-qubit error. We define the bias towards Pauli Z errors as

$$\eta = \frac{p_Z}{p_X + p_Y}, \tag{1.61}$$

and we make the simplifying assumption that $p_X = p_Y$, such that

$$p_Z = \frac{\eta}{\eta + 1} p \quad \text{and} \quad p_X = p_Y = \frac{1}{2(\eta + 1)} p. \tag{1.62}$$

The biased-depolarizing channel is then defined as

$$\mathcal{E}(\rho) = (1 - p)\rho + \frac{1}{2(\eta + 1)} p \left(X\rho X + Y\rho Y \right) + \frac{\eta}{\eta + 1} p Z\rho Z, \tag{1.63}$$

such that with $\eta = 1/2$ it corresponds to the standard depolarizing channel, see Eqn. 1.60, and in the limit $\eta = \infty$ it corresponds to the phase-flip channel, see Eqn. 1.59. This definition is used in Chapters 2 and 4; in Chapter 3 we also consider bias towards Pauli Y errors, which is defined analogously. We note that this is a common definition of biased-depolarizing noise, although it is not unique [26, 32, 33, 36, 37, 39, 47].

1.4 Classical error correction

In this section, we introduce the concepts of classical error correction that are relevant to this thesis. The concepts presented here are covered in more depth in Ref. [48]. We start by discussing the repetition code, the simplest classical code, allowing us to introduce some concepts that have analogues in quantum error correction. We finish by presenting the cycle code, which, as we show in Chapter 3, forms part of the structure of the quantum surface code in the presence of infinitely-biased noise.

1.4.1 Classical repetition code

The aim of classical error correction is to implement a reliable *logical* system to store or send information using unreliable *physical* systems. The simplest classical error correction code that protects a bit of information against a single-bit flip error, i.e. an error that flips 0 to 1 and vice versa, is the three-bit *repetition code*. The three-bit repetition code encodes a *logical bit* into three *physical bits* by simply copying the bit value. The logical $\overline{0}$ and $\overline{1}$ states are encoded into *codewords* as

$$\overline{0} = 000 \quad \text{and} \quad \overline{1} = 111, \tag{1.64}$$

respectively. If a single bit-flip error occurs on one of the physical bits, then we can correct the error by measuring each physical bit and taking a majority vote. Clearly if two bit-flip errors occur, then we can detect the error but we cannot correct it with confidence, and if three bit-flip errors occur, then we cannot even detect the error.

We can easily see that the three-bit repetition code reduces the probability of an uncorrectable error. Suppose we have a binary symmetric channel, i.e. the probability of a bit-flip error occurring on any single bit is p and the probability of no error occurring on any single bit is $(1 - p)$. The *physical error probability* on an unencoded bit is then p. However, since a single bit-flip error can be corrected on a logical bit, the *logical error probability* on the encoded bit is

$\overline{p} = 3(1-p)p^2 + p^3$, which reflects the three ways in which two bit-flips can occur and the one way in which three bit-flips can occur. Therefore, provided $p < 1/2$, the three-bit repetition code reduces the probability of an uncorrectable error.

The repetition code is an example of a *binary linear code*. We say that $\mathcal{C}$ is a $[n,k]$ binary linear code, if it is a subspace $\mathcal{C} \subseteq \mathbb{Z}_2^n$ over $\mathbb{Z}_2$ of dimension k. That is $\mathcal{C}$ is a group of length-n binary vectors (codewords), generated by k linearly independent codewords under bitwise addition and multiplication by 0 or 1 As such, $\mathcal{C}$ encodes k logical bits into n physical bits. For a binary linear code $\mathcal{C}$, we define two important matrices. The *generator matrix* G is a $k \times n$ binary matrix whose rows are a minimal basis of codewords for $\mathcal{C}$. The generator matrix provides a way to encode a length-k binary vector x into a length-n binary vector as $\overline{x} = xG$. The *check matrix* H is a $(n-k) \times n$ binary matrix whose rows are *parity checks* for $\mathcal{C}$, or equivalently a minimal basis of codewords for the code $\mathcal{C}^\perp$ dual to $\mathcal{C}$, where $\mathcal{C}^\perp = \{x \in \mathbb{Z}_2^n \mid x \cdot c \; \forall \; c \in \mathcal{C}\}$. The check matrix provides a way to extract a *syndrome* that gives information about error location on a length-n binary vector x as $s = Hx^T$.

As an example, the three-bit repetition code is a $[3,1]$ code. A generator matrix for the three-bit repetition code is

$$G = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \tag{1.65}$$

and we encode 0 and 1 as

$$\overline{0} = 0 \cdot G = 0 \cdot \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = (000) \quad \text{and} \quad \overline{1} = 1 \cdot G = 1 \cdot \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = (111). \tag{1.66}$$

A check matrix for the three-bit repetition code is

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \tag{1.67}$$

and if $x = (001)$ is the result of a codeword suffering a single bit-flip error, then the syndrome is

$$s = Hx^T = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \end{pmatrix}, \tag{1.68}$$

indicating that the single bit-flip error occurred on the last physical bit.

Like codewords, errors can be represented as binary vectors, where 1 indicates a bit-flip is applied at that location of a codeword. The *weight* of a codeword or an error is the number non-zero bits. The *distance* of a linear code is the weight of the minimum-weight non-zero codeword. If we want to change the information encoded in a code without decoding, then a *logical operator* must be applied to convert one codeword to another. The distance of a linear code is also the weight of the minimum-weight logical operator. A code of distance d can detect up to $(d-1)$ errors and correct up to $(d-1)/2$ errors. An $[n,k]$ code with distance d is denoted as an $[n,k,d]$ code; for example, the three-bit repetition code has a single logical operator (111) and therefore has distance $d = 3$, and we say it is a $[3,1,3]$ code.

The three-bit repetition code is a member of the family of n-bit repetition codes, defined by encoding a logical bit into n physical bits in exactly the same way. For any family of codes, an important figure of merit is the critical error rate p_c, known as the *error threshold*, below which the probability of a logical error can be made arbitrarily small by increasing the code size. In general the error threshold depends on both the choice of code and the given error channel; for example, n-bit repetition codes with bit-flip errors have a threshold of $p_c = 1/2$.

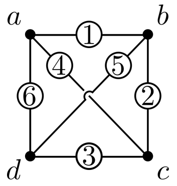
Another figure of merit for a family of codes is the *rate* of the code. For an individual code the rate is k/n and for a family of codes the rate is $\lim_{n\to\infty} k/n$; for example, n-bit repetition codes have a rate of $\lim_{n\to\infty} 1/n = 0$.

1.4.2 Classical cycle code

The *cycle code* features in Chapter 3 where it is formally but concisely defined, see Section 3.3.2. Here we give a more descriptive definition of the cycle code and provide an illustrative example using the concepts and terminology introduced in the previous subsection.

The cycle code $\mathcal{C}_m$ encodes $(m-1)$ logical bits into e physical bits. Consider a complete graph with m vertices and $e = m(m-1)/2$ edges, where $m \geq 3$, and associate binary vectors $x \in \mathbb{Z}_2^e$ with that graph such that each bit of x is associated with an edge. The codewords of $\mathcal{C}_m$ are generated by the set of binary vectors with 1 for edges incident on a vertex, and 0 elsewhere; there are $(m-1)$ linearly independent codewords. The parity checks of $\mathcal{C}_m$ are the binary vectors with 1 for cycles (triangles) of edges, and 0 elsewhere; there are $(e-m+1)$ linearly independent parity checks.

As an example, consider the cycle code $\mathcal{C}_4$ that encodes $(m-1) = 3$ logical bits into $e = 6$ physical bits, i.e. $\mathcal{C}_4$ is a $[6, 3]$ code. The complete graph associated with $\mathcal{C}_4$, labeling the four vertices a, b, c, and d and the six edges 1 to 6, is



Using the edge labeling to index the bits of the binary vectors in $\mathbb{Z}_2^6$, a generating set of codewords corresponding to the vertices a, b, and c gives the rows of a generator matrix for $\mathcal{C}_4$

$$G = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}. \tag{1.69}$$

(The codeword corresponding to vertex d is redundant, being the bitwise sum of the rows of G.) A complete set of linearly independent parity checks, corresponding to the cycles of edges 156, 124, and 235, gives the rows of a check matrix for $\mathcal{C}_4$

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}. \tag{1.70}$$

(The parity check corresponding to the cycle of edges 346 is redundant, being the bitwise sum of the rows of H.)

We show in Chapter 3 that the quantum surface code subject to infinitely-biased noise is equivalent to a cycle code concatenated with several repetition codes. Code concatenation means replacing physical bits in one code with logical bits of another code. Considering the interplay between the cycle code and repetition codes in this concatenation provides insights into the performance of the surface code with highly-biased noise.

1.5 Quantum error correction

In this section, we introduce the concepts of quantum error correction that are relevant to this thesis. The concepts presented here are covered in more depth in Ref. [49] and the references given in the subsections.

The aim of quantum error correction is to implement reliable logical qubits using unreliable physical qubits; (higher-dimensional quantum systems may also be used but we restrict ourselves to qubits in this thesis). As mentioned in Section 1.1, quantum computations rely on superposition, and other features of quantum systems such as entanglement and interference, in order to outperform classical computations for certain problems. This means that quantum error correction should preserve the superposition state of logical qubits, and implement measurement and evolution of logical qubits without spreading errors uncontrollably, such that noiseless quantum computations can be arbitrarily well approximated.

We start by looking at challenges and solutions to preserving the superposition state of a logical qubit, covering the fundamentals of quantum error correction. We then introduce the stabilizer formalism, which can be used to efficiently describe the quantum error correcting codes considered in this thesis. Next we look at how measurement and evolution of logical qubits can be implemented to achieve fault-tolerance before outlining some criteria for evaluating codes. Finally, we introduce topological codes and, in particular, the surface code, which is the main subject of this thesis.

1.5.1 Fundamentals

We saw in Section 1.4, how classical error correction can protect a bit state, either 0 or 1, from random bit-flips. The equivalent task in quantum error correction is to protect the general pure qubit state, $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, from any continuous rotation or even entanglement with its environment. This raises a number of challenges specific to quantum error correction.

Challenges

There are several barriers to quantum error correction that prevent us from naively applying the techniques of classical error correction:

- No-cloning theorem prevents copying.
- Measurement destroys superposition.
- Qubit statespace is continuous.

The no-cloning theorem [50–52] prevents copying of arbitrary unknown quantum states; how can redundancy be used to protect qubits? Measurement destroys superposition; how can we look at the qubits to see what errors have occurred? The qubit statespace is continuous; how can we distinguish a state with a small error from an equally valid nearby state?

With the bit-flip code, we will see how entangled states, incorporating redundancy, and stabilizer measurements can be used to protect the superposition of interest against single bit-flip errors, without cloning unknown states. With the Shor code, we will see how this protection can be extended to handle single phase-flip errors as well. With error discretization, we will see that we only need consider two discrete types of error in order to protect against continuous rotations and entanglement with the environment, and therefore the Shor code protects against all single qubit errors. Finally, we will state and briefly discuss the conditions under which a quantum error correcting code corrects a set of errors.

Bit-flip code

Consider the general pure qubit state (recall Eqn. 1.1)

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1.71}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. Preserving the superposition of this state means preserving the coefficients α and β.

The *bit-flip code* [1], which protects against a single bit-flip error, does not copy the unknown state $|\psi\rangle$, which would violate the no-cloning theorem, but rather encodes $|\psi\rangle$ in the three-qubit logical state $\left|\overline{\psi}\right\rangle$ as

$$\left|\overline{\psi}\right\rangle = \alpha\,|000\rangle + \beta\,|111\rangle\,, \tag{1.72}$$

such that $\left|\overline{\psi}\right\rangle$ is the superposition of two logical codewords $\left|\overline{0}\right\rangle = |000\rangle$ and $\left|\overline{1}\right\rangle = |111\rangle$. This is an extension of the two-qubit bit-flip detection code described at the end of Section 1.3.3. Consider the eight-dimensional Hilbert space $\mathcal{H}_3 = \text{span}\{|a_1\,a_2\,a_3\rangle \mid a_j \in \mathbb{Z}_2\}$ partitioned into four orthogonal subspaces

$$\mathcal{V}_{00} = \text{span}\{|000\rangle\,, |111\rangle\} \tag{1.73}$$
$$\mathcal{V}_{01} = \text{span}\{|001\rangle\,, |110\rangle\} \tag{1.74}$$
$$\mathcal{V}_{11} = \text{span}\{|010\rangle\,, |101\rangle\} \tag{1.75}$$
$$\mathcal{V}_{10} = \text{span}\{|100\rangle\,, |011\rangle\}. \tag{1.76}$$

The logical state $\left|\overline{\psi}\right\rangle$ is in the subspace $\mathcal{V}_{00}$, which we call the *codespace*. Any single bit-flip error rotates the logical state out of the codespace and into one of the other subspaces; for example, a bit-flip error on the last qubit, $IIX = I \otimes I \otimes X$, rotates the logical state into $\mathcal{V}_{01}$

$$IIX\left|\overline{\psi}\right\rangle = IIX(\alpha\,|000\rangle + \beta\,|111\rangle) = \alpha\,|001\rangle + \beta\,|110\rangle \in \mathcal{V}_{01}. \tag{1.77}$$

We can use the stabilizer measurements ZZI and IZZ to determine into which subspace the logical state has rotated. In particular, if we define a syndrome vector $s \in \mathbb{Z}_2^2$, for the observables $M_1 = ZZI$ and $M_2 = IZZ$, as

$$s = (s_1\,s_2) \text{ such that } s_j = \begin{cases} 0, & \text{if measurement of } M_j \text{ yields } +1 \\ 1, & \text{otherwise} \end{cases}, \tag{1.78}$$

then the syndrome $s = (a\,b)$ indicates that the logical state has been rotated to the subspace $\mathcal{V}_{ab}$; for example a bit-flip error on the last qubit rotates the logical state into $\mathcal{V}_{01}$ as revealed by the syndrome $s = (0\,1)$. Assuming no more than one single bit-flip error has occurred, we know from the syndrome which qubit has flipped and we can recover the error-free logical state $\left|\overline{\psi}\right\rangle$ by applying an appropriate operator; for example if the syndrome is $s = (0\,1)$ then we apply the *recovery* operator IIX to recover $\left|\overline{\psi}\right\rangle$. Crucially, throughout the error correction process, the coefficients α and β are preserved, and hence the superposition of interest is preserved.

In anticipation of the stabilizer formalism, defined in Section 1.5.2, we briefly introduce the notion of quantum logical operators. The minimum-weight logical bit-flip operator for the bit-flip code is given by $\overline{X} = XXX$, i.e. $\overline{X}\left|\overline{0}\right\rangle = \left|\overline{1}\right\rangle$ and $\overline{X}\left|\overline{1}\right\rangle = \left|\overline{0}\right\rangle$. A minimum-weight logical phase-flip operator for the bit-flip code is given by $\overline{Z} = IIZ$, i.e. $\overline{Z}\left|\overline{0}\right\rangle = \left|\overline{0}\right\rangle$ and $\overline{Z}\left|\overline{1}\right\rangle = -\left|\overline{1}\right\rangle$. These logical operators keep the logical state in the codespace $\mathcal{V}_{00}$, i.e. $\overline{X}\left|\overline{\psi}\right\rangle \in \mathcal{V}_{00}$ and $\overline{Z}\left|\overline{\psi}\right\rangle \in \mathcal{V}_{00}$. The weight of such an n-qubit Pauli operator is the number of qubits it acts upon non-trivially.

We have seen that the bit-flip code is similar to the classical three-bit repetition code, see Section 1.4, in that it protects against any single bit-flip error. However, in addition to bit-flip errors, qubits are also susceptible to phase-flip errors. The fact that $\overline{Z}$ consists of a single Pauli Z operator reveals that the bit-flip code does not protect against single phase-flip errors. Fortunately, a phase-flip code can be defined by direct analogy to the bit-flip code and the two can be combined to form the Shor code [7] that corrects both single bit-flip and single phase-flip errors.

Shor code

The effect of a bit-flip error, X, is to flip the eigenvectors, $|0\rangle$ and $|1\rangle$, of the Pauli Z operator, i.e. $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. This is mirrored by the effect of a phase-flip error, Z, on the eigenvectors, $|+\rangle$ and $|-\rangle$, of the X operator, i.e. $Z|+\rangle = |-\rangle$ and $Z|-\rangle = |+\rangle$. Therefore the construction of the *phase-flip code*, which protects against a single phase-flip error, mirrors that of the bit-flip code. The phase-flip code encodes the general pure qubit state, see Eqn. 1.71, in the three-qubit logical state $\left|\overline{\psi}\right\rangle$ as

$$\left|\overline{\psi}\right\rangle = \alpha\,|{+}{+}{+}\rangle + \beta\,|{-}{-}{-}\rangle \tag{1.79}$$

such that $\left|\overline{\psi}\right\rangle$ is the superposition of two logical codewords $\left|\overline{0}\right\rangle = |{+}{+}{+}\rangle$ and $\left|\overline{1}\right\rangle = |{-}{-}{-}\rangle$. Using stabilizer measurements XXI and IXX, the analysis of the bit-flip code, above, carries over to show that the phase-flip code protects against a single phase-flip error. We note that the logical codewords of the phase-flip code expressed in the computational basis are

$$\left|\overline{0}\right\rangle = |{+}{+}{+}\rangle = \frac{1}{\sqrt{8}}(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \tag{1.80}$$

$$\left|\overline{1}\right\rangle = |{-}{-}{-}\rangle = \frac{1}{\sqrt{8}}(|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle). \tag{1.81}$$

The *Shor code* [7] concatenates three copies of the bit-flip code, at the bottom level, into a phase flip code, at the top level, to give a nine-qubit code that protects against any single bit-flip or any single phase-flip error, or combination of the two. Concatenation of quantum codes is analogous to concatenation of classical codes in that it involves replacing physical qubits in one code with logical qubits of another code. The Shor code encodes the general pure qubit state, see Eqn. 1.71, into the nine-qubit logical state $\left|\overline{\psi}\right\rangle$ using the logical code words $\left|\overline{0}\right\rangle = |000\rangle$ and $\left|\overline{1}\right\rangle = |000\rangle$ of the bit-flip code, see Eqn. 1.72 within the phase-flip code, see

Eqn. 1.79, expressed in the computational basis, see Eqn. 1.80 and 1.81, as

$$\begin{aligned}|\overline{\psi}\rangle &= \alpha\frac{1}{\sqrt{8}}(|\overline{0}\rangle + |\overline{1}\rangle) \otimes (|\overline{0}\rangle + |\overline{1}\rangle) \otimes (|\overline{0}\rangle + |\overline{1}\rangle) \\ &\quad + \beta\frac{1}{\sqrt{8}}(|\overline{0}\rangle - |\overline{1}\rangle) \otimes (|\overline{0}\rangle - |\overline{1}\rangle) \otimes (|\overline{0}\rangle - |\overline{1}\rangle) \\ &= \alpha\frac{1}{\sqrt{8}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \\ &\quad + \beta\frac{1}{\sqrt{8}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)\end{aligned} \quad . \tag{1.82}$$

The stabilizer measurements of the Shor code are

$$Z_1Z_2,\ Z_2Z_3, \quad Z_4Z_5,\ Z_5Z_6, \quad Z_7Z_8,\ Z_8Z_9, \tag{1.83}$$

$$X_1X_2X_3X_4X_5X_6,\ X_4X_5X_6X_7X_8X_9 \tag{1.84}$$

where subscripts index the qubit to which the operator applies. These stabilizer measurements follow from those of the bit-flip and phase-flip codes. The pairs of stabilizer measurements given in the top row are just those of the bottom-level bit-flip codes. The pair of stabilizer measurements given in the bottom row correspond to the stabilizer measurements for the top-level phase-flip code taking into account that, for the bit-flip code, $\overline{X} = XXX$.

The Shor code protects against any single bit-flip error, X, by virtue of the bottom-level bit-flip codes. It also protects against any single phase-flip error, Z, thanks to the top-level phase-flip code; there is some degeneracy in the syndrome, for example the errors Z_1, Z_2 or Z_3 each give the same syndrome, but this is not an issue since the same recovery operator, say Z_1, successfully corrects each of these errors. Finally, we note that since the Pauli Y operator is the product of X and Z up to a global phase, the Shor code protects against any single-qubit Pauli error. In the next subsection, we see that this is sufficient to protect against any single-qubit error.

Error discretization

The discretization of errors has been described as the *magic* [53] of quantum error correction. It allows a quantum state, which occupies a continuous statespace, to be protected from an infinite number of continuous errors, whilst only considering a finite number of discrete errors [54].

Any n-qubit coherent (i.e. unitary) error operator E can be expressed as a linear combination of operators from the n-qubit Pauli group $\mathcal{P}$, see Section 1.3.2. Suppose a logical qubit of some code $\mathcal{C}$ is in the state $|\overline{\psi}\rangle$, and the error E acts on $|\overline{\psi}\rangle$, then the state $E|\overline{\psi}\rangle$ can be expressed as a superposition of terms $\sigma_j|\overline{\psi}\rangle$ where $\sigma_j \in \mathcal{P}$. The stabilizer measurements of $\mathcal{C}$, which form part of the error correction process, project $E|\overline{\psi}\rangle$ onto one of the $\sigma_j|\overline{\psi}\rangle$ terms, (or a subset if the code is degenerate). Provided $\mathcal{C}$ can correct each σ_j, then $\mathcal{C}$ corrects E. It follows that a code that corrects all weight-t Pauli errors, corrects all t-qubit coherent errors.

Beyond coherent errors, the most general error, including interaction with the environment, can be described as an error channel that maps the pure logical state to a mixed state, see Section 1.3.4. Such an error channel $\mathcal{E}$ operating on a state ρ can be written as $\rho \to \mathcal{E}(\rho) = \sum_j E_j\rho E_j^\dagger$ where $\sum_j E_j^\dagger E_j = 1$ and E_j is in the span of the Pauli operators. Before the application of $\mathcal{E}$, the state ρ takes the form $\rho = |\overline{\psi}\rangle\langle\overline{\psi}|$, where $|\overline{\psi}\rangle$ is a logical state in the codespace of $\mathcal{C}$. After the application of $\mathcal{E}$, the state ρ is in a mixed state $\mathcal{E}(\rho)$ that can be considered a probabilistic sum of states $E_j \left|\overline{\psi}\right\rangle$. Provided $\mathcal{C}$ corrects each E_j, then $\mathcal{C}$ corrects $\mathcal{E}$. It turns out, therefore, that a code that corrects all weight t Pauli errors, corrects all t-qubit errors in general.

As promised, we have seen that the Shor code, which corrects single-qubit X and Z errors, corrects all single-qubit errors. In fact, codes that correct up to t errors, also correct small errors, on more than t qubits up to order ϵ^t. For example, suppose each qubit, labeled j, of the nine qubits in the Shor code experiences a small error, $I + \epsilon E_j$, then the total error E can be expanded as a sum of terms with no error, single-qubit errors and multi-qubit errors as

$$E = \bigotimes_j (I + \epsilon E_j) = I + \epsilon(E_1 + E_2 + \ldots + E_9) + O(\epsilon^2) \tag{1.85}$$

where E_j is a single-qubit error acting on qubit j. The Shor code, which corrects single-qubit errors, corrects E to order ϵ, which is a significant improvement provided ϵ is small.

Error correction conditions

The conditions under which a code corrects a set of errors is an important theorem [55, 56] in quantum error correction, which we now state and briefly discuss.

Theorem 1.1. *Suppose $\mathcal{E}$ is a set of errors $\{E_a\}$ acting on a Hilbert space $\mathcal{H}$. A quantum error correcting code, defined by the subspace $\mathcal{C} \subseteq H$, with basis $\{|\phi_j\rangle\}$, corrects any error in $\mathcal{E}$ if and only if*

$$\langle\phi_j| E_a^\dagger E_b |\phi_k\rangle = c_{ab}\delta_{jk} \tag{1.86}$$

where c_{ab} is a constant independent of j and k and $c_{ab} = c_{ba}^*$.

The above theorem is in a form that applies to subspace codes, which includes the codes considered in this thesis, but it can be generalized to other classes of error correction [57, 58]. It is illuminating to unpack Eqn. 1.86. The condition with $j \neq k$ says that an erroneous version of one codeword must remain distinguishable from all other (possibly erroneous) codewords; that is orthogonal states must remain orthogonal under any error. The condition with $j = k$ says that if an erroneous version of one codeword overlaps with a (possibly) erroneous version of the same codeword, then that overlap must be the same for all codewords altered by the same error so that the relative amplitude of the codewords is preserved in superposition; this allows for degeneracy where different errors map to the same logical state.

1.5.2 Stabilizer codes

The bit-flip code and Shor code, see Section 1.5.1, are examples of a particular class of quantum error correcting codes known as *stabilizer* codes [44, 59]. Stabilizer codes provide a straightforward generalization of classical binary linear codes, see Section 1.4, and as such are relatively straightforward to analyze algebraically and numerically.

Formalism

A stabilizer code encodes k logical qubits into n physical qubits using a codespace that is a 2^k-dimensional subspace of the 2^n-dimensional Hilbert space. The codespace $\mathcal{H}_C$ is defined, in terms of codewords $|\psi\rangle$, as

$$\mathcal{H}_C = \{|\psi\rangle \in \mathcal{H} \mid g\,|\psi\rangle = |\psi\rangle \;\; \forall\; g \in \mathcal{S}\} \tag{1.87}$$

where the stabilizer group $\mathcal{S}$ is an Abelian subgroup of the n-qubit Pauli group $\mathcal{P}$, see Eqn. 1.30, such that $-I \notin \mathcal{S}$. The centralizer of $\mathcal{S}$ is $\mathcal{Z}(\mathcal{S}) = \{P \in \mathcal{P} \mid PS = SP \;\forall\; S \in \mathcal{S}\}$. A stabilizer code $\mathcal{C}(\mathcal{S})$ is defined by a minimal set of m generators S_j, sometimes called *parity checks*, of $\mathcal{S}$

$$\mathcal{S} = \langle S_1, S_2, \ldots, S_m\rangle \tag{1.88}$$

and a pair of logical operators $(\overline{X}_j, \overline{Z}_j)$ for each encoded qubit such that

$$\overline{X}_j, \overline{Z}_j \in \mathcal{Z}(\mathcal{S}) \setminus \mathcal{S} \quad \text{and} \quad \overline{X}_j\overline{Z}_j = -\overline{Z}_j\overline{X}_j. \tag{1.89}$$

Defined in this way, the stabilizer code $\mathcal{C}(\mathcal{S})$ encodes $k = n - m$ logical qubits in n physical qubits and has distance

$$d = \min_{P \in \mathcal{Z}(\mathcal{S}) \setminus \mathcal{S}} \operatorname{wt}(P). \tag{1.90}$$

where $\operatorname{wt}(P)$ is the number of qubits upon which P acts non-trivially, so it corrects up to $t = (d-1)/2$ errors. We label $\mathcal{C}(\mathcal{S})$ as an $[[n, k, d]]$; note the use of double brackets for quantum codes.

There are several methods for constructing stabilizer codes, such as the CSS and homological constructions [49]; the latter method yields topological stabilizer codes and the surface code, in particular, see Section 1.5.5, which is the main subject of this thesis. There are also generalizations of stabilizer codes, such as subsystem codes [60, 61], and, of course, non-stabilizer codes [49]; we do not consider such codes other than to provide context for the surface code.

Decoding

The process of error correction is known as *decoding*; although it is not the opposite of encoding. The aim is to restore the logical qubit(s) to the codespace without inducing a non-trivial logical operation.

A stabilizer measurement is performed for each of the generators S_j of $\mathcal{S} = \langle S_1, S_2, \ldots, S_m\rangle$, and we define a syndrome vector $s \in \mathbb{Z}_2^m$ as

$$s = (s_1\, s_2\, \ldots\, s_m) \text{ such that } s_j = \begin{cases} 0, & \text{if measurement of } S_j \text{ yields } +1 \\ 1, & \text{otherwise} \end{cases}. \tag{1.91}$$

In terms of the error correction conditions, see Theorem 1.1, for a set of errors $\mathcal{E} = \{E_a\}$, if $E_a^\dagger E_b \notin \mathcal{Z}(\mathcal{S})$ then the syndrome can distinguish them, and if $E_a^\dagger E_b \in \mathcal{S}$ then E_a and E_b act the same on the codewords of $\mathcal{C}(\mathcal{S})$ and there is no need to distinguish them. That is $\mathcal{C}(\mathcal{S})$ corrects any error in $\mathcal{E}$ if and only if $E_a^\dagger E_b \notin \mathcal{Z}(\mathcal{S}) \setminus S \;\forall\; E_a, E_b \in \mathcal{E}$ [54].

The syndrome is used to attempt to identify, with high probability, a recovery operator R that reverses the action of the error E. A successful recovery operator acts equivalently to the error, i.e. $R = ES$ for some $S \in \mathcal{S}$.

Minimum-weight decoding [16] is perhaps the simplest approach to decoding. The idea is to choose a recovery operator $R = E_s$, where E_s is an error consistent with the syndrome s with minimum weight $\operatorname{wt}(E_s)$. The choice of R is not necessarily unique and the approach does not take into account such degeneracy. However, minimum-weight decoding often works well when the error rate is low and, in many cases, it can be implemented efficiently. We describe such an implementation for the surface code in Section 1.5.5 and define an adaptation to biased noise in Chapter 4.

Maximum-likelihood decoding [16] is the optimal approach to decoding in that it maximizes the probability of successful recovery. The idea is to consider equivalence classes of errors (cosets of the stabilizer group) and select a recovery operator from the most-likely class. Consider the codespace state ρ subject to a general Pauli error channel given by

$$\mathcal{E}(\rho) = \sum_{P \in \mathcal{P}} \pi(P) P \rho P^\dagger \tag{1.92}$$

where π is a probability distribution on the Pauli group $\mathcal{P}$. The action of $P, Q \in \mathcal{P}$ on ρ is the same, i.e. $P\rho P^\dagger = Q\rho Q^\dagger$, whenever $P\mathcal{S} = Q\mathcal{S}$, where $P\mathcal{S} = \{PS \mid S \in \mathcal{S}\}$ is a coset of the stabilizer group $\mathcal{S}$. $\mathcal{P}$ is a disjoint union of cosets $P_\alpha\mathcal{S}$, where P_α is some fixed representative of $P_\alpha\mathcal{S}$. The error channel can therefore be rewritten as

$$\mathcal{E}(\rho) = \sum_{\alpha} \pi(P_\alpha\mathcal{S}) P_\alpha \rho P_\alpha^\dagger \tag{1.93}$$

where $\pi(P\mathcal{S}) = \sum_{S \in \mathcal{S}} \pi(PS)$ is a coset probability. If P_s is some fixed Pauli operator consistent with syndrome s, then the set of all Pauli operators consistent with syndrome s is a coset of the centralizer of $\mathcal{S}$, $P_s\mathcal{Z}(\mathcal{S})$, which can be expressed as the disjoint union

$$P_s\mathcal{Z}(\mathcal{S}) = P_s\mathcal{S} \cup P_s\overline{X}\mathcal{S} \cup P_s\overline{Y}\mathcal{S} \cup P_s\overline{Z}\mathcal{S} \tag{1.94}$$

where $\overline{X}$, $\overline{Y} = -i\overline{Z}\overline{X}$ and $\overline{Z}$ are the logical operators for the encoded qubit (for simplicity we assume a single encoded qubit but the generalization to multiple encoded qubits is straightforward). Therefore, the post-measurement state can be expressed in terms of coset probabilities as

$$\rho_s = \pi(P_s\mathcal{S}) P_s \rho P_s^\dagger + \pi(P_s\overline{X}\mathcal{S}) P_s\overline{X}\rho\overline{X}P_s^\dagger + \pi(P_s\overline{Y}\mathcal{S}) P_s\overline{Y}\rho\overline{Y}P_s^\dagger + \pi(P_s\overline{Z}\mathcal{S}) P_s\overline{Z}\rho\overline{Z}P_s^\dagger \tag{1.95}$$

and we maximize the probability of successful recovery by choosing the recovery operator to be any element of the most-likely coset $P_s\mathcal{S}$, $P_s\overline{X}\mathcal{S}$, $P_s\overline{Y}\mathcal{S}$ or $P_s\overline{Z}\mathcal{S}$. In summary, the approach consists of selecting any Pauli operator consistent with the syndrome, evaluating 4^k coset probabilities, and returning a recovery operator from the most-likely coset. The evaluation of each coset probability is a sum over $|\mathcal{S}|$ elements, which is exponential in n. Therefore, although maximum-likelihood decoding maximizes the probability of successful recovery its exact implementation is not efficient. We describe an efficient approximate implementation for surface codes in Section 1.5.5, and define adaptations to biased noise and color codes in Chapter 3.

1.5.3 Fault-tolerance

In the preceding sections, we have assumed perfectly reliable measurements in describing how quantum error correcting codes protect a quantum state against noise. A practical code must handle faulty measurements to implement *fault-tolerant* error correction. The overhead of handling faulty measurements is one of the criteria by which codes are evaluated. One common approach is to repeat the measurements in time and decode the resulting higher-dimensional syndrome for both qubit and measurement errors [15]; we describe this approach for the surface code in Section 1.5.5 and use it in Chapter 4. Some codes, such as the 3D gauge color code [62] and 4D toric code [15], admit single-shot error correction where meta-checks provide additional information about error location.

Computation

Fault-tolerant quantum error correction, as described above, implements a quantum memory. Full fault-tolerant quantum computation requires additional protocols to perform a universal set of logical gates in a protected way, i.e. the Clifford gates and a non-Clifford gate such as the T-gate, see Section 1.3.4. Such protocols fall outside the scope of this thesis, however we briefly mention some of them since the overhead of implementing them is another criterion by which codes are evaluated. The main idea is to implement logical gates directly on the encoded state, whilst avoiding the spread of errors beyond the code distance. A low-overhead approach is to use transversal gates, where a logical gate is performed by applying single-qubit gates to the physical qubits of the encoded state, or two-qubit gates between corresponding pairs of physical qubits of encoded states; Eastin and Knill [63] have shown that no single code admits a universal set of transversal gates. Another approach uses constant-depth circuits, which also limits the spread of errors; Bravyi and Koenig [64] have shown that no 2D stabilizer code admits a universal set using this approach. There are ways to circumvent some of these restrictions, such as gauge fixing, in which some logical qubits are not used to protect against errors but to switch between codes with different sets of transversal gates; for example the 2D color code (with transversal Clifford gates) and the 3D color code (with transversal T-gate) [65, 66]. Other approaches include lattice surgery, involving multi-qubit logical measurements, and encoding qubits in topological defects such as punctures or dislocations and braiding or fusing these defects; for example these techniques can be used to implement the Clifford gates on the surface code [15, 67–69]. Finally we mention magic state distillation [70], which is usually considered high-overhead and involves 'distilling' a high-fidelity state that, when teleported into the code, can be used to implement the desired logical gate.

Threshold theorem

The *threshold theorem* has several formulations [8–12], we state the version given in Ref. [16].

Theorem 1.2. *An ideal quantum circuit of size N can be simulated with arbitrary small error ϵ by a noisy quantum circuit subjected to independent stochastic noise of strength less than $p < p_c$ where the noisy quantum circuit is of size $O(N(\log(N/\epsilon))^m)$ with some constant m.*

This is perhaps the most important theorem in quantum computation, since it tells us that quantum computation is possible, with some physically reasonable assumptions about the noise. The proof involves analyzing fault-tolerant protocols for state preparation, logical gates and measurement, along with a family of quantum error correcting codes, whose distance grows with the number of physical qubits.

In the context of quantum error correction, we define the *threshold error rate* as the physical error rate below which the logical error rate can be made arbitrarily small by increasing the code size. The threshold error rate, so defined, is a function of the code, error channel and decoder. We are interested in the threshold error rate at three levels of abstraction: code capacity, phenomenological fault-tolerant, and circuit-based fault-tolerant. The *code-capacity* threshold considers that errors may alter the state of physical qubits but measurements are perfectly reliable. The *phenomenological fault-tolerant* threshold, in addition to errors on qubits, considers that errors may alter stabilizer measurement outcomes without regard to the details of the measurement circuits. The *circuit-based fault-tolerant* threshold, in addition to errors on qubits, considers that errors may occur at any step of the stabilizer measurement circuits. Each threshold definition has its merits. In Chapters 2 and 3, we evaluate the code-capacity threshold using maximum-likelihood decoding in order to study inherent properties of the surface code with biased noise. In Chapter 4, we evaluate phenomenological fault-tolerant thresholds to demonstrate how the surface code performs with biased noise in an experimentally realistic scenario assuming faulty measurements and using an efficient but suboptimal decoder. If a particular hardware implementation is to be considered then the circuit-based fault-tolerant threshold is of interest to get a maximum target error rate for the physical qubits. In all cases, the threshold error rate is an important criterion by which a code is evaluated.

1.5.4 Code criteria

We are now in a position to list some criteria by which quantum error correcting codes can be evaluated. The following criteria are independent of physical architecture:

- High threshold error rate.
- Low decoding complexity.
- High encoding rate (k/n).
- Low logical gate overhead.

The following criteria are important for many physical architectures:

- Local stabilizers.
- 2D layout.

In section 1.5.5, we highlight why the surface code is considered one of the most promising practical codes, whilst noting its limitations. In Chapters 2, 3 and 4, we show that the surface code can achieve even higher threshold error rates with relevant noise models whilst maintaining decoding efficiency.

1.5.5 Topological codes

Topological stabilizer codes are a class of stabilizer code that has some very interesting features. The main idea is that logical qubits are encoded in topological degrees of freedom, whose extent is large or even global, while stabilizers are local. Assuming errors act locally and can be corrected or restricted in extent, then the logical qubits are protected from noise.

Surface code construction

The most well-known and well-studied topological stabilizer code is the surface code. The term *surface code* is sometimes used interchangeably with *toric code* [13], which has periodic boundaries, and *planar code* [14], which has open boundaries. We will generally use surface code when referring to the case with open boundaries, and either qualify the term or use toric code when referring to the case with periodic boundaries.

The surface code with periodic boundaries, or toric code, is defined as follows. Define a square lattice cellulation of a torus, which can be represented as a square with periodic boundaries, see Fig. 1.2. To each edge of the lattice assign a physical qubit, to each vertex assign an X-type stabilizer, and to each plaquette assign a Z-type stabilizer. An X-type stabilizer is a product of Pauli X operators on qubits assigned to edges incident to the vertex. A Z-type stabilizer is a product of Pauli Z operators on qubits assigned to edges forming the boundary of the plaquette. Such stabilizers commute, since stabilizers of different types either coincide on no edges or and even number of edges. The set of all such stabilizers forms an over-complete generating set of the stabilizer group $\mathcal{S}$ for the toric code. Logical operators on the toric code are formed of non-trivial closed loops of X or Z operators, i.e. loops that cannot be formed as products of stabilizers. The distance of the toric code is the number of edges in the smallest non-trivial closed loop. The toric code, so defined, with $d \times d$ horizontal edges, encodes two logical qubits in $n = 2d^2$ physical qubits and has distance d.

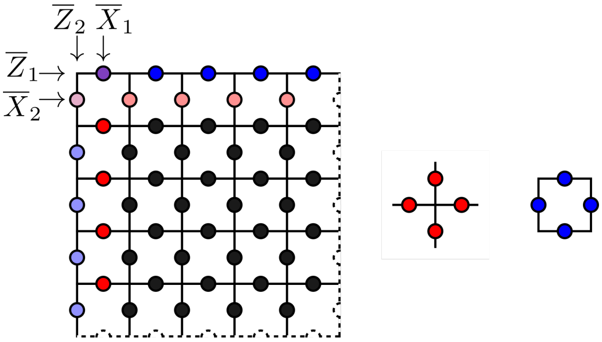


Figure 1.2: Surface code with periodic boundaries (toric code). Physical qubits are on edges. Logical operators for encoded qubits labeled 1 and 2 are: $\overline{X}_1$ ($\overline{Z}_1$) a column (row) of Pauli X (Z) operators on horizontal edges, and $\overline{X}_2$ ($\overline{Z}_2$) a row (column) of Pauli X (Z) operators on vertical edges. X-type stabilizers are a product of Pauli X operators on edges incident to a vertex. Z-type stabilizers are a product of Pauli Z operators on edges forming the boundary of a plaquette.

The above description can be expressed as a homological construction and generalized to codes with boundaries and higher dimensions [49]. For example, the torus is a 2-dimensional manifold, and given a square lattice cellulation, we assign qubits to 1-cells (edges), X-type stabilizers to 0-cells (vertices), and Z-type stabilizers to 2-cells (plaquettes), and the logical operators are non-trivial 1-cycles and 1-cocycles.

The surface code with open boundaries, or planar code, follows a similar homological construction relative to its boundaries, see Fig. 1.3. Physical qubits are assigned to edges as for the toric code. The stabilizers are defined as for the toric code, except at the boundaries where they are supported on just three qubits. The top/bottom boundaries with 3-qubit X type stabilizers are called *smooth*, and the left/right boundaries with 3-qubit Z-type stabilizers are called *rough*. The logical operators are formed of a cycle of Pauli X operators between opposing smooth boundaries, and a cycle of Pauli Z operators between opposing rough boundaries. The surface code, so defined, with $d \times d$ horizontal edges, encodes one logical qubit in $n = 2d^2 - 2d + 1$ physical qubits and has distance d.

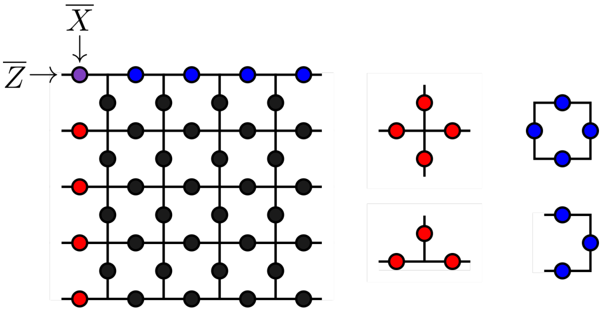


Figure 1.3: Surface code with open boundaries (planar code). Physical qubits are on edges. Logical operators for the encoded qubit are: $\overline{X}$ ($\overline{Z}$) a column (row) of Pauli X (Z) operators on horizontal edges. X-type stabilizers are a product of Pauli X operators on edges incident to a vertex. Z-type stabilizers are a product of Pauli Z operators on edges forming the boundary of a plaquette.

There are several variants of surface code, as well as many other topological codes [49]. In this thesis, we focus on the most well-studied topological code, the surface code, as defined above, and tailor it to biased noise models. The modifications that we analyze are changing the basis of Pauli operator used to define stabilizers and logical operators, introduced in Chapter 2, and changing relative boundary lengths and orientation, introduced in Chapter 3. These modifications are analyzed in a fault-tolerant context in Chapter 4. For comparison, we also analyze the performance of another 2D topological stabilizer code, the color code [71], in Chapter 3, where is it defined.

Surface code decoding

Correcting an error on the surface code starts, as for any stabilizer code, with the measurement of stabilizers yielding a syndrome. The topological nature of the surface code with local stabilizers means that the spatial location of the stabilizers that are violated by a given error provides additional information that can be used in correcting the error. We start by describing an efficient minimum-weight decoding algorithm that exploits this spatial location information to effectively decode the surface code for certain error channels. We then outline how this decoding algorithm can be extended to the fault-tolerant context of unreliable measurements. Next we describe an approximate maximum-likelihood decoding algorithm that effectively decodes the surface code for general Pauli error channels subject to a trade-off between efficiency and accuracy. Finally we provide an illustration that constrasts how the two algorithms perform for a particular error configuration. Variations on the decoding algorithms described here are used throughout Chapters 2, 3 and 4.

The basic *minimum-weight matching* [15] (MWM) decoder exploits the spatial location information of violated stabilizers as follows. The codespace of the surface code can be described as the ground state of the Hamiltonian $H = -\sum_v A_v - \sum_p B_p$, where A_v and B_p are X-type vertex and Z-type plaquette stabilizers, respectively. Stabilizers that are violated by a given error can be interpreted as excitations or quasiparticles called *anyons* or *defects*. Since there are two types of stabilizer A_v and B_p, there are two types of defects, usually labelled m and e respectively, by analogy with magnetic and electric charges. Suppose an error E consists of Pauli X operators in the bulk of the code, due to the bit-flip channel. A symmetry of the code and this noise model is that e defects occur in pairs at each end of a string of X operators, see Fig. 1.4. Any pair of e defects can be fused and annihilated by applying a recovery operator R that applies strings of X operators between them. Decoding is successful if the combined recovery and error are in the stabilizer group, i.e. $RE \in \mathcal{S}$. That is if the combined recovery and error form trivial closed loops of X operators, which are products of X-type stabilizers. MWM decoding constructs a recovery by building a graph of e defects with edges between defect pairs weighted by a Manhattan distance function, performing minimum-weight perfect matching over the graph, and applying X operators along the shortest path between matched defects. A second round of decoding is performed for m defects to correct Z errors. On the surface code with open boundaries, e (m) defects can also be paired to rough (smooth) boundaries, which can be implemented by matching to 'virtual' defects off the boundaries. This decoding algorithm does not take into account degeneracy or correlations between X and Z errors, corresponding to Y errors, although some attempts have been made to modify it to partially account for such correlations [21, 72, 73]. However the decoder is efficient, since minimum-weight perfect matching, using the Blossom algorithm [74], has runtime $O(n^3)$, in the worst case, and $O(n)$ [75], on average, where n is the number of physical qubits in the code. For certain noise models, specifically the bit-flip channel, it achieves a threshold close to that of the optimal decoder, as detailed in the following subsection.

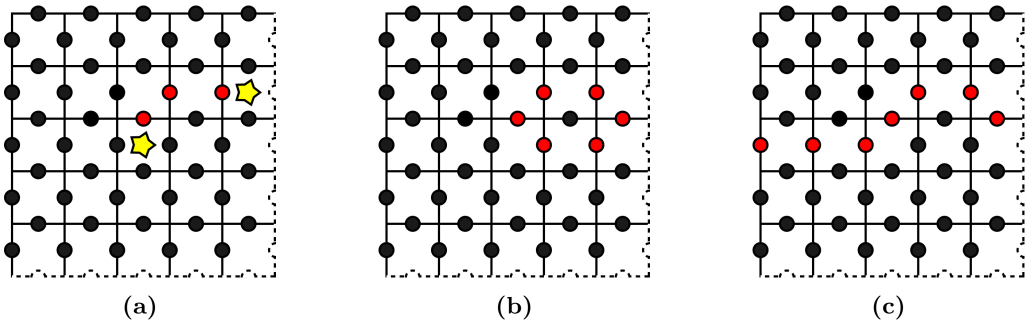
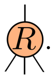


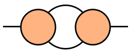
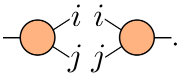
Figure 1.4: Minimum-weight matching decoding of the surface code. (a) Error consisting of a string of Pauli X operators (red circles) and violated stabilizers (yellow stars). (b) Error and successful recovery is a trivial closed loop of X operators, i.e. a product of X-type stabilizers. (c) Error and unsuccessful recovery is a non-trivial closed loop of X operators.

The MWM decoder is readily extended to the fault-tolerant context of unreliable measurements. In this case, stabilizer measurements are repeated in time so that the syndrome is 2+1 dimensional. Defects are located not at the location of violated stabilizers but at locations where, possibly faulty, stabilizer measurement outcomes change from +1 to −1 or vice versa. This can be viewed as replacing the 2-dimensional stabilizers, or parity checks, violated by qubit errors, with 3-dimensional parity checks violated by qubit and measurement errors. The decoding then proceeds as for the case with reliable measurements, except the distance function between defects may be modified due to the relative probability of qubit and measurement errors. In Chapter 4, we define a decoder that uses MWM to exploit symmetries of a modified surface code with biased noise.

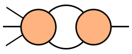
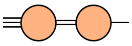
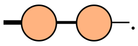
We now introduce a graphical calculus, known as *tensor network notation* [76], that will be useful in describing the approximate maximum-likelihood decoder. A tensor, which can be thought of as a multi-dimensional array for our purposes, is represented by a node with legs, where the number of legs is the number of indices of the array. In relation to Einstein notation, we have

$R^{\rho}{}_{\sigma\mu\nu} =$ . (1.96)

Tensors can be contracted by joining legs, which corresponds to summing over indices

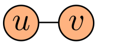
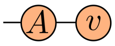
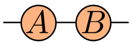
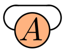
 $= \sum_{i,j}$ . (1.97)

Legs can be grouped or split, which corresponds to reducing or increasing the number of indices while increasing or reducing the size of indices accordingly

 $=$  $=$ . (1.98)

Some common operations are shown in different equivalent notations in Table 1.1.

Table 1.1: Common operations represented in matrix / vector, Einstein and tensor network notations.

Matrix/vector	Einstein	Tensor network
$u \cdot v$	$u_\alpha v^\alpha$	
Av	$A^\alpha_\beta v^\beta$	
AB	$A^\alpha_\beta B^\beta_\gamma$	
$\mathrm{tr}(A)$	A^α_α	

An *approximate maximum-likelihood* decoder for the surface code was introduced by Bravyi, Suchara and Vargo [77]; we refer to it as the BSV decoder. Recalling Section 1.5.2, exact maximum-likelihood decoding for the surface code proceeds as follows: find any Pauli operator P_s consistent with the syndrome s (this is easily done by fusing defects to boundaries); evaluate the coset probabilities $\pi(P_s\mathcal{S})$, $\pi(P_s\overline{X}\mathcal{S})$, $\pi(P_s\overline{Y}\mathcal{S})$, $\pi(P_s\overline{Z}\mathcal{S})$ where $\mathcal{S}$ is the stabilizer group; and return the recovery operator R as any element from the most-likely coset. Evaluating the exact coset probabilities is highly inefficient. The idea of the BSV decoder is to define a tensor network that would contract to the exact coset probability but to perform the contraction approximately. The geometry of the tensor network, which is defined in Fig. 1.5, respects that of the code with a local tensor defined for each stabilizer and qubit location. The *bond dimension* or size of each index of the local tensors is initially 2. The outer columns of the tensor network have the form of matrix product states (MPS), and the inner columns have the form of matrix product operators (MPO) [76]. The bond dimension of each MPS / MPO is the maximum bond dimension between tensors within the MPS / MPO and is therefore initially 2. Approximate contraction of the tensor network is tuned via a parameter, χ, which bounds the bond dimension and provides a trade-off between efficiency and accuracy. The contraction proceeds as follows: the leftmost MPO is contracted into the left MPS (i.e. tensors are contracted pairwise between the MPS and MPO) resulting in an updated left MPS with bond dimension 4; this contraction is repeated, with successive leftmost MPOs, resulting in the bond dimension of the left MPS doubling each time; if at any step the bond dimension of the left MPS exceeds χ, a truncation procedure is applied to reduce its bond dimension to χ, finally the tensor network consists of a single column which is contracted exactly. The truncation procedure [78, 79] puts the MPS in a canonical form and sweeps through the chain of tensors using the singular value decomposition to reduce the size of each tensor by retaining only the χ largest Schmidt values. With χ exponential in n, the number of physical qubits, the contraction value of the tensor network is exact but the contraction is highly inefficient. A smaller value of χ improves the efficiency of the contraction but may reduce the accuracy. The approximate contraction has runtime $O(n\chi^3)$. For certain noise models, it has been shown that the BSV decoder closely approximates exact maximum-likelihood decoding with modest χ [77].

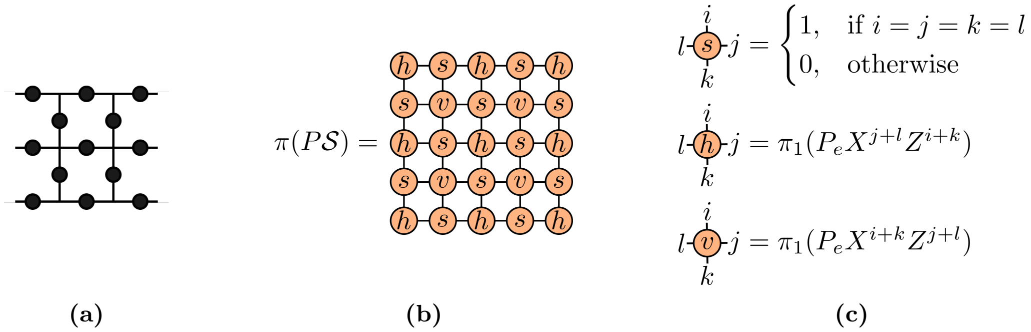

$\pi(P\mathcal{S}) =$
h
s
v
$l\text{-}s\text{-}j = \begin{cases} 1, & \text{if } i = j = k = l \\ 0, & \text{otherwise} \end{cases}$
$l\text{-}h\text{-}j = \pi_1(P_e X^{j+l} Z^{i+k})$
$l\text{-}v\text{-}j = \pi_1(P_e X^{i+k} Z^{j+l})$
i
k
(a)
(b)
(c)

Figure 1.5: Tensor network construction for coset probability used in approximate maximum-likelihood decoding of the surface code with the BSV decoder. (a) 3×3 surface code with qubits assigned to edges and X-type (Z-type) stabilizers assigned to vertices (plaquettes). (b) Corresponding tensor network that evaluates to the coset probability $\pi(P\mathcal{S})$, where P is an n-qubit Pauli operator, chosen to be consistent with the syndrome, and $\mathcal{S}$ is the stabilizer group. The tensors labeled s, h and v correspond to stabilizers, qubits on horizontal edges and qubits on vertical edges, respectively. (c) Definitions of the s, h and v tensors. The s tensors impose valid stabilizers. The h and v tensor elements take values from the probability distribution π_1 over the single-qubit Pauli group, where P_e is the restriction of P to the edge corresponding to the tensor.

The BSV decoder has not yet been extended to the fault-tolerant context of unreliable measurements. This would involve approximate contraction of 3D tensor networks and it is an open question whether this can be done efficiently with sufficient accuracy for approximate maximum-likelihood decoding. In Chapter 2, we use the BSV decoder to analyze thresholds of the surface code with biased noise. In Chapter 3, we define a similar tensor network decoder for the surface code that achieves improved performance with biased noise, and we define an analogous decoder for the color code to perform a comparative analysis.

Figure 1.6 provides an illustration that contrasts how the MWM decoder fails whereas the BSV decoder succeeds for a particular error configuration. This type of error configuration is studied in Chapter 3, and exploited in Chapters 2 and 4 to achieve improved decoding performance.

Surface code features

In Section 1.1, we stated that the surface code is one of the most promising quantum error correcting codes. We now provide some justification for that statement by considering how well the surface code meets the code criteria given in Section 1.5.4.

As a 2D topological stabilizer code, the surface code has a 2D layout and local stabilizers, satisfying an important criteria for many physical architectures. In such architectures, designed with scalability in mind, qubits are closely packed facilitating local interactions and favoring a

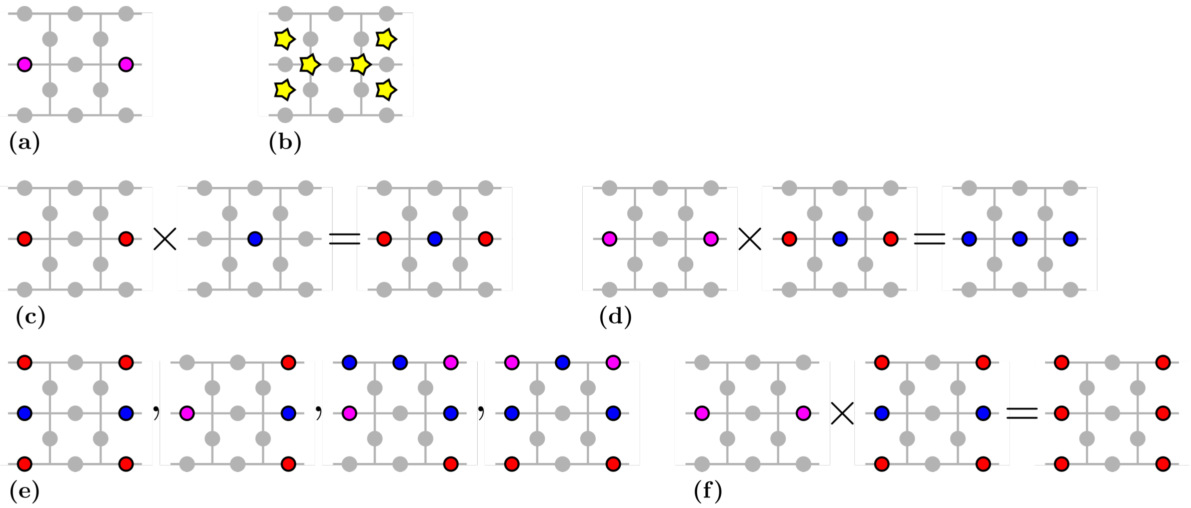


Figure 1.6: Example error configuration that causes the MWM decoder to fail, whereas the BSV decoder succeeds. (a) Error configuration consisting of Pauli Y operators on two qubits. (b) Corresponding syndrome consisting of plaquette and vertex defects. (c) The MWM decoder constructs a recovery operator by first pairing plaquette defects and then pairing vertex defects. (In this case, the combined recovery operator is not of minimum weight.) (d) Applying the recovery operator from MWM decoding to the original error results in a non-trivial logical operation. (e) The BSV decoder constructs a Pauli operator P_s consistent with the syndrome and determines the most-likely of the four logical cosets $P_s\mathcal{S}$, $P_s\overline{X}\mathcal{S}$, $P_s\overline{Y}\mathcal{S}$ and $P_s\overline{Z}\mathcal{S}$, where $\mathcal{S}$ is the stabilizer group. (In this case, the most-likely coset is $P_s\mathcal{S}$). (f) Applying the recovery operator from BSV decoding to the original error results in a trivial operation (i.e. a product of stabilizers.)

2D layout for ease of addressing qubits. Another strength of the surface code, as a topological code, is that it really represents a family of codes whose distance naturally increases with code size, allowing a target logical failure rate to be achieved in a straightforward manner.

The ease with which a universal set of logical gates can be implemented is another important criteria. Some of the lowest-overhead approaches are transversal gates or constant-depth circuits. As discussed in Section 1.5.3, no single code admits a universal set of transversal gates [63] and no 2D stabilizer code admits a universal set using constant-depth circuits [64]. However, relatively low-overhead approaches [15, 67–69] have been found to implement a universal set of logical gates on the surface code.

One major limitation of the surface code is its encoding rate, i.e. the ratio of logical qubits to physical qubits, k/n. Increasing the size of a surface code in terms of number of physical qubits n, increases the distance d but does not increase the number of logical qubits k. In the limit as n tends to infinity, the surface code has zero rate. However, it has been shown that any 2D topological code must satisfy the bound $kd^2 \leq O(n)$ [80], and that the surface code saturates this bound.

A major strength of the surface code is its high threshold error rates, which are amongst the highest known [16] and can be achieved with low-complexity decoders. For topological codes, the error threshold can be related to a phase transition of a classical statistical mechanical model with quenched disorder [15]. Using this approach optimal code-capacity thresholds of the surface code are estimated to be $p_c \approx 10.9\%$ [18, 19] with bit-flip noise and $p_c \approx 18.9\%$ [20] with depolarizing noise. The minimum-weight matching decoder, described in the previous subsection, is an efficient matching decoder that achieves code-capacity thresholds of $p_c \approx 10.3\%$ [22] with bit-flip noise, and $p_c \approx 15.5\%$ [23] with depolarizing noise; with some modifications to take account of correlations between X and Z errors this has been improved to $p_c \approx 17.8\%$ [21]. The approximate maximum-likelihood decoder, described in the previous subsection, is an efficient tensor-network decoder that achieves very close to optimal code-capacity thresholds for bit-flip and depolarizing noise [77]; indeed, there is a direct mapping between such tensor-network decoders and the statistical mechanical models mentioned above [81]. The color code, which can be seen as a folded surface code [82], exhibits similar high thresholds with bit-flip and depolarizing noise [20]. In Chapters 2 and 3, we use tensor-network decoders not only to confirm the optimal code-capacity thresholds with bit-flip and depolarizing noise on the surface code and color code, but also to demonstrate ultrahigh code-capacity thresholds with biased noise on the surface code. In the fault-tolerant context, the surface code also achieves some of the highest thresholds. The optimal phenomenological fault-tolerant threshold of the surface code with bit-flip noise is estimated using statistical mechanical methods to be $p_c \approx 3.3\%$ [24]; a similar analysis with depolarizing noise is yet to be performed but treating bit flips and phase flips separately gives a lower bound of $p_c \approx (3/2)3.3\% = 4.95\%$. The minimum-weight matching decoder achieves a phenomenological fault-tolerant threshold of $p_c \approx 2.9\%$ [22] with bit-flip noise. Tensor-network decoders have not yet been shown to efficiently generalize to fault-tolerant decoding. In Chapter 4, we define an efficient matching decoder that exploits symmetries of the code and noise model to achieve phenomenological fault-tolerant thresholds in excess of 5% with biased noise.

Codes other than the surface code can support low-overhead universal logical gates, particularly if we are prepared to relax the 2D layout requirement [65, 66] and allow code switching, as discussed in Section 1.5.3. Moving to higher dimensions also opens up the possibility of self-correcting quantum memories and single-shot error correction [15, 62, 83, 84]. 2D subsystem codes have a more relaxed bound on rate $kd \leq O(n)$ with $d^2 \leq O(n)$ [85, 86]. Allowing non-local stabilizers, in low-density parity check (LDPC) codes for example, brings the possibility of constant-rate codes [87–91]. However, all of these alternative approaches introduce significant practical challenges, such as lack of realizations in three physical dimensions, lack of asymptotic thresholds or efficient decoding, or even lack of known constructions. If, for practical reasons, we restrict our attention to 2D layouts with local stabilizers, then the surface code is currently one of the most promising known codes, along with the closely-related color code. In this thesis, we identify previously unknown features of the surface code that make it an even more appealing code.

1.6 Simulations

In this section, we introduce Qecsim [92], the software package developed to perform the numerical analysis presented in Chapters 2, 3 and 4. This numerical analysis required nearly 1 500 000 core-hours of simulations on HPC facilities provided by the National Computing Infrastructure and the University of Sydney.

Qecsim is a Python3 [93] library for simulating quantum error correction using stabilizer codes. It is lightweight, modular and extensible, allowing additional codes, error models and decoders to be plugged in. Codes, error models and decoders are defined using a rich object model. The code and error model objects expose data as arrays for efficient syndrome evaluation and decoding. All processing of arrays is carried out efficiently using linear algebra functions of the NumPy [94] and SciPy [95] libraries. All computationally intensive tasks are processed using subroutines implemented in C/C++ [96].

The core modules of Qecsim are qecsim.model, qecsim.app, qecsim.cli and qecsim.paulitools. The qecsim.model module defines three key abstract classes: StabilizerCode, ErrorModel and Decoder, see Fig. 1.7. Simulations are executed via the API by passing concrete implementations of the key classes to run functions of the qecsim.app module, see Algorithm 1.1. The output statistics of simulations are in a structured form and can be merged by calling the merge function of the qecsim.app module. The qecsim.cli module exposes a CLI for the run and merge functions of the qecsim.app module. Finally the qecsim.paulitools module provides functions to manipulate Pauli operators in both string and binary symplectic form [49]; for example the 4-qubit Pauli $IZYX$ operator is represented, up to a global phase, as $[0\,0\,1\,1|0\,1\,1\,0]$ where the left-side represents $IIXX$ and the right-side represents $IZZI$. These core modules are supplemented by the shared packages qecsim.tensortools and qecsim.pypm, which provide support for tensor-network and matching decoders. The core of Qecsim is lightweight with core modules implemented in just 759 source lines of code (SLOC) and the shared packages in a further 206 SLOC, excluding tests.

Several concrete implementations of codes, error models and decoders are provided as plugins. Code implementations include 5-qubit, Steane, toric, planar, color, rotated-toric and rotated-planar. Error model implementations include bit-flip, phase-flip, depolarizing, biased-depolarizing and spatially-correlated. Decoder implementations include a minimum-weight decoder for 5-qubit and Steane codes, various tensor-network decoders for planar and color codes, and various matching decoders for toric and planar codes. The plugins are implemented in 4894 SLOC in total ($\sim$ 140 SLOC on average), excluding tests.

Finally, Qecsim provides a fully-documented API and CLI and has a suite of over 3000 unit tests with 97% code coverage. It is our intention to make Qecsim available to the wider community in the near future.

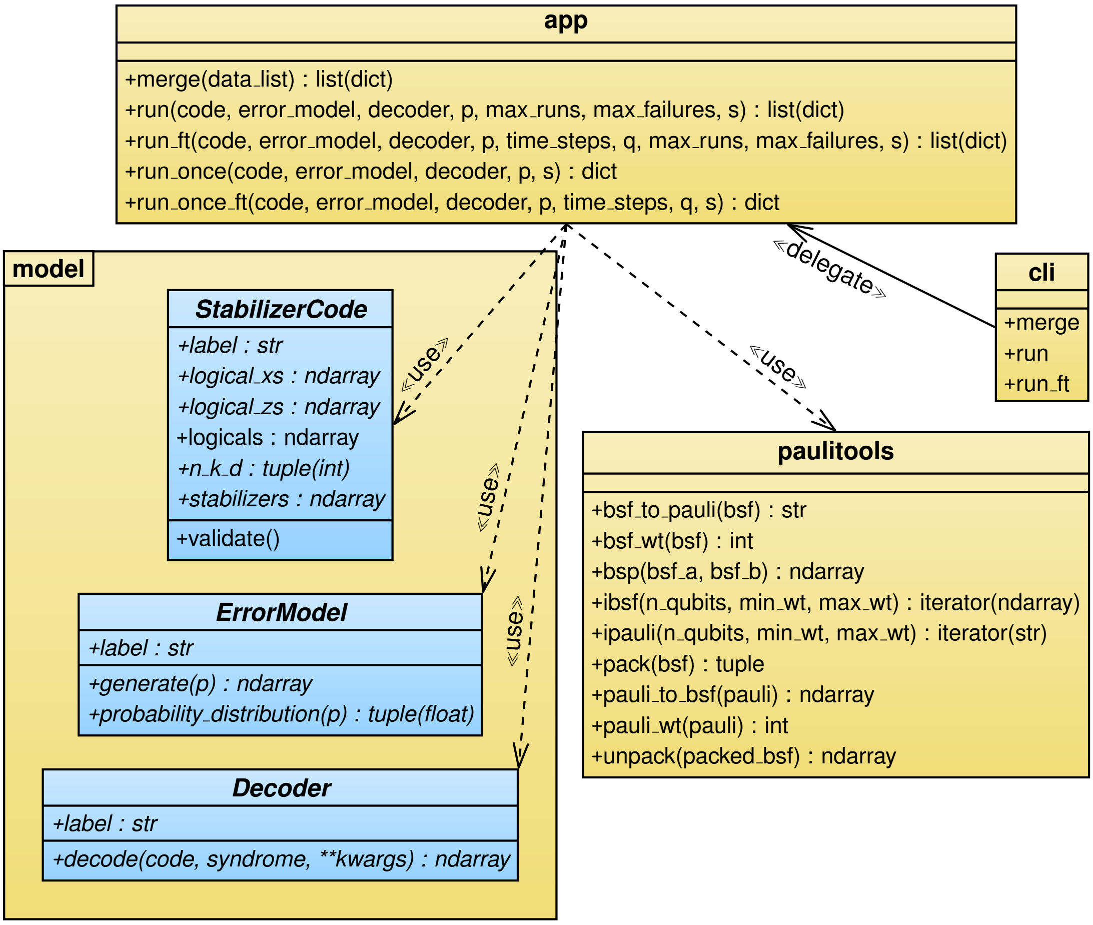

app
+merge(data_list) : list(dict)
+run(code, error_model, decoder, p, max_runs, max_failures, s) : list(dict)
+run_ft(code, error_model, decoder, p, time_steps, q, max_runs, max_failures, s) : list(dict)
+run_once(code, error_model, decoder, p, s) : dict
+run_once_ft(code, error_model, decoder, p, time_steps, q, s) : dict
model
StabilizerCode
+label : str
+logical_xs : ndarray
+logical_zs : ndarray
+logicals : ndarray
+n_k_d : tuple(int)
+stabilizers : ndarray
+validate()
ErrorModel
+label : str
+generate(p) : ndarray
+probability_distribution(p) : tuple(float)
Decoder
+label : str
+decode(code, syndrome, **kwargs) : ndarray
«use»
«use»
«use»
«use»
«delegate»
cli
+merge
+run
+run_ft
paulitools
+bsf_to_pauli(bsf) : str
+bsf_wt(bsf) : int
+bsp(bsf_a, bsf_b) : ndarray
+ibsf(n_qubits, min_wt, max_wt) : iterator(ndarray)
+ipauli(n_qubits, min_wt, max_wt) : iterator(str)
+pack(bsf) : tuple
+pauli_to_bsf(pauli) : ndarray
+pauli_wt(pauli) : int
+unpack(packed_bsf) : ndarray

Figure 1.7: Qecsim core Python modules: qecsim.model, qecsim.app, qecsim.cli, and qecsim.paulitools. Plugins provide implementations of the key abstract classes: StabilizerCode, ErrorModel and Decoder. Parameters p, q and s are qubit and measurement error probabilities and random seed, respectively. Output type *ndarray* is the NumPy array datatype.

```
Function run_once:
    // code, error_model and decoder are concrete implementations provided by plugins
    Input: code, error_model, decoder, error_probability
    Output: success
    $S \leftarrow$ code.stabilizers;                              // extract code properties
    $L \leftarrow$ code.logicals;
    $e \leftarrow$ error_model.generate(code, error_probability);   // delegate error generation
    $y \leftarrow e \odot S^T$;                                       // evaluate syndrome
    $r \leftarrow$ decoder.decode(code, $y$);                        // delegate decoding
    assert $(r \oplus e) \odot S^T = 0$;                  // test commutation with stabilizers
    success $\iff$ $(r \oplus e) \odot L^T = 0$;              // test commutation with logicals
    return success;
```

Algorithm 1.1: An example run function from the Qecsim qecsim.app module. This run function executes a single simulation assuming reliable stabilizer measurements. Pauli operators are in binary symplectic form, and the binary symplectic product $\odot$ is defined as $A \odot B \equiv A\Lambda B \bmod 2$, where $\Lambda = \left(\begin{smallmatrix} 0 & I \\ I & 0 \end{smallmatrix}\right)$ and 0 and I are appropriately-sized null and identity matrices.

References

[1] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, 10th ed., (2011).

[2] B. P. Lanyon, J. D. Whitfield, G. G. Gillett, *et al.*, "Towards quantum chemistry on a quantum computer," *Nature Chemistry*, **2**, p. 106, arXiv:0905.0887, (2010).

[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, **549**, p. 195, arXiv:1611.09347, (2017).

[4] F. Arute, K. Arya, R. Babbush, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, **574**, 7779, pp. 505–510, (2019).

[5] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff, "Leveraging secondary storage to simulate deep 54-qubit sycamore circuits," arXiv:1910.09534, (2019).

[6] S. Aaronson, "Why Google quantum supremacy milestone matters," *The New York Times*, (2019). `https://nyti.ms/2BVxMjj`.

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

[8] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error," in *Proc. STOC 1997*, (1997), arXiv:quant-ph/9611025.

[9] D. Aharonov and M. Ben-Or, "Fault-tolerant quantum computation with constant error rate," *SIAM J. Comput.*, **38**, pp. 1207–1282, arXiv:quant-ph/9906129, (2008).

[10] P. Aliferis, D. Gottesman, and J. Preskill, "Quantum accuracy threshold for concatenated distance-3 codes," *Quantum Info. Comput.*, **6**, pp. 97–165, arXiv:quant-ph/0504218, (2006).

[11] A. Y. Kitaev, "Quantum computations: algorithms and error correction," *Russian Mathematical Surveys*, **52**, pp. 1191–1249, (1997).

[12] E. Knill, R. Laflamme, and W. H. Zurek, "Resilient quantum computation," *Science*, **279**, 5349, pp. 342–345, arXiv:quant-ph/9702058, (1998).

[13] A. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys. (N. Y.)*, **303**, 1, pp. 2 – 30, arXiv:quant-ph/9707021, (2003).

[14] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," arXiv:quant-ph/9811052, (1998).

[15] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, **43**, 9, pp. 4452–4505, arXiv:quant-ph/0110143, (2002).

[16] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, **87**, pp. 307–346, arXiv:1302.3428, (2015).

[17] E. T. Campbell, B. M. Terhal, and C. Vuillot, "Roads towards fault-tolerant universal quantum computation," *Nature*, **549**, pp. 172–179, arXiv:1612.07330, (2017).

[18] F. Merz and J. T. Chalker, "Two-dimensional random-bond Ising model, free fermions, and the network model," *Phys. Rev. B*, **65**, p. 054425, arXiv:cond-mat/0106023, (2002).

[19] A. Honecker, M. Picco, and P. Pujol, "Universality class of the Nishimori point in the 2D $\pm J$ random-bond Ising model," *Phys. Rev. Lett.*, **87**, p. 047201, arXiv:cond-mat/0010143, (2001).

[20] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, "Strong resilience of topological codes to depolarization," *Phys. Rev. X*, **2**, p. 021004, arXiv:1202.1852, (2012).

[21] B. Criger and I. Ashraf, "Multi-path Summation for Decoding 2D Topological Codes," *Quantum*, **2**, p. 102, arXiv:1709.02154, (2018).

[22] C. Wang, J. Harrington, and J. Preskill, “Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory,” *Ann. Phys. (N. Y.)*, **303**, 1, pp. 31 – 58, arXiv:quant-ph/0207088, (2003).

[23] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg, “Threshold error rates for the toric and planar codes,” *Quantum Info. Comput.*, **10**, pp. 456–469, arXiv:0905.0531, (2010).

[24] T. Ohno, G. Arakawa, I. Ichinose, and T. Matsui, “Phase structure of the random-plaquette Z2 gauge model: accuracy threshold for a toric quantum memory,” *Nucl. Phys. B*, **697**, 3, pp. 462 – 480, arXiv:quant-ph/0401101, (2004).

[25] R. Raussendorf and J. Harrington, “Fault-tolerant quantum computation with high threshold in two dimensions,” *Phys. Rev. Lett.*, **98**, p. 190504, arXiv:quant-ph/0610082, (2007).

[26] P. Aliferis, F. Brito, D. P. DiVincenzo, J. Preskill, M. Steffen, and B. M. Terhal, “Fault-tolerant computing with biased-noise superconducting qubits: A case study,” *New J. Phys.*, **11**, 1, p. 013061, arXiv:0806.0383, (2009).

[27] M. D. Shulman, O. E. Dial, S. P. Harvey, H. Bluhm, V. Umansky, and A. Yacoby, “Demonstration of entanglement of electrostatically coupled singlet-triplet qubits,” *Science*, **336**, 6078, pp. 202–205, arXiv:1202.1828, (2012).

[28] D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, “Quantum computations on a topologically encoded qubit,” *Science*, **345**, pp. 302–305, arXiv:1403.5426, (2014).

[29] L. Ioffe and M. Mézard, “Asymmetric quantum error-correcting codes,” *Phys. Rev. A*, **75**, p. 032345, arXiv:quant-ph/0606107, (2007).

[30] P. K. Sarvepalli, A. Klappenecker, and M. Rötteler, “Asymmetric quantum codes: constructions, bounds and performance,” *Proc. R. Soc. A*, **465**, 2105, pp. 1645–1672, (2009).

[31] G. G. La Guardia, “On the construction of asymmetric quantum codes,” *Int. J. Theor. Phys.*, **53**, pp. 2312–2322, (2014).

[32] A. Robertson, C. Granade, S. D. Bartlett, and S. T. Flammia, “Tailored Codes for Small Quantum Memories,” *Phys. Rev. Applied*, **8**, p. 064004, arXiv:1703.08179, (2017).

[33] P. Aliferis and J. Preskill, “Fault-tolerant quantum computation against biased noise,” *Phys. Rev. A*, **78**, p. 052331, arXiv:0710.1301, (2008).

[34] A. M. Stephens, W. J. Munro, and K. Nemoto, “High-threshold topological quantum error correction against biased noise,” *Phys. Rev. A*, **88**, p. 060301, arXiv:1308.4776, (2013).

[35] A. M. Stephens, Z. W. E. Evans, S. J. Devitt, and L. C. L. Hollenberg, “Asymmetric quantum error correction via code conversion,” *Phys. Rev. A*, **77**, p. 062335, arXiv:0708.3969, (2008).

[36] J. Napp and J. Preskill, “Optimal Bacon-Shor codes,” *Quantum Inf. Comput.*, **13**, pp. 0490–0510, arXiv:1209.0794, (2013).

[37] P. Brooks and J. Preskill, “Fault-tolerant quantum computation with asymmetric Bacon-Shor codes,” *Phys. Rev. A*, **87**, p. 032310, arXiv:1211.1400, (2013).

[38] M. Li, D. Miller, M. Newman, Y. Wu, and K. R. Brown, “2D compass codes,” *Phys. Rev. X*, **9**, p. 021041, arXiv:1809.01193, (2019).

[39] B. Röthlisberger, J. R. Wootton, R. M. Heath, J. K. Pachos, and D. Loss, “Incoherent dynamics in the toric code subject to disorder,” *Phys. Rev. A*, **85**, p. 022313, arXiv:1112.1613, (2012).

[40] X. Xu, Q. Zhao, X. Yuan, and S. C. Benjamin, “A high threshold code for modular hardware with asymmetric noise,” arXiv:1812.01505, (2018).

[41] N. S. Yanofsky and M. A. Mannucci, Quantum Computing for Computer Scientists. Cambridge University Press, 1st ed., (2008).

[42] P. A. M. Dirac, "A new notation for quantum mechanics," *Math. Proc. Camb. Philos. Soc.*, **35**, 3, p. 416–418, (1939).

[43] M. Born, "Zur quantenmechanik der stoßvorgänge," *Zeitschrift für Physik*, **37**, pp. 863–867, (1926).

[44] D. Gottesman, "The Heisenberg representation of quantum computers," in *Group theoretical methods in physics. Proceedings, 22nd International Colloquium, Group22, ICGTMP'98, Hobart, Australia, July 13-17, 1998*, pp. 32–43, (1998), arXiv:quant-ph/9807006.

[45] W. F. Stinespring, "Positive functions on C*-algebras," *Proc. Am. Math. Soc.*, **6**, 2, pp. 211–216, (1955).

[46] K. Kraus, A. Böhm, J. D. Dollard, and W. H. Wootters, States, Effects, and Operations Fundamental Notions of Quantum Theory, 190. (1983).

[47] P. Webster, S. D. Bartlett, and D. Poulin, "Reducing the overhead for quantum computation when noise is biased," *Phys. Rev. A*, **92**, p. 062309, arXiv:1509.05032, (2015).

[48] S. Loepp and W. K. Wootters, Protecting Information: From Classical Error Correction to Quantum Cryptography. Cambridge University Press, (2006).

[49] D. A. Lidar and T. A. Brun, eds., Quantum Error Correction. Cambridge University Press, (2013).

[50] J. L. Park, "The concept of transition in quantum mechanics," *Found. Phys.*, **1**, 1, pp. 23–33, (1970).

[51] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, **299**, 5886, pp. 802–803, (1982).

[52] D. Dieks, "Communication by EPR devices," *Phys. Lett. A*, **92**, 6, pp. 271 – 272, (1982).

[53] D. Gottesman, "Quantum error correction and fault tolerance - CSSQI 2012," (2012). `https://youtu.be/ltJ1jXQeDl8?t=1485`.

[54] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," in *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, 68, pp. 13–58, (2010), arXiv:0904.2557.

[55] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction," *Phys. Rev. A*, **54**, pp. 3824–3851, arXiv:quant-ph/9604024, (1996).

[56] E. Knill and R. Laflamme, "Theory of quantum error-correcting codes," *Phys. Rev. A*, **55**, pp. 900–911, arXiv:quant-ph/9604034, (1997).

[57] D. Kribs, R. Laflamme, and D. Poulin, "Unified and generalized approach to quantum error correction," *Phys. Rev. Lett.*, **94**, p. 180501, arXiv:quant-ph/0412076, (2005).

[58] M. A. Nielsen and D. Poulin, "Algebraic and information-theoretic conditions for operator quantum error correction," *Phys. Rev. A*, **75**, p. 064304, arXiv:quant-ph/0506069, (2007).

[59] D. Gottesman, "Stabilizer codes and quantum error correction," *Ph.D. Thesis (CalTech)*, arXiv:quant-ph/9705052, (1997).

[60] D. W. Kribs, R. Laflamme, D. Poulin, and M. Lesosky, "Operator quantum error correction," *Quantum Info. Comput.*, **6**, pp. 382–399, arXiv:quant-ph/0504189, (2006).

[61] D. Poulin, "Stabilizer formalism for operator quantum error correction," *Phys. Rev. Lett.*, **95**, p. 230504, arXiv:quant-ph/0508131, (2005).

[62] H. Bombín, "Single-shot fault-tolerant quantum error correction," *Phys. Rev. X*, **5**, p. 031043, arXiv:1404.5504, (2015).

[63] B. Eastin and E. Knill, "Restrictions on transversal encoded quantum gate sets," *Phys. Rev. Lett.*, **102**, p. 110502, arXiv:0811.4262, (2009).

[64] S. Bravyi and R. König, "Classification of topologically protected gates for local stabilizer codes," *Phys. Rev. Lett.*, **110**, p. 170503, arXiv:1206.1609, (2013).

[65] H. Bombín, "Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes," *New J. Phys.*, **17**, p. 083002, arXiv:1311.0879, (2015).

[66] H. Bombín, "Dimensional jump in quantum error correction," *New J. Phys.*, **18**, p. 043038, arXiv:1412.5079, (2016).

[67] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery," *New J. Phys.*, **14**, p. 123011, arXiv:1111.4022, (2012).

[68] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, "Poking holes and cutting corners to achieve Clifford gates with the surface code," *Phys. Rev. X*, **7**, p. 021029, arXiv:1609.04673, (2017).

[69] B. J. Brown, "A fault-tolerant non-Clifford gate for the surface code in two dimensions," arXiv:1903.11634, (2019).

[70] S. Bravyi and A. Kitaev, "Universal quantum computation with ideal Clifford gates and noisy ancillas," *Phys. Rev. A*, **71**, p. 022316, arXiv:quant-ph/0403025, (2005).

[71] H. Bombin and M. A. Martin-Delgado, "Topological quantum distillation," *Phys. Rev. Lett.*, **97**, p. 180501, arXiv:quant-ph/0605138, (2006).

[72] N. Delfosse and J. Tillich, "A decoding algorithm for CSS codes using the X/Z correlations," in *2014 IEEE International Symposium on Information Theory*, pp. 1071–1075, (2014), arXiv:1401.6975.

[73] A. G. Fowler, "Optimal complexity correction of correlated errors in the surface code," arXiv:1310.0863, (2013).

[74] J. Edmonds, "Paths, trees and flowers," *Canad. J. Math.*, **17**, p. 449, (1965).

[75] A. G. Fowler, "Minimum weight perfect matching of fault-tolerant topological quantum error correction in average O(1) parallel time," *Quantum Info. Comput.*, **15**, pp. 145–158, arXiv:1307.1740, (2015).

[76] J. C. Bridgeman and C. T. Chubb, "Hand-waving and interpretive dance: an introductory course on tensor networks," *J. Phys. A*, **50**, p. 223001, arXiv:1603.03039, (2017).

[77] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Phys. Rev. A*, **90**, p. 032326, arXiv:1405.4883, (2014).

[78] F. Verstraete and J. I. Cirac, "Renormalization algorithms for quantum-many body systems in two and higher dimensions," (2004), arXiv:cond-mat/0407066.

[79] V. Murg, F. Verstraete, and J. I. Cirac, "Variational study of hard-core bosons in a two-dimensional optical lattice using projected entangled pair states," *Phys. Rev. A*, **75**, p. 033605, arXiv:cond-mat/0611522, (2007).

[80] S. Bravyi, D. Poulin, and B. Terhal, "Tradeoffs for reliable quantum information storage in 2D systems," *Phys. Rev. Lett.*, **104**, p. 050503, arXiv:0909.5200, (2010).

[81] C. T. Chubb and S. T. Flammia, "Statistical mechanical models for quantum codes with correlated noise," arXiv:1809.10704, (2018).

[82] A. Kubica, B. Yoshida, and F. Pastawski, "Unfolding the color code," *New J. Phys.*, **17**, p. 083026, arXiv:1503.02065, (2015).

[83] R. Alicki, M. Horodecki, P. Horodecki, and R. Horodecki, "On thermal stability of topological qubit in Kitaev's 4D model," *Open Syst. Inf. Dyn.*, **17**, 01, pp. 1–20, arXiv:0811.0033, (2010).

[84] H. Bombin, R. W. Chhajlany, M. Horodecki, and M. A. Martin-Delgado, "Self-correcting quantum computers," *New J. Phys.*, **15**, p. 055023, arXiv:0907.5228, (2013).

[85] S. Bravyi, "Subsystem codes with spatially local generators," *Phys. Rev. A*, **83**, p. 012320, arXiv:1008.1029, (2011).

[86] T. J. Yoder, "Optimal quantum subsystem codes in two dimensions," *Phys. Rev. A*, **99**, p. 052333, arXiv:1901.06319, (2019).

[87] J. Tillich and G. Zémor, "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Trans. Inf. Theory*, **60**, pp. 1193–1202, arXiv:0903.0566, (2014).

[88] A. A. Kovalev and L. P. Pryadko, "Improved quantum hypergraph-product LDPC codes," in *2012 IEEE International Symposium on Information Theory Proceedings*, pp. 348–352, (2012), arXiv:1202.0928.

[89] M. H. Freedman and M. B. Hastings, "Quantum systems on non-k-hyperfinite complexes: A generalization of classical statistical mechanics on expander graphs," *Quantum Info. Comput.*, **14**, pp. 144–180, arXiv:1301.1363, (2014).

[90] M. B. Hastings, "Decoding in hyperbolic spaces: Quantum LDPCs codes with linear rate and efficient error correction," *Quantum Info. Comput.*, **14**, pp. 1187–1202, arXiv:1312.2546, (2014).

[91] D. Gottesman, "Fault-tolerant quantum computation with constant overhead," *Quantum Info. Comput.*, **14**, pp. 1338–1372, arXiv:1310.2984, (2014).

[92] D. K. Tuckett, "Qecsim." `https://davidtuckett.com/qit/qecsim/`, (2019).

[93] G. van Rossum and F. L. Drake, The Python Language Reference Manual. Network Theory Ltd., (2011).

[94] T. E. Oliphant, A guide to NumPy, 1. Trelgol Publishing USA, (2006). `https://www.numpy.org/`.

[95] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," (2001–). `https://www.scipy.org/`.

[96] B. Stroustrup, The C++ Programming Language. Addison-Wesley Professional, 4th ed., (2013).

2 Ultrahigh error threshold for surface codes with biased noise

DOI: 10.1103/PhysRevLett.120.050505, arXiv: 1708.08474

© 2018 American Physical Society

Preamble

In this chapter, we consider surface code thresholds in an idealized context with reliable measurements. Previous studies of surface code thresholds had focused on pure X (bit-flip) or Z (phase-flip) noise and depolarizing noise. We discover that the threshold of the surface code with pure Y noise is much higher. Tailoring the code, with a simple modification, we demonstrate that ultrahigh thresholds can be achieved with the dephasing or Z-biased noise that is prevalent in many experimental setups.

Abstract

We show that a simple modification of the surface code can exhibit an enormous gain in the error correction threshold for a noise model in which Pauli Z errors occur more frequently than X or Y errors. Such biased noise, where dephasing dominates, is ubiquitous in many quantum architectures. In the limit of pure dephasing noise we find a threshold of 43.7(1)% using a tensor network decoder proposed by Bravyi, Suchara and Vargo. The threshold remains surprisingly large in the regime of realistic noise bias ratios, for example 28.2(2)% at a bias of 10. The performance is, in fact, at or near the hashing bound for all values of the bias. The modified surface code still uses only weight-4 stabilizers on a square lattice, but merely requires measuring products of Y instead of Z around the faces, as this doubles the number of useful syndrome bits associated with the dominant Z errors. Our results demonstrate that large efficiency gains can be found by appropriately tailoring codes and decoders to realistic noise models, even under the locality constraints of topological codes.

2.1 Introduction

For quantum computing to be possible, fragile quantum information must be protected from errors by encoding it in a suitable quantum error correcting code. The surface code [1] (and related topological stabilizer codes [2]) are quite remarkable among the diverse range of quantum error correcting codes in their ability to protect quantum information against local noise. Topological codes can have surprisingly large *error thresholds*—the break-even error rate below which errors can be corrected with arbitrarily high probability—despite using stabilizers that

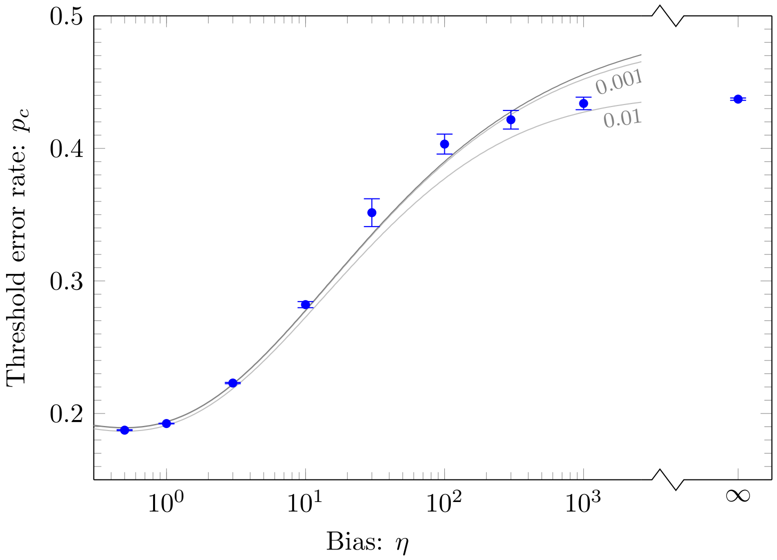


Figure 2.1: Threshold error rate p_c as a function of bias η. The dark gray line is the zero-rate hashing bound for the associated Pauli error channel. Lighter gray lines show the hashing bound for rates $R = 0.001$ and 0.01 for comparison; the surface code family has rate $1/n$ for n qubits. Blue points show the estimates for the threshold using the fitting procedure described in the main text together with 1-standard-deviation error bars. The point at the largest bias value corresponds to infinite bias, i.e., only Z errors.

act on only a small number of neighboring qubits [3]. It is the combination of these high error thresholds and local stabilizers that make topological codes, and the surface code in particular, popular choices for many quantum computing architectures.

Here we demonstrate a significant increase in the error threshold for a surface code when the noise is *biased*, i.e., when one Pauli error occurs at a higher rate than others. For qubits defined by nondegenerate energy levels with a Hamiltonian proportional to Z, the noise model is typically described by a dephasing (Z-error) rate that is much greater than the rates for relaxation and other energy-nonpreserving errors. Such biased noise is common in many quantum architectures, including superconducting qubits [4], quantum dots [5], and trapped ions [6], among others. The increased error threshold is achieved by tailoring the standard surface code stabilizers to the noise in an extremely simple way and by employing a decoder that accounts for correlations in the error syndrome. In particular, using the tensor network decoder of Bravyi, Suchara and Vargo (BSV) [7], we give evidence that the error correction threshold of this tailored surface code with pure Z noise is $p_c = 43.7(1)\%$, a fourfold increase over the optimal surface code threshold for pure Z noise of 10.9% [7].

These gains result from the following simple observations. For a Z error in the standard formulation of the surface code, the stabilizers consisting of products of Z around each plaquette of the square lattice contribute no useful syndrome information. Exchanging these Z-type stabilizers with products of Y around each plaquette still results in a valid quantum surface code, since these Y-type stabilizers will commute with the original X-type stabilizers. But now there are twice as many bits of syndrome information about the Z errors. Taking advantage of these extra syndrome bits requires an optimized decoder that can use the correlations between the two syndrome types. The standard decoder based on minimum-weight matching breaks down at this point, but the BSV decoder is specifically designed to handle such correlations. We show that the parameter χ, which defines the scale of correlation in the BSV decoder, needs to be large to achieve optimal decoding, so in that sense accounting for these correlations is actually necessary. These two ideas—doubling the number of useful syndrome bits and a decoder that makes optimal use of them—give an intuition that captures the essential reason for the increased threshold. It is nonetheless remarkable just how large an effect this simple change makes.

We also consider more general Pauli error models, where Z errors occur more frequently than X and Y errors with a nonzero bias ratio of the error rates. We show that the tailored surface code exhibits these significant gains in the error threshold even for modest error biases in physically relevant regimes: for biases of 10 (meaning dephasing errors occur 10 times more frequently than all other errors), the error threshold is already 28.2(2)%. Figure 2.1 presents our main result of the threshold scaling as a function of bias. Notably, we find that the tailored surface code together with the BSV decoder performs near the hashing bound for all values of the bias.

2.2 Error correction with the surface code

The surface code [1] is defined by a 2D square lattice having qubits on the edges with a set of local stabilizer generators. In the usual prescription, for each vertex (or plaquette), the stabilizer consists of the product of the X (or Z) operators acting on the neighboring edges. We simply exchange the roles of Z and Y, as shown in Fig. 2.2. By choosing appropriate "rough" and "smooth" boundary conditions along the vertical and horizontal edges, the code space encodes one logical qubit into the joint $+1$ eigenspace of all the commuting stabilizers with a code distance d given by the linear size of the lattice.

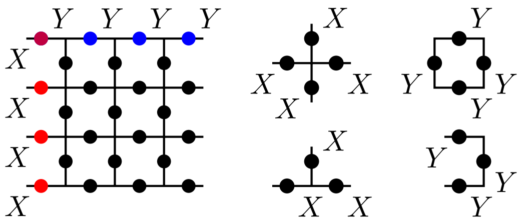


Figure 2.2: The modified surface code, tailored for biased Z noise, with logical operators given by a product of Y along the top edge and a product of X along the left edge. The stabilizers are shown at right.

A large effort has been devoted to understanding error correction of the surface code and the closely-related toric code [8]. The majority of this effort has focused on the cases of either pure Z noise, or depolarizing noise where X, Y, and Z errors happen with equal probability; see Refs. [2, 9] for recent literature reviews. Once a noise model is fixed, one must define a decoder, and the most popular choice is based on minimum-weight matching (MWM). This decoder treats X and Z noise independently, and it has an error threshold of around 10.3% for pure Z noise with a naive implementation [3, 10], or 10.6% with some further optimization [11]. Many other decoders have been proposed, however, and these are judged according to their various strengths and weaknesses, including the threshold error rate, the logical failure rate below threshold, robustness to measurement errors (fault tolerance), speed, and parallelizability. Of particular note are the decoders of Refs. [12–20], since these either can handle, or can be modified to handle, correlations beyond the paradigm of independent X and Z errors.

2.3 The BSV decoder

Our choice of the BSV decoder [7] is motivated by the fact that it gives an efficient approximation to the optimal maximum likelihood (ML) decoder, which maximizes the *a posteriori* probability of a given logical error conditioned on an observed syndrome. This decoder has also previously been used to do nearly optimal decoding of depolarizing noise [7], achieving an error threshold close to estimates from statistical physics arguments that the threshold should be 18.9% [21]. [In fact, our own estimate of the depolarizing threshold using the BSV decoder is 18.7(1)%.] Because it approximates the ML decoder, the BSV decoder is a natural choice for finding the maximum value of the threshold for biased noise models.

The decoder works by defining a tensor network with local tensors associated with the qubits and stabilizers of the code. The geometry of the tensor network respects the geometry of the code. Each index on the local tensors has dimension 2 initially, but during the contraction sequence, this dimension grows until it is bounded by χ, called the bond dimension. When χ is exponentially large in n, the number of physical qubits, then the contraction value of the tensor network returns the exact probabilities conditioned on the syndrome of each of the four logical error classes. Such an implementation would be highly inefficient, but using a truncation procedure during the tensor contraction allows one to work with any fixed value of $\chi \geq 2$ with a polynomial runtime of $O(n\chi^3)$. In this way, the algorithm provides an efficient and tunable approximation of the exact ML decoder, and in practice small values of χ were observed to work well [7]. We refer the reader to Ref. [7] for the full details of this decoder.

2.4 Biased Pauli error model

A Pauli error channel is defined by an array $\boldsymbol{p} = (1 - p, p_x, p_y, p_z)$ corresponding to the probabilities for each Pauli operator I (no error), X, Y, and Z, respectively. We define $p = p_x + p_y + p_z$ to be the probability of any single-qubit error, and we always consider the case of independent, identically distributed noise. We define the bias η to be the ratio of the probability of a Z error occurring to the total probability of a non-Z Pauli error occurring, so that $\eta = p_z/(p_x + p_y)$. For simplicity, we consider the special case $p_x = p_y$ in what follows. Then for total error probability p, Z errors occur with probability $p_z = [\eta/(\eta + 1)]p$, and $p_x = p_y = [1/2(\eta + 1)]p$. When $\eta = 1/2$, this gives the standard depolarizing channel with probability $p/3$ for each nontrivial Pauli error, and taking the limit $\eta \to \infty$ gives only Z errors with probability p. Biased Pauli error models have been considered by a number of authors [4, 22–27], but we note that there are several different conventions for the definition of bias. Comparison between channels with different bias but the same total error rate is facilitated by the fact that the channel fidelity to the identity is a function only of p.

2.5 Hashing bound

The quantum capacity is the maximum achievable rate at which one can transmit quantum information through a noisy channel [28]. The hashing bound [29–31] is an achievable rate which is generally less than the quantum capacity [32]. For Pauli error channels, the hashing bound takes a particularly simple form [28] and says that there exist quantum stabilizer codes that achieve a rate $R = 1 - H(\boldsymbol{p})$, with H being the Shannon entropy. The proof of achievability involves using random codes, and it is generally hard to find explicit codes and decoders that perform at or above this rate for an arbitrary channel, especially if one wishes to impose additional constraints such as local stabilizers. The quantum capacity itself is still unknown for any Pauli channel where at least two of (p_x, p_y, p_z) are nonzero.

2.6 Numerics

Our numerical implementation makes only a minor modification to the BSV decoder. To avoid changing the definitions of the tensors used in Ref. [7], we use the symmetry by which we can exchange the role of Z noise in the modified surface code with the role of Y noise in the standard surface code. Then all of the definitions in Ref. [7] carry over unchanged. The only difference is that we perform two tensor network contractions for each decoding sequence. There is an arbitrary choice as to whether to contract the network row-wise or column-wise. Rather than pick just one, we average the values of both contractions. We empirically observe improved performance with this modification.

For each value of the bias $\eta \in \{0.5,\ 1,\ 3,\ 10,\ 30,\ 100,\ 300,\ 1000,\ \infty\}$, we estimate the logical failure rate f using the BSV decoder to obtain the sample mean failure rate on 30 000 random trials for a selection of physical error rates p in the region near the threshold p_c for code distances $d \in \{9,\ 13,\ 17,\ 21\}$. We use a rather large value of the bond dimension χ for our simulations, specifically $\chi = 48$, although for bias $\eta < 30$ we already observe that the decoder converges well with $\chi = 36$. However, we still do not observe complete convergence of the decoder at $\chi = 48$ in the regime of intermediate bias around $\eta = 100$. The decoder convergence with χ is displayed in Fig. 2.4, which shows the estimate of the logical failure rate for the $d = 21$ code near the threshold. Performance of the decoder and convergence with χ generally improve as bias increases again beyond $\eta = 300$, but it is likely that further improvements are possible in the intermediate bias regime. Although the decoder at $\chi = 48$ is not achieving an optimal failure rate in the intermediate regime, we see excellent convergence for most of the range of bias and across the full range of bias we observe threshold behavior. Moreover, this threshold is at the hashing bound for all $\eta \leq 100$. In the regions that are a fixed distance below the threshold, as in Fig. 2.3, we observe an exponential decay in the logical failure rate $f \sim \exp(-\alpha d)$, where α may depend on the bias and is an increasing function of $(p_c - p)$. This constitutes strong evidence of an error correction threshold.

We note that $\chi = 48$ was the largest used in our simulations, so we do not know if the saturation of the decoder performance for bias $\eta \geq 300$ is a real effect, or a side effect of having too small a value of χ. Although we observe convergence and threshold behavior, we do not know how much the performance might improve for larger values of χ since, as seen in Fig. 2.4, there is apparently still some room for improvement. It is possible that the saturation is a real effect, however, since even at infinite bias there are still logical errors of weight $2d = O(\sqrt{n})$ that consist only of Z errors. This is in contrast to the classical repetition code, which has a threshold of 50% and a distance $O(n)$. One possibility to address this is to use a surface code with side lengths $L \times W$, where L and W are relatively prime, for example just choosing $W = L + 1$. We empirically observe that the Z-distance (i.e., the distance when restricted only to Z errors) of the code scales like $O(n)$ for this modification of the surface code. In

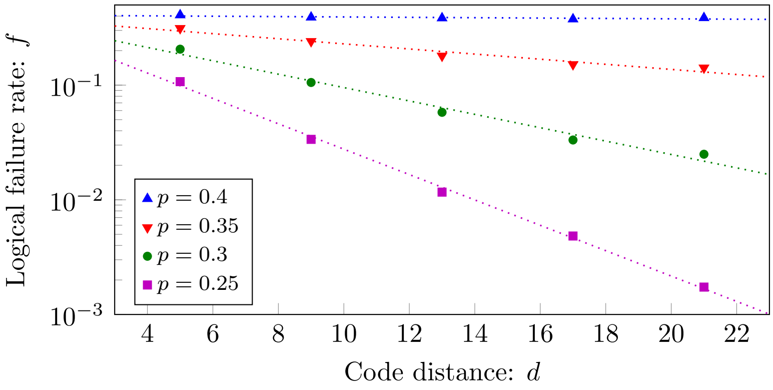


Figure 2.3: Exponential decay of the logical failure rate f with respect to code distance d in the regime $p < p_c$ for $\eta = 100$ and $\chi = 48$. We observe scaling behavior of the form $f \sim \exp(-\alpha d)$ where α depends on the bias and is an increasing function of $(p_c - p)$. In this bias regime, the decoder performance is likely farthest from optimal, but the decay is still clearly exponential over this range. Other values of η show the same general scaling behavior, though with different decay rates α. The statistical error bars from 30 000 trials per point are smaller than the individual plot points in every case.

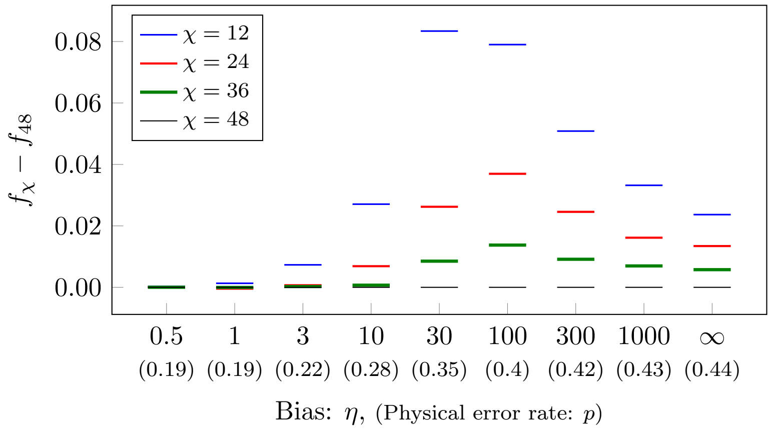


Figure 2.4: Convergence of the decoder as a function of χ near the threshold for distance $d = 21$. We observe that the logical failure rates f_χ stabilize with increasing χ for both low and high biases. However, in the intermediate bias regime f_χ is still decreasing noticeably between increments of χ, suggesting that $\chi > 48$ would be required for a good approximation to the optimal ML decoder.

fact, on a toric code with L and W both odd and relatively prime, the Z-distance is provably $O(n)$ [33]. These observations are currently being explored, and will be addressed in more detail in forthcoming work[1].

To obtain an explicit estimate of the threshold p_c, we use the critical exponent method of Ref. [10]. If we define a correlation length $\xi = (p - p_c)^{-\nu}$ for some critical exponent ν, then in the regime where $d \gg \xi$ we expect that the behavior of the code is scale invariant. In this regime, since the code distance d corresponds to a physical length, the failure probability should depend only on the dimensionless ratio d/ξ, a conjecture that was first empirically verified in Ref. [10]. This suggests defining a rescaled variable $x = (d/\xi)^{1/\nu} = (p - p_c)d^{1/\nu}$ so that the failure rate expanded as a power series in x is explicitly scale invariant at the critical

[1] The *forthcoming work* is included in Chapter 3.

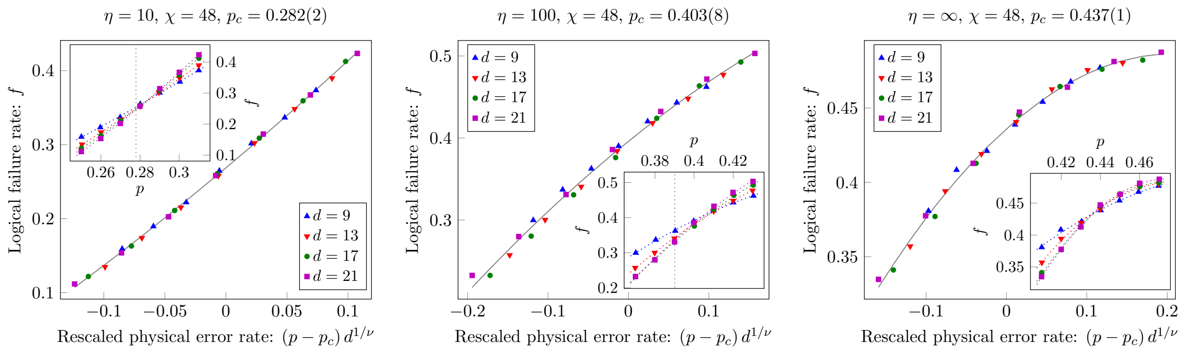


Figure 2.5: Logical failure rate f as a function of the rescaled error rate $x = (p - p_c)d^{1/\nu}$ for biases $\eta \in \{10,\ 100,\ \infty\}$. The solid line is the best fit to the model $f = A + Bx + Cx^2$. The insets show the raw sample means over 30 000 runs for various values of p, and the dotted gray vertical line indicates the hashing bound. Even for the case of $\eta = 100$ where the decoder performance was likely furthest from optimal we still see good agreement with the fit model.

point p_c corresponding to $x = 0$. It is then natural to consider a model for the failure rate given by a truncated Taylor expansion in the neighborhood around p_c. We use a quadratic model, $f = A + Bx + Cx^2$, and then fit to this model to find p_c, ν and the nuisance parameters A, B, C. A discussion on the limits of the validity of this universal scaling hypothesis can be found in Ref. [34]. We plot our estimates of f for various values of p and d for the representative cases of $\eta \in \{10,\ 100,\ \infty\}$ in Fig. 2.5 together with rescaled data as a function of x. A visual inspection confirms good qualitative agreement with the model.

The critical exponents method gives precise estimates of p_c with low statistical uncertainty. However, systematic biases might affect the accuracy of the estimate and must be accounted for. Finite-size effects typically cause threshold estimates to decrease as larger and larger code distances are added to the estimate. Additionally, the suboptimality of the decoder due to small χ values in the intermediate bias regime may have overestimated each individual logical failure rate. This latter effect does not directly imply that we have also overestimated the threshold p_c, and the data remain consistent with the fit model in spite of this as can be seen in Fig. 2.5. On balance, we expect that our estimates might decrease somewhat in the intermediate bias regime. Our final error bars were obtained by jackknife resampling, i.e. by computing, for each fixed η, the spread in estimates for p_c when rerunning the fit procedure with a single distance d removed, for each choice of d. Our results are summarized in Fig. 2.1.

2.7 Fault tolerant syndrome extraction

Our study has focused on the error correction threshold under the assumption of ideal syndrome extraction. To see if the gains observed in this setting carry over to applications in fault-tolerant quantum computing, one would need to consider the effects of faulty syndrome measurements and gates. A full fault-tolerant analysis is beyond the scope of this work, but we briefly consider the key issues here.

First, the BSV decoder that we have used to investigate this ultrahigh error threshold is not fault tolerant, but some clustering decoders are [13]. Developing efficient, practical fault-tolerant decoders with the highest achievable thresholds remains a significant challenge for the field.

An added complication with a biased noise model is that the gates that perform the syndrome extraction must at least approximately preserve the noise bias in order to maintain an advantage [4]. For the tailored surface code studied here, one could appeal to the techniques of Refs. [4, 25], where we note that Y-type syndromes can be measured using a minor modification of the X-syndrome measurement scheme. We note that these syndrome extraction circuits are significantly more complex (involving the use of both ancilla cat states and gate teleportation) compared with the standard approach for the surface code with unbiased noise, and this added complexity will undoubtedly reduce the threshold.

More optimistically, we note that the standard method for syndrome extraction in the surface code [35] can be directly adapted to this tailored code and maintains biased noise on the data qubits. Ancilla qubits are placed in the centers of both the plaquette and vertex stabilizers of Fig. 2.2, and they will be both initialized and measured in the X basis. Sequences of controlled-X (vertex) and controlled-Y (plaquette) gates, with the ancilla as the control and data qubits as the target, yield the required syndrome measurements analogous to the standard method. In this scheme, we note that high-rate Z errors on the ancilla are never mapped to the data qubits; low-rate X and Y errors on the ancilla can cause errors on the data qubits but the noise remains biased. Measurement errors will occur at the high rate, but this can be accommodated by repeated measurement. Note that, as argued by Aliferis and Preskill [4], native controlled-X and controlled-Y gates are perhaps not well motivated in a system with a noise bias, but nonetheless this simple scheme illustrates that, in principle, syndromes can be extracted in this code while preserving the noise bias. To develop a full fault-tolerant syndrome extraction circuit in a noise-biased system would require a complete specification of the native gates in the system and an understanding of their associated noise models.

2.8 Discussion

Our numerical results strongly suggest that in systems that exhibit an error bias, there are significant gains to be had for quantum error correction with codes and decoders that are tailored to exploit this bias. It is remarkable that the tailored surface code performs at the hashing bound across a large range of biases. This means that it is not just a good code for a particular error model, but broadly good for any local Pauli error channel once it is tailored to the specific noise bias. It is also remarkable that a topological code, limited to local stabilizers, does so well in this regard.

Many realizations of qubits based on nondegenerate energy levels of some quantum system have a bias—often quite significant—towards dephasing (Z errors) relative to energy-nonconserving errors (X and Y errors). This suggests tailoring other codes, and in particular other topological codes, to have error syndromes generated by X- and Y-type stabilizers. Even larger gains might be had by considering biased noise in qudit surface codes [36, 37].

For qubit topological stabilizer codes, the threshold for exact ML decoding with general Pauli noise can be determined using the techniques of Ref. [21], which mapped the ML decoder's threshold to a phase transition in a pair of coupled random-bond Ising models. It would be interesting to explore this phase boundary for general Pauli noise beyond the depolarizing channel that was studied numerically in Ref. [21].

We have employed the BSV decoder to obtain our threshold estimates because of its near-optimal performance, but it is not the most efficient or practical decoder for many purposes. One outstanding challenge is to find good practical decoders that can work as well or nearly as well across a range of biases. The clustering-type decoders [12, 13] appear well suited for this task, and they have the added advantage that some versions of these decoders (e.g., Ref. [38]) generalize naturally to all Abelian anyon models such as the qudit surface codes.

The most pressing open question related to this work is whether the substantial gains observed here can be preserved in the context of fault-tolerant quantum computing.

Acknowledgements

This work is supported by the Australian Research Council (ARC) via Centre of Excellence in Engineered Quantum Systems (EQuS) Project No. CE110001013 and Future Fellowship No. FT130101744, by the U.S. Army Research Office Grants No. W911NF-14-1-0098 and No. W911NF-14-1-0103, and by the Sydney Informatics Hub for access to high-performance computing resources.

References

[1] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," arXiv:quant-ph/9811052, (1998).

[2] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, **87**, p. 307, arXiv:1302.3428, (2015).

[3] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys. (N.Y.)*, **43**, 9, pp. 4452–4505, arXiv:quant-ph/0110143, (2002).

[4] P. Aliferis, F. Brito, D. P. DiVincenzo, J. Preskill, M. Steffen, and B. M. Terhal, "Fault-tolerant computing with biased-noise superconducting qubits: A case study," *New J. Phys.*, **11**, 1, p. 013061, arXiv:0806.0383, (2009).

[5] M. D. Shulman, O. E. Dial, S. P. Harvey, H. Bluhm, V. Umansky, and A. Yacoby, "Demonstration of entanglement of electrostatically coupled singlet-triplet qubits," *Science*, **336**, pp. 202–205, arXiv:1202.1828, (2012).

[6] D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, "Quantum computations on a topologically encoded qubit," *Science*, **345**, 6194, pp. 302–305, arXiv:1403.5426, (2014).

[7] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Phys. Rev. A*, **90**, p. 032326, arXiv:1405.4883, (2014).

[8] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys. (Amsterdam)*, **303**, 1, pp. 2–30, arXiv:quant-ph/9707021, (2003).

[9] B. J. Brown, D. Loss, J. K. Pachos, C. N. Self, and J. R. Wootton, "Quantum memories at finite temperature," *Rev. Mod. Phys.*, **88**, p. 045005, arXiv:1411.6643, (2016).

[10] C. Wang, J. Harrington, and J. Preskill, "Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory," *Ann. Phys. (Amsterdam)*, **303**, pp. 31–58, arXiv:quant-ph/0207088, (2003).

[11] T. M. Stace and S. D. Barrett, "Error correction and degeneracy in surface codes suffering loss," *Phys. Rev. A*, **81**, p. 022317, arXiv:0912.1159, (2010).

[12] G. Duclos-Cianci and D. Poulin, "Fast Decoders for Topological Quantum Codes," *Phys. Rev. Lett.*, **104**, p. 050504, arXiv:0911.0581, (2010).

[13] G. Duclos-Cianci and D. Poulin, "Fault-tolerant renormalization group decoder for Abelian topological codes," *Quantum Inf. Comput.*, **14**, pp. 0721–0740, arXiv:1304.6100, (2014).

[14] A. G. Fowler, "Optimal complexity correction of correlated errors in the surface code," arXiv:1310.0863, (2013).

[15] J. R. Wootton and D. Loss, "High Threshold Error Correction for the Surface Code," *Phys. Rev. Lett.*, **109**, 16, p. 160503, arXiv:1202.4316, (2012).

[16] 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*, IEEE, (2014), arXiv:1401.6975.

[17] A. Hutter, J. R. Wootton, and D. Loss, "Efficient Markov chain Monte Carlo algorithm for the surface code," *Phys. Rev. A*, **89**, p. 022326, arXiv:1302.2669, (2014).

[18] G. Torlai and R. G. Melko, "Neural Decoder for Topological Codes," *Phys. Rev. Lett.*, **119**, p. 030501, arXiv:1610.04238, (2017).

[19] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, "Machine-learning-assisted correction of correlated qubit errors in a topological code," arXiv:1705.07855, (2017).

[20] S. Krastanov and L. Jiang, "Deep neural network probabilistic decoder for stabilizer codes," *Sci. Rep.*, **7**, 1, p. 11003, arXiv:1705.09334, (2017).

[21] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, "Strong Resilience of Topological Codes to Depolarization," *Phys. Rev. X*, **2**, p. 021004, arXiv:1202.1852, (2012).

[22] P. Aliferis and J. Preskill, "Fault-tolerant quantum computation against biased noise," *Phys. Rev. A*, **78**, p. 052331, arXiv:0710.1301, (2008).

[23] B. Röthlisberger, J. R. Wootton, R. M. Heath, J. K. Pachos, and D. Loss, "Incoherent dynamics in the toric code subject to disorder," *Phys. Rev. A*, **85**, p. 022313, arXiv:1112.1613, (2012).

[24] J. Napp and J. Preskill, "Optimal Bacon-Shor codes," *Quantum Inf. Comput.*, **13**, pp. 0490–0510, arXiv:1209.0794, (2013).

[25] P. Brooks and J. Preskill, "Fault-tolerant quantum computation with asymmetric Bacon-Shor codes," *Phys. Rev. A*, **87**, p. 032310, arXiv:1211.1400, (2013).

[26] P. Webster, S. D. Bartlett, and D. Poulin, "Reducing the overhead for quantum computation when noise is biased," *Phys. Rev. A*, **92**, p. 062309, arXiv:1509.05032, (2015).

[27] A. Robertson, C. Granade, S. D. Bartlett, and S. T. Flammia, "Tailored Codes for Small Quantum Memories," *Phys. Rev. Applied*, **8**, p. 064004, arXiv:1703.08179, (2017).

[28] M. Wilde, Quantum Information Theory. Cambridge, England: Cambridge University Press, , Cambridge, England(2013), arXiv:1106.1445.

[29] S. Lloyd, "Capacity of the noisy quantum channel," *Phys. Rev. A*, **55**, pp. 1613–1622, arXiv:quant-ph/9604015, (1997).

[30] P. W. Shor, "The quantum channel capacity and coherent information." lecture notes, MSRI Workshop on Quantum Computation, San Francisco, (November 2002).

[31] I. Devetak, "The private classical capacity and quantum capacity of a quantum channel," *IEEE Trans. Inf. Theory*, **51**, p. 44, arXiv:quant-ph/0304127, (2005).

[32] D. P. DiVincenzo, P. W. Shor, and J. A. Smolin, "Quantum-channel capacity of very noisy channels," *Phys. Rev. A*, **57**, pp. 830–839, arXiv:quant-ph/9706061, (1998).

[33] S. Bravyi. private communication, (2017).

[34] F. H. E. Watson and S. D. Barrett, "Logical error rate scaling of the toric code," *New J. Phys.*, **16**, p. 093045, arXiv:1312.5213, (2014).

[35] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, **86**, p. 032324, arXiv:1208.0928, (2012).

[36] H. Anwar, B. J. Brown, E. T. Campbell, and D. E. Browne, "Fast decoders for qudit topological codes," *New J. Phys.*, **16**, p. 063038, arXiv:1311.4895, (2014).

[37] F. H. E. Watson, H. Anwar, and D. E. Browne, "Fast fault-tolerant decoder for qubit and qudit surface codes," *Phys. Rev. A*, **92**, p. 032309, arXiv:1411.3028, (2015).

[38] S. Bravyi and J. Haah, "Quantum Self-Correction in the 3D Cubic Code Model," *Phys. Rev. Lett.*, **111**, 20, p. 200501, arXiv:1112.3252, (2013).

3 Tailoring surface codes for highly biased noise

DOI: 10.1103/PhysRevX.9.041031, arXiv: 1812.08186

🅭 Creative Commons Attribution 4.0 International

Preamble

In this chapter, we reveal the structure of the tailored surface code with pure Z noise that is responsible for the ultrahigh thresholds discovered in Chapter 2. Furthermore, we show that, by modifying the code boundary, this structure can be leveraged to tune the distance and number of failure modes of the code with respect to pure Z noise. Using codes with such modified boundaries, we demonstrate a significant reduction in logical failure rate with Z-biased noise.

Abstract

The surface code, with a simple modification, exhibits ultrahigh error-correction thresholds when the noise is biased toward dephasing. Here, we identify features of the surface code responsible for these ultrahigh thresholds. We provide strong evidence that the threshold error rate of the surface code tracks the hashing bound exactly for all biases and show how to exploit these features to achieve significant improvement in logical failure rate. First, we consider the infinite bias limit, meaning pure dephasing. We prove that the error threshold of the modified surface code for pure dephasing noise is 50%, i.e., that all qubits are fully dephased, and this threshold can be achieved by a polynomial-time decoding algorithm. We demonstrate that the subthreshold behavior of the code depends critically on the precise shape and boundary conditions of the code. That is, for rectangular surface codes with standard rough and smooth open boundaries, it is controlled by the parameter $g = \gcd(j, k)$, where j and k are dimensions of the surface code lattice. We demonstrate a significant improvement in logical failure rate with pure dephasing for *coprime* codes that have $g = 1$, and closely-related *rotated* codes, which have a modified boundary. The effect is dramatic: The same logical failure rate achievable with a square surface code and n physical qubits can be obtained with a coprime or rotated surface code using only $O(\sqrt{n})$ physical qubits. Finally, we use approximate maximum-likelihood decoding to demonstrate that this improvement persists for a general Pauli noise biased toward dephasing. In particular, comparing with a square surface code, we observe a significant improvement in logical failure rate against biased noise using a rotated surface code with approximately half the number of physical qubits.

3.1 Introduction

Quantum error-correcting codes are expected to play a fundamental role in enabling quantum computers to operate at a large scale in the presence of noise. The surface code [1], an example of a topological stabilizer code [2], is one of the most studied and promising candidates, giving excellence performance for error correction while requiring only check operators (stabilizers) acting on a small number of neighboring qubits [3].

The error-correction threshold of a code family, which denotes the physical error rate below which the logical failure rate can be made arbitrarily small by increasing the code size, is strongly dependent on the noise model. The most commonly studied noise model is uniform depolarization of all qubits, where independent single-qubit Pauli X, Y, and Z errors occur at equal rates. However, in many quantum architectures such as certain superconducting qubits [4], quantum dots [5], and trapped ions [6], among others, the noise is biased toward dephasing, meaning that Z errors occur much more frequently than other errors. Recently, it was shown that, with a simple modification, the surface code exhibits ultrahigh thresholds with such Z-biased noise [7], where bias is defined as the ratio of the probability of a high-rate Z error over the probability of a low-rate X or Y error.

In this paper, we identify and characterize the features of the noise-tailored surface code that contribute to its ultrahigh thresholds with Z-biased noise and demonstrate a further significant improvement in logical failure rate. We note that the modification of the surface code, described in Ref. [7], simply exchanges the roles of Z and Y operators in stabilizer and logical operator definitions. Therefore, results for the modified surface code with Z-biased noise can equivalently be expressed in terms of the unmodified surface code and Y-biased noise, where Y errors occur more frequently than X or Z errors. In order to frame our analysis in the context of the familiar unmodified surface code and to simplify comparison with other codes, we consider pure Y noise and Y-biased noise on the surface code, with X- and Z-parity checks, throughout this paper. However, we emphasize that our results apply equally to the modified surface code with pure Z noise or the Z-biased noise prevalent in many quantum architectures.

Our main numerical result is to demonstrate that the threshold error rate of the tailored surface code saturates the hashing bound for all biases. While the numerical results of Ref. [7] indicate that the threshold error rate of the tailored surface code approaches the hashing bound for low to moderate bias, the threshold estimates fall short for higher and infinite bias. Using a tensor-network decoder that converges much more strongly with biased noise, we significantly improve on the results of Ref. [7]. Our new results are summarized in Fig. 3.1, providing strong evidence that the hashing bound can be achieved with a tailored surface code.

Our main analytical result is a structural theorem that reveals a hidden concatenated form of the surface code. We show that, in the limit of pure Y noise, the surface code can be viewed as a classical concatenated code with two concatenation levels. The top level contains the so-called cycle code whose parity checks correspond to cycles in the complete graph. The bottom level contains several copies of the repetition code. We prove that the cycle code has an error threshold of 50% and give an efficient decoding algorithm that achieves this threshold. As a corollary, we show that the threshold of the surface code with pure Y noise is 50%, thus answering an open question posed in Ref. [7]. The concatenated structure described above is controlled by the parameter $g = \gcd(j, k)$, where j and k are dimensions of the surface code lattice. In particular, the top-level cycle code has length $O(g^2)$, while the bottom-level

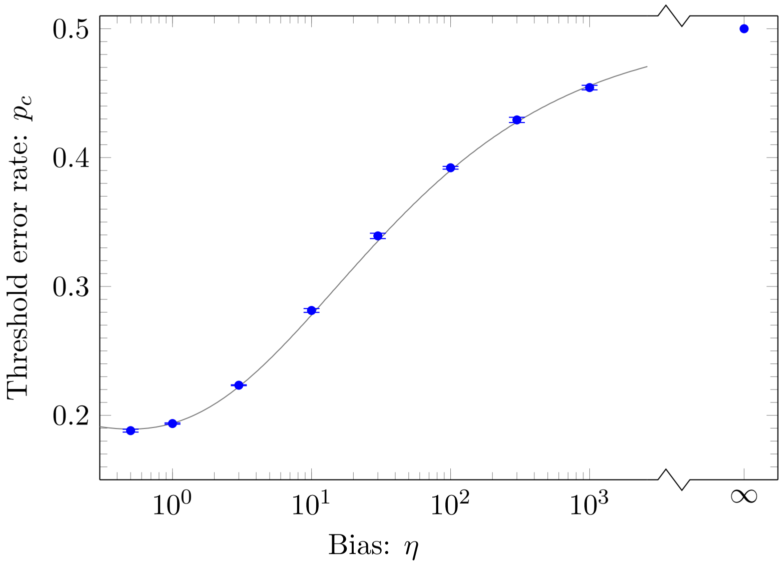


Figure 3.1: Threshold error rate p_c as a function of bias η. Points show threshold estimates for the surface code. Error bars indicate one standard deviation relative to the fitting procedure. The point at the smallest bias corresponds to $\eta = 0.5$ or standard depolarizing noise. The point at infinite bias indicates the analytically proven 50% threshold value. The gray line is the hashing bound for the associated Pauli error channel.

repetition codes have length $O(jk/g^2)$. Two important special cases are *coprime codes* and *square codes* that have $g = 1$ and $g = j = k$, respectively. Informally, a coprime surface code can be viewed as a repetition code, whereas a square surface code can be viewed as a cycle code (in the limit of pure Y noise). We also show that a closely-related family of surface codes called *rotated* codes (defined by boundaries formed at 45° relative to the standard surface code family) can also be seen as repetition codes against pure Y noise. Although the repetition and the cycle codes both have a 50% error threshold, we argue that the former performs much better in the subthreshold regime. This result suggests that coprime and rotated surface codes may have an intrinsic advantage in correcting strongly biased noise.

We present further insights into the origins of the ultrahigh thresholds by investigating the form of logical operators. We show that logical operators consistent with pure Y noise are much rarer and heavier than those consistent with pure X or Z noise, and their structure depends strongly on the parameter g. In particular, there are 2^{g-1} Y-type logical operators of which the minimum weight is $(2g-1)(jk/g^2)$, which compares to $2^{j(k-1)}$ X-type logical operators of which the minimum weight is j. In the case of coprime codes, there is only one Y-type logical operator, and its weight is jk. Hence, the distance of coprime codes to pure Y noise is $O(n)$, whereas for square codes it is $O(\sqrt{n})$. We extend these results to rotated surface codes. We find that rotated codes, with odd linear dimensions, have similar features to coprime codes; in particular, they admit only one Y-type logical operator, and its weight is n. This result is a further improvement over coprime codes, since rotated surface codes are, in a sense, optimal [8]. That is, they achieve the same distance as standard surface codes with approximately half the number of physical qubits.

Leveraging features of the structure of rotated codes with pure Y noise, we develop a tensor-network decoder that achieves much more strongly converged decoding with Y-biased

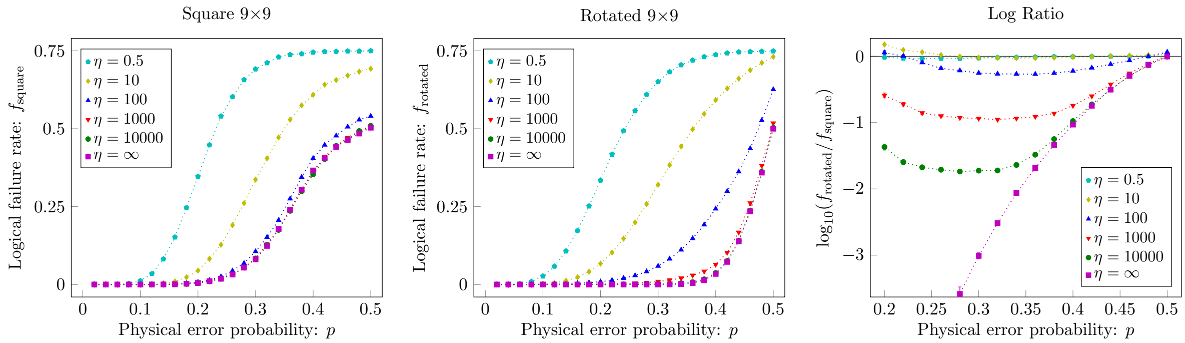


Figure 3.2: Logical failure rates f_{square} and f_{rotated} as a function of physical error probability p for small comparable square and rotated 9×9 codes and the logarithm of the ratio of logical failure rates $\log_{10}(f_{\text{rotated}}/f_{\text{square}})$ with noise biases $\eta \in \{0.5, 10, 100, 1000, 10\,000, \infty\}$. Error bars indicate one standard deviation. Data points are sample means over 30 000 and 1 200 000 runs for the square and rotated codes, respectively, using approximate maximum-likelihood decoding converged to within half a standard deviation for both codes. Dotted lines connect successive data points for a given η.

noise compared with the decoder in Ref. [9] and exact maximum-likelihood decoding in the limit of pure Y noise.

We perform numerical simulations, using exact maximum-likelihood decoding to confirm the 50% threshold for the surface code with pure Y noise and demonstrate a significant reduction in logical failure rate for coprime and rotated codes compared to square codes with pure Y noise. In particular, we demonstrate that the logical failure rate decays exponentially with the distance to pure Y noise such that a target logical failure rate may be achieved with quadratically fewer physical qubits by using coprime or rotated codes compared with standard (square) surface codes.

Finally, we demonstrate a remarkable property of surface codes: By *removing* approximately half the physical qubits from a square code to yield a rotated code with the same odd linear dimensions, we observe a significant reduction in logical failure rate with biased noise. Specifically, we perform numerical simulations, using strongly converged approximate maximum-likelihood decoding, to demonstrate the aforementioned significant reduction in logical failure rate against biased noise that is achieved using a rotated $j \times j$ code, containing $n = j^2$ physical qubits, compared to a square $j \times j$ code, containing $n = 2j^2 - 2j + 1$ physical qubits. Figure 3.2 summarizes this result, comparing logical failure rate as a function of physical error probability for a rotated 9×9 code (81 qubits) and a square 9×9 code (145 qubits) across a range of biases. We see that the advantage of the rotated code over the square code is greatest in the limit of pure Y noise ($\eta = \infty$) and remains significant down to a more modest bias, $\eta = 100$ (where Y errors are 100 times more likely than both X and Z errors). We further argue that, for a given bias, the relative advantage of (odd) rotated codes over square codes increases with code size, until low-rate errors become the dominant source of logical failure and high-rate errors are effectively suppressed, motivating the search for efficient near-optimal biased-noise decoders for rotated codes.

Note that this performance with biased noise is not shared by all topological codes; in stark contrast, the triangular 6.6.6 color code [10] exhibits a decrease in threshold with bias; see Sec. 3.A.

The paper is structured as follows. Section 3.2 provides some definitions used throughout the paper. Our main analytical results for surface codes with pure Y noise are in Sec. 3.3. Our numerical results for surface codes with pure Y noise and Y-biased noise are in Secs. 3.4 and 3.5, respectively. Section 3.6 defines the tensor-network decoder used in simulations of Y-biased noise on rotated codes. We conclude in Sec. 3.7 with a discussion of our results in the context of prior work and raise some open questions for future work. Finally, Section 3.A gives comparative results for color codes, and Section 3.B defines the exact maximum-likelihood decoder used in simulations of pure Y noise on square and coprime surface codes.

3.2 Definitions

Standard surface code.— We consider $j \times k$ standard surface codes [1] on a square lattice with "smooth" top and bottom boundaries and "rough" left and right boundaries. Physical qubits are associated with edges on the lattice. Following the usual convention, stabilizer generators consist of X operators on edges around vertices, $A_v = \prod_{e \in v} X_e$, and Z operators on edges around plaquettes, $B_p = \prod_{e \in p} Z_e$. The stabilizer group is, therefore, $\mathcal{G} = \langle A_v, B_p \rangle$. Up to multiplication by an element of $\mathcal{G}$, the $\overline{X}$ $(\overline{Z})$ logical operator consists of X (Z) operators along the left (top) edge, such that $\overline{X}, \overline{Z} \in \mathcal{C}(\mathcal{G}) \setminus \mathcal{G}$ and $\overline{X}\,\overline{Z} = -\overline{Z}\,\overline{X}$, where $\mathcal{C}(\mathcal{G}) = \{f \in \mathcal{P} : fg = gf \ \forall \ g \in \mathcal{G}\}$ is the centralizer of $\mathcal{G}$ and $\mathcal{P}$ is the group of n-qubit Paulis. As such, a $j \times k$ surface code encodes one logical qubit into $n = 2jk - j - k + 1$ physical qubits with distance $d = \min(j, k)$. Figure 3.3 illustrates a 4×5 surface code.

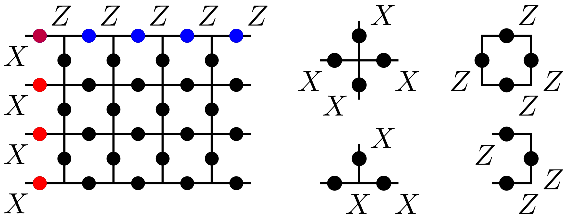


Figure 3.3: Standard 4×5 surface code, with logical operators given by a product of X along the left edge and a product of Z along the top edge. Stabilizer generators are shown at the right.

Rotated surface code.— We also consider rotated surface codes, which are defined by drawing the boundary at 45° relative to the standard surface code lattice [8]; see Fig. 3.4(a). As with standard codes, stabilizer generators consist of X (Z) operators on edges around vertices (plaquettes), with these restricted to two qubits on the boundaries. The $\overline{X}$ $(\overline{Z})$ logical operator consists of X (Z) operators along the northeast (northwest) edge. The rotated code is usually, and equivalently, depicted as in Fig. 3.4(b), where shaded and blank faces correspond to X- and Z-type stabilizer generators, respectively. As such, a rotated $j \times k$ surface code encodes one logical qubit into $n = jk$ physical qubits with distance $d = \min(j, k)$. Unless otherwise stated, we consider rotated surface codes with j and k odd.

Surface code families.— For standard $j \times k$ surface codes, we define the following code families: *square* where $j = k$; $\gcd(j, k) = g$ const; and *coprime* where g=1 (special case of g constant). In addition, for rotated $j \times k$ surface codes, we define the family of *rotated* codes with j and k odd.

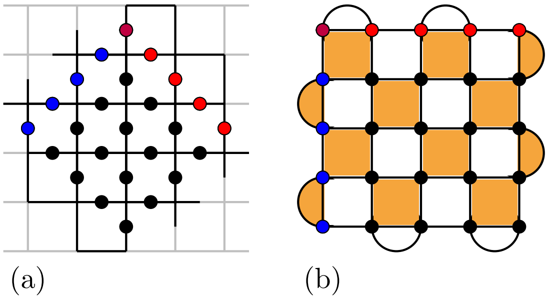


Figure 3.4: (a) Rotated 5×5 surface code defined by drawing the boundary at 45° relative to the surface code lattice. Logical operators are given by a product of X along the northeast edge and Z along the northwest edge. As with the standard code, stabilizer generators consist of X (Z) operators on edges around vertices (plaquettes). (b) Rotated 5×5 surface code as it is usually, and equivalently, depicted, where shaded (blank) faces corresponding to X-type (Z-type) stabilizer generators.

Y-type stabilizers and logical operators.— We define a Y-type stabilizer to be any operator on a code that is in the stabilizer group $\mathcal{G}$ and consists only of Y and identity single-qubit Paulis. We define a Y-type logical operator to be any operator on a code that is in $\mathcal{C}(\mathcal{G}) \setminus \mathcal{G}$ and consists only of Y and identity single-qubit Paulis. We define X- and Z-type stabilizers and logical operators analogously. As usual, the weight of an operator is the number of nonidentity single-qubit Paulis applied by the operator.

Y-distance.— We define Y-distance, or distance d_Y to pure Y noise, of a code as the weight of the minimum-weight Y-type logical operator. X- and Z-distance are defined analogously. The overall distance of the code is defined in the usual way and is upper bounded by $\min(d_X, d_Y, d_Z)$.

Y-biased noise.— Several conventions have previously been used to define biased Pauli noise models [4, 7, 11–23]. We adapt the approach of Ref. [7] to Y-biased noise, by considering an independent, identically distributed Pauli noise model defined by an array $\boldsymbol{p} = (1-p, p_X, p_Y, p_Z)$ corresponding to the probabilities of each single-qubit Pauli I (no error), X, Y, and Z, respectively, such that the probability of any error on a single qubit is $p = p_X + p_Y + p_Z$. We define bias η to be the ratio of the probability of a Y error to the probability of a non-Y error such that $\eta = p_Y/(p_X + p_Z)$. For simplicity, we restrict to the case $p_X = p_Z$. With this definition $\eta = 1/2$ corresponds to standard depolarizing noise with $p_X = p_Y = p_Z = p/3$, and the limit $\eta \to \infty$ corresponds to pure Y noise, i.e., only Y errors with probability p. We define X- and Z-biased noise analogously.

3.3 Features of surface codes with pure Y noise

In this section, we present our analytical results for surface codes with pure Y noise. In Secs. 3.3.1–3.3.4, we present results for standard surface codes, and, in Sec. 3.3.5, we relate these results to rotated surface codes. We first highlight the specificities of syndromes of pure Y noise. Our main result reveals that error correction with the standard surface code with pure Y noise is equivalent to a concatenation of two classical codes: the repetition code at the bottom level and the cycle code at the top level. As a corollary, we show that the surface code with pure Y noise has a threshold of 50%. We also highlight that, for standard $j \times k$ surface codes with small $g = \gcd(j, k)$, the more effective repetition code dominates the performance of the code. We then give explicit formulas for the minimum weight and count of Y-type logical operators. Finally, we relate these results to rotated surface codes. These results explain the origins of the ultrahigh thresholds of the surface code with Y-biased noise, as seen in Ref. [7] and improved in Sec. 3.5.1, as well as the lower logical failure rates seen with coprime and rotated surface codes, presented in Secs. 3.4.1 and 3.5.2.

3.3.1 Syndromes of pure Y noise

An obvious feature of Y noise on the surface code is that Y errors anticommute with both X- and Z-type stabilizer generators, providing additional bits of syndrome information. For comparison, Fig. 3.5 shows a sample of Y-error configurations alongside identically placed X- and Z-error configurations with corresponding anticommuting syndrome locations for each error type. In each case, we see that Y-error strings anticommute with more syndrome locations than X- or Z-error strings, providing the decoder with more information about the location of errors to be corrected.

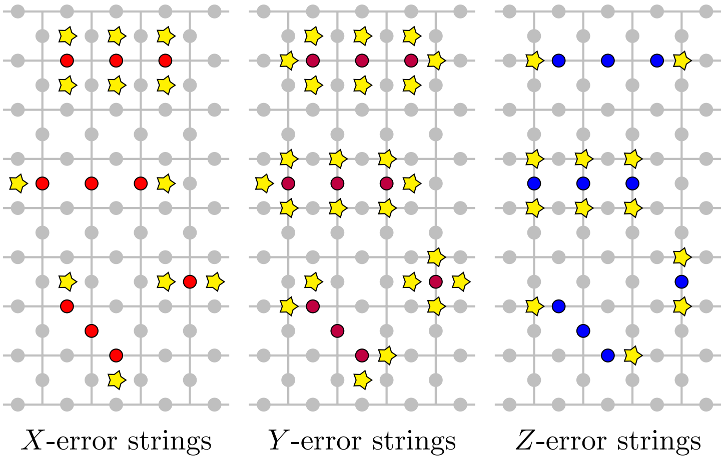


Figure 3.5: A sample of X-, Y-, and Z-error strings, indicated by colored circles, with corresponding anticommuting syndrome locations, indicated by yellow stars.

We remark that the displacement between the X- and Z-type stabilizer generators appears to be significant. For example, the color 6.6.6 code has colocated X- and Z-type stabilizer generators, so that, even if Y errors anticommute with more stabilizer generators, the number of distinct syndrome locations triggered by Y errors is no greater than for X or Z errors.

3.3.2 Structure of the standard surface code with pure Y noise

In this section, we consider standard surface codes subject to pure Y noise. We describe a polynomial-time decoding algorithm and prove that it achieves an error threshold of 50%. We also derive an exponential upper bound on the probability of logical errors in the subthreshold regime. Our main result is a structural theorem that reveals a hidden concatenated structure of the surface code and highlights the role of the parameter $g = \gcd(j, k)$. The theorem implies that error correction with the surface code subject to Y noise can be viewed as a concatenation of two classical codes: the repetition code at the bottom level and the so-called cycle code at the top level. Both codes admit efficient decoding algorithms and have an error threshold of 50%, although the repetition code scores much better in terms of the logical error probability. We show that, for a fixed number of qubits, the size of each code can vary drastically depending on the value of g. Loosely speaking, the error-correction workload is shared between the two codes such that for small g the dominant contribution comes from the more effective repetition code, which explains the enhanced performance of coprime surface codes ($g = 1$) observed in the numerics.

Concatenated structure

Consider a Pauli error

$$P(y) \equiv Y_1^{y_1} \otimes Y_2^{y_2} \otimes \cdots \otimes Y_n^{y_n}, \tag{3.1}$$

where $y \in \{0, 1\}^n$. As described in Sec. 3.3.1, the syndrome of $P(y)$ is given by

$$A_v(y) = \sum_{e \in v} y_e \qquad \text{and} \qquad B_p(y) = \sum_{e \in p} y_e \tag{3.2}$$

where v and p run over all vertices and all plaquettes of the lattice and the sums are modulo two. A decoding algorithm takes as input the error syndrome and outputs a candidate recovery operator $P(y')$ that agrees with the observed syndrome. The decoding succeeds if $y' = y$ and fails otherwise. [More generally, the decoder needs to identify only the equivalence class of errors that contains $P(y)$, where the equivalence is defined modulo stabilizers of the surface code.]

Consider a classical linear code of length n defined by the parity checks $A_v(y) = 0$ and $B_p(y) = 0$ for all v and p. We shall refer to this code as a *Y-code*. As described above, error correction for the surface code subject to Y-noise is equivalent to error correction for the Y-code subject to classical bit-flip errors. We now establish the structure of the Y-code. For any integer $m \geq 3$, let K_m be the complete graph with m vertices and $e = m(m-1)/2$ edges. Consider bit strings $x \in \{0, 1\}^e$ such that bits of x are associated with edges of the graph K_m. Let $x_{i,j}$ be the bit associated with an edge (i, j). Here it is understood that $x_{i,j} = x_{j,i}$. Define a *cycle code* $\mathcal{C}_m$ of order m that encodes $m - 1$ bits into e bits with parity checks

$$x_{i,j} \oplus x_{j,k} \oplus x_{i,k} = 0 \qquad \text{for all } 1 \leq i < j < k \leq m. \tag{3.3}$$

Thus, parity checks of $\mathcal{C}_m$ correspond to cycles (triangles) in the graph K_m. Note that Eq. (3.3) defines a redundant set of parity checks. It is well known that any connected graph with m vertices and e edges has $e - m + 1$ independent cycles. Thus, $\mathcal{C}_m$ has $e - (m - 1)$ independent parity checks. The number of encoded bits is $m - 1$. Note that $\mathcal{C}_2$ is a trivial code (it has no parity checks). Let $\mathrm{REP}(m)$ be the repetition code that encodes one bit into m bits. We can now describe the structure of the Y-code.

Theorem 3.1 (Y-code structure)**.** *The Y-code is a concatenation of the cycle code $\mathcal{C}_{g+1}$ at the top level and $g(g+1)/2$ repetition codes at the bottom level. The latter consists of repetition codes $\mathrm{REP}(jk/g^2)$, $\mathrm{REP}(2jk/g^2)$, and $\mathrm{REP}(4jk/g^2)$ with multiplicities 1, $2(g-1)$, and $g(g+1)/2 - 2g + 1$, respectively.*

An important corollary of the theorem is that a decoding algorithm for the cycle code can be directly applied to correcting Y errors in the surface code. Indeed, a decoder for the Y-code can be constructed in a level-by-level fashion such that the bottom-level repetition codes are decoded first and the top-level cycle code is decoded afterwards.

For example, Theorem 3.1 implies that, with pure Y noise, a coprime ($g = 1$) surface code is essentially a single repetition code of a size growing linearly with n, whereas a square surface code is equivalent to the concatenation of bottom-level fixed-size repetition codes REP(1), REP(2), and REP(4) and a top-level cycle code of a size growing linearly with n, where n is the number of physical qubits in the surface code.

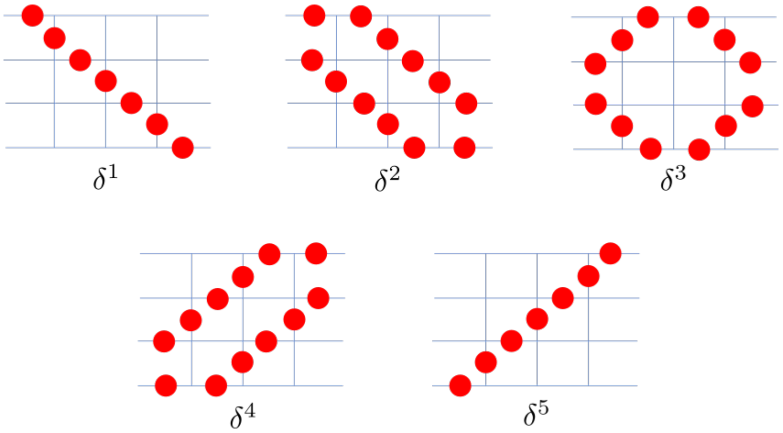


Figure 3.6: Diagonals δ^i for the 4×4 surface code. We consider the symmetry group $\mathcal{R}$ generated by reflections of the lattice against δ^1 and δ^5. Note that any diagonal δ^i is symmetric under reflections from $\mathcal{R}$.

Proof. Let us first prove the theorem in the special case of square surface codes, $j = k = g$. Let $\mathcal{G} \subset \{0,1\}^n$ be the code space of the Y-code. We use a particular basis set of codewords called *diagonals*. The $j \times j$ lattice has $j+1$ diagonals denoted $\delta^1, \delta^2, \ldots, \delta^{j+1} \in \mathcal{G}$; see Fig. 3.6. Given a codeword $y \in \mathcal{G}$, let $\partial y \in \{0,1\}^j$ be the restriction of y onto the top horizontal row of edges in the surface code lattice. We claim that y is uniquely determined by ∂y. Indeed, let $H_1, \ldots, H_j$ be the rows of horizontal edges (counting from the top). Let $V_2, \ldots, V_j$ be the rows of vertical edges (counting from the top). By definition, the restriction of y onto H_1 coincides with ∂y. Suppose the restriction of y onto $H_1 V_2 \ldots H_p$ is already determined (initially $p = 1$). Vertex parity checks $A_v(y) = 0$ located at the row H_p then determine the restriction of y onto V_{p+1}. Likewise, suppose the restriction of y onto $H_1 V_2 \ldots H_p V_p$ is already determined. Plaquette parity checks $B_p(y) = 0$ located at the row V_p then determine the restriction of y onto H_{p+1}. Proceeding inductively shows that any codeword $y \in \mathcal{G}$ is uniquely determined by ∂y.

Define bit strings

$$e^1 = 100\ldots0, \quad e^2 = 010\ldots0, \quad e^3 = 001\ldots0 \quad \text{etc.}$$

Then $\partial\delta^1 = e^1$, $\partial\delta^i = e^{i-1} + e^i$ for $2 \le i \le j$, and $\partial\delta^{j+1} = e^j$; see Fig. 3.6. It follows that $\partial\delta^1, \ldots, \partial\delta^j$ span the binary space $\{0,1\}^j$. Accordingly, the diagonals $\delta^1, \ldots, \delta^j$ span the code space $\mathcal{G}$ and

$$\delta^{j+1} = \delta^1 \oplus \delta^2 \oplus \cdots \oplus \delta^j.$$

In particular, $\dim(\mathcal{G}) = j$, that is, the Y-code encodes j bits into n bits.

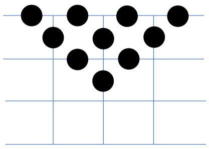


Figure 3.7: A set of qubits $\mathcal{O}$ such that each orbit of $\mathcal{R}$ contains exactly one qubit from $\mathcal{O}$. In this example the group $\mathcal{R}$ has ten orbits of size 1, 2, and 4.

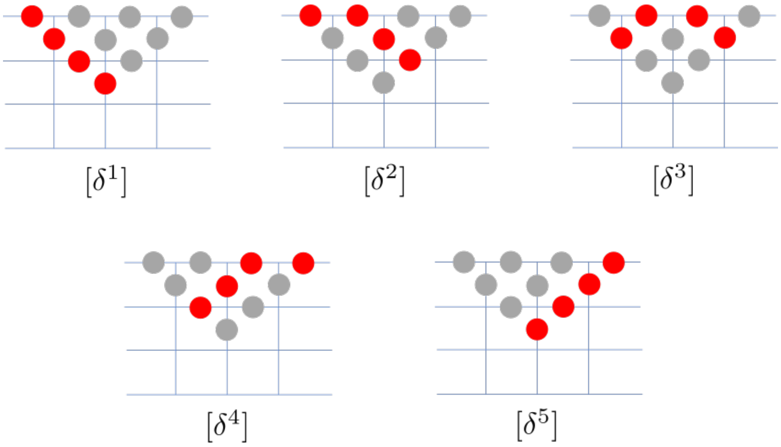


Figure 3.8: Restrictions of the diagonal δ^i onto $\mathcal{O}$ define a basis set of codewords for the top-level code.

Let $\mathcal{R} \cong \mathbb{Z}_2 \times \mathbb{Z}_2$ be a group generated by reflections of the lattice against the diagonals δ^1 and δ^{j+1}. Note that any diagonal δ^i is invariant under reflections from $\mathcal{R}$; see Fig. 3.6. Suppose f is an edge of the surface code lattice. Let $\mathcal{R}(f)$ be the orbit of f under the action of $\mathcal{R}$. The above shows that any diagonal δ^i is constant on orbits of $\mathcal{R}$; that is, $\mathcal{R}(f) = \mathcal{R}(g)$ implies that $\delta^i_f = \delta^i_g$. Since the diagonals δ^i span the full code space $\mathcal{G}$, we conclude that any codeword $y \in \mathcal{G}$ is constant on orbits of $\mathcal{R}$; that is, $\mathcal{R}(f) = \mathcal{R}(g)$ implies that $y_f = y_g$. Equivalently, each orbit of $\mathcal{R}$ of size m gives rise to the repetition code $\mathrm{REP}(m)$. A simple counting shows that $\mathcal{R}$ has a single orbit of size 1 (the central vertical edge) and $2(j-1)$ orbits of size 2 (pairs of qubits located on the diagonals δ^1 and δ^{j+1}), whereas all remaining orbits have size 4, which proves the last statement of the theorem (in the special case $j = k$).

Fix a set of qubits $\mathcal{O}$ such that each orbit of $\mathcal{R}$ contains exactly one qubit from $\mathcal{O}$. In other words, $\mathcal{O}$ is a set of orbit representatives. We choose $\mathcal{O}$ as shown in Fig. 3.7. A simple counting shows that $|\mathcal{O}| = j(j+1)/2$. Consider a codeword $y \in \mathcal{G}$ and let $[y] \in \{0,1\}^{|\mathcal{O}|}$ be a vector obtained by restricting y onto $\mathcal{O}$. We define the top-level code as a linear subspace $\mathcal{L} \subseteq \{0,1\}^{|\mathcal{O}|}$ spanned by vectors $[y]$ with $y \in \mathcal{G}$. Equivalently, $\mathcal{L}$ is spanned by vectors $[\delta^i]$ with $i = 1, \ldots, j+1$. A direct inspection shows that each qubit $e \in \mathcal{O}$ belongs to exactly two vectors $[\delta^i]$ and $[\delta^k]$ for some $i \neq k$; see Fig. 3.8 for an example. Thus, one can identify $\mathcal{O}$ with the set of edges of the complete graph K_{j+1}, whereas the vectors $[\delta^i]$ can be identified with "vertex stabilizers" in K_{j+1}. In other words, the support of each vector $[\delta^i]$ coincides with the set of edges incident to some vertex of K_{j+1}. We conclude that parity checks of $\mathcal{L}$ correspond to closed loops in K_{j+1}. Thus, the top-level code coincides with the cycle code $\mathcal{C}_{j+1}$.

The above proves the theorem in the special case $j = k$. Consider now the general case

$j \neq k$. Let us tile the surface code lattice by $t = jk/g^2$ tiles of size $g \times g$ as shown in Fig. 3.9. Note that each horizontal edge is fully contained in some tile. Let us say that a vertical edge is a boundary edge if it overlaps with the boundary of some adjacent tiles. If one ignores the boundary edges, each tile contains a single copy of the $g \times g$ surface code. For each tile, define the diagonals $\delta^1, \delta^2, \ldots, \delta^{g+1}$ as above. Let $\mathcal{G}$ be the code space of the Y-code for the full $j \times k$ lattice. Recall that any codeword $y \in \mathcal{G}$ is fully determined by its projection ∂y onto the top horizontal row of edges. Using this property, one can easily verify that the code space $\mathcal{G}$ is spanned by "extended diagonals" Δ^i such that the restriction of Δ^i onto the top-left tile coincides with δ^i and Δ^i alternates between δ^i and δ^{g+2-i} in a checkerboard fashion; see Fig. 3.10. An example of the extended diagonal Δ^1 is shown in Fig. 3.9. By definition, Δ^i has no support on the boundary edges, which implies that the Y-code has a weight-1 parity check for each boundary edge. Ignoring such weight-1 checks, each codeword Δ^i consists of t copies of the diagonal δ^i with some copies being reflected. Considering t copies of each codeword instead of a single copy is equivalent to replacing the repetition codes REP(1), REP(2), and REP(4) in the above analysis by REP(t), REP($2t$), and REP($4t$), respectively, where $t = jk/g^2$ is the number of tiles. □

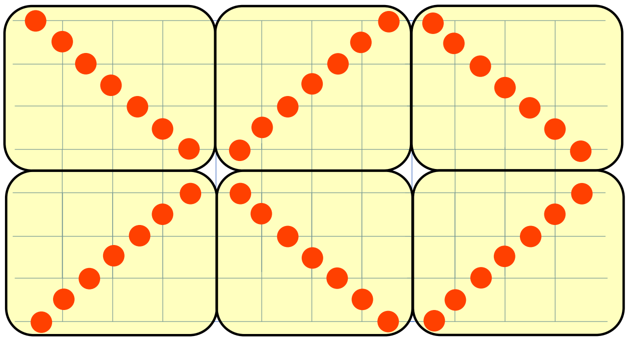


Figure 3.9: Partition of the 8×12 surface code into 4×4 tiles. Solid red circles: The extended diagonal Δ^1 alternating between δ^1 and δ^5; see Fig. 3.6.

$$\Delta^i = \begin{array}{|c|c|c|} \hline \delta^i & \delta^{g+2-i} & \delta^i \\ \hline \delta^{g+2-i} & \delta^i & \delta^{g+2-i} \\ \hline \end{array}$$

Figure 3.10: Extended diagonal Δ^i.

Figure 3.11 provides an illustration of the mapping between the classical cycle code $\mathcal{C}_4$ and the 3×3 standard surface code with pure Y noise. With appropriate numbering of bits and qubits, cycle code codewords are identified with surface code Y-type stabilizers and logical operators, whereas cycle code parity checks are identified with surface code vertex and plaquette parity checks. This mapping applies to the surface code with pure Y noise; see Ref. [24] for a discussion of toric code construction in terms of cycle codes.

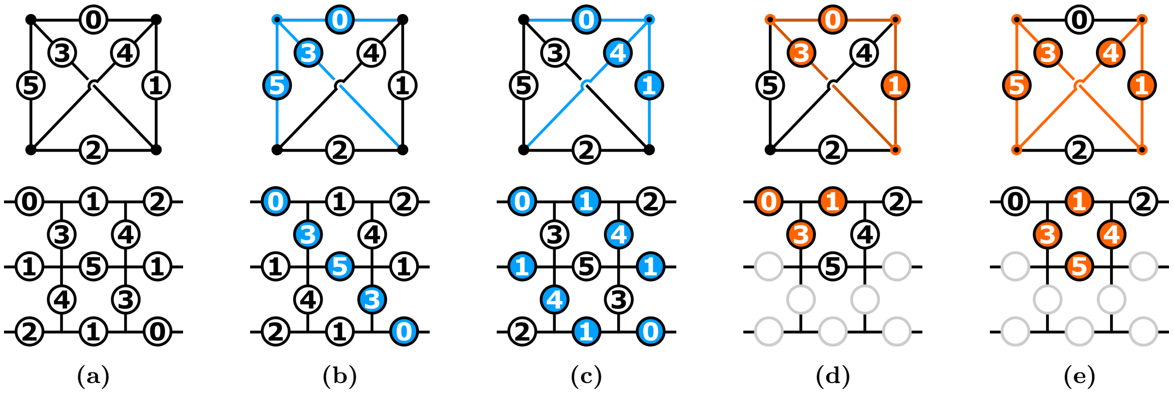


Figure 3.11: Examples of the mapping between the classical cycle code $\mathcal{C}_4$ and the 3×3 standard surface code with pure Y noise. (a) Bits on the edges of the cycle code (upper) are mapped to qubits on edges of the surface code (lower) by a matching number; repeated numbering of qubits indicate that they form an underlying repetition code. (b) and (c) Cycle code codewords are identified with Y-type logical operators or Y-type stabilizers on the surface code. (d) and (e) Cycle code parity checks are identified with vertex or plaquette parity checks on the surface code; (for clarity, only one representative qubit from each underlying repetition code is shown).

Decoding the cycle code

Here, we consider the cycle code subject to random errors. We give a polynomial-time decoding algorithm that achieves the error threshold of 50%. Fix some integer $m \geq 3$ and consider the cycle code $\mathcal{C}_m$ defined earlier, see Eq. (3.3). Recall that $\mathcal{C}_m$ has length $n = m(m-1)/2$. We consider independent and identically distributed (IID) bit-flip errors such that each bit is flipped with probability $p \in [0, 1/2)$. Define an error bias $\epsilon > 0$ such that

$$2p(1-p) = \frac{1}{2} - \epsilon. \tag{3.4}$$

Lemma 3.1 (Cycle code decoder)**.** *Let $e \in \{0,1\}^n$ be a random IID error with a bias ϵ. There exists an algorithm that takes as input the syndrome of e and outputs a bit string $e' \in \{0,1\}^n$ such that*

$$\mathrm{Prob}[e' = e] \geq 1 - 2m^2 \cdot \exp\left(-2\epsilon^2 m\right). \tag{3.5}$$

The algorithm has runtime $O(m^3)$.

Proof. Recall that the cycle code $\mathcal{C}_m$ is defined on the complete graph with m vertices such that each bit of $\mathcal{C}_m$ is located on some edge (i,j) of the graph. Let $e_{i,j}$ be the error bit associated with an edge (i,j). We begin by giving a subroutine that identifies a single error bit $e_{i,j}$. Without loss of generality, consider the edge $(1,2)$. This edge is contained in $m-2$ triangles that give rise to syndrome bits

$$\begin{aligned} s_3 &= e_{1,2} \oplus e_{2,3} \oplus e_{3,1}, \\ s_4 &= e_{1,2} \oplus e_{2,4} \oplus e_{4,1}, \\ &\cdots \\ s_m &= e_{1,2} \oplus e_{2,m} \oplus e_{m,1}. \end{aligned} \tag{3.6}$$

Since errors on different edges of each triangle are independent, the conditional probability distributions of syndromes s_j for a given error bit $e_{1,2}$ are

$$\begin{aligned}
\mathrm{Prob}[s_j = 1|e_{1,2} = 0] &= \frac{1}{2} - \epsilon, \\
\mathrm{Prob}[s_j = 0|e_{1,2} = 0] &= \frac{1}{2} + \epsilon, \\
\mathrm{Prob}[s_j = 1|e_{1,2} = 1] &= \frac{1}{2} + \epsilon, \\
\mathrm{Prob}[s_j = 0|e_{1,2} = 1] &= \frac{1}{2} - \epsilon.
\end{aligned}$$

Furthermore, since different triangles in Eq. (3.6) intersect only on the edge $(1, 2)$, we have

$$\mathrm{Prob}[s_3, \ldots, s_m|e_{1,2}] = \prod_{j=3}^{m} \mathrm{Prob}[s_j|e_{1,2}]. \tag{3.7}$$

This equation is an IID distribution of $m - 2$ bits which is ϵ biased toward $e_{1,2}$. Hoeffding's inequality gives

$$\mathrm{Prob}[s_3 + \ldots + s_m \geq m/2|e_{1,2} = 0] \leq 4 \exp\left(-2\epsilon^2 m\right)$$

and

$$\mathrm{Prob}[s_3 + \ldots + s_m \leq m/2|e_{1,2} = 1] \leq 4 \exp\left(-2\epsilon^2 m\right).$$

The desired subroutine outputs $e_{1,2} = 0$ if $s_3 + \ldots + s_m \leq m/2$ and $e_{1,2} = 1$ otherwise. Clearly, the above calculations take time $O(m)$.

The full decoding algorithm applies the above subroutine independently to each edge of the graph learning error bits one by one. By the union bound, such an algorithm misidentifies the error with a probability of at most $2m^2 \exp\left(-2\epsilon^2 m\right)$ since the complete graph K_m has $m(m - 1)/2$ edges. The overall runtime of the algorithm is $O(m^3)$. □

Note that the decoding algorithm of Lemma 3.1 can be viewed as a single round of the standard belief-propagation algorithm, which is commonly used to decode classical low-density parity check (LDPC) codes; see Ref [25] for more on efficient cycle code decoding. Also recall that the cycle code $\mathcal{C}_m$ has length $n \sim m^2/2$. Thus, the probability of a logical error in Eq. (3.5) decays exponentially with $\sqrt{n}$ [this scaling is unavoidable, since the cycle code $\mathcal{C}_m$ has distance $O(m)$]. As a consequence, the proposed decoder performs very poorly in the small-bias regime. For example, reducing the error rate from 49% to 1% requires code length $n \approx 10^{17}$ [here, we use Eq. (3.5) as a rough estimate of the logical error probability]. In contrast, the logical error probability of the repetition code $\mathrm{REP}(n)$ decays exponentially with n.

3.3.3 Threshold of the standard surface code with pure Y noise

The standard surface code with pure Y noise is equivalent to a concatenation of two classical codes, as shown above, and both of these classical codes have thresholds of 50%. These results lead directly to the fact that the threshold of the surface code with pure Y noise is 50%. Indeed, let us employ the level-by-level decoding strategy such that the bottom-level repetition codes are decoded first. Assume that the pure Y noise has error rate $p < 1/2$. Then, the jth repetition code makes a logical error with probability $p_j \leq p < 1/2$. The effective error model for the top-level cycle code is a product of symmetric binary channels with error rates $p_1, \ldots, p_m \le p$, where $m = g(g+1)/2$ is the length of the cycle code. One can easily verify that the decoder of Lemma 3.1 corrects such a random error with a probability given by Eqs. (3.4) and (3.5). Finally, Theorem 3.1 implies that each parity check of the repetition or the cycle code is a linear combination (modulo two) of the plaquette and vertex parity checks of Eq. (3.2). The coefficients in this linear combination can be found by solving a suitable system of linear equations in time $O(n^3)$, which enables an efficient conversion between the surface code syndrome and the syndromes of the bottom-level and the top-level code. To conclude, Theorem 3.1 and Lemma 3.1 have the following corollary.

Corollary 3.1 (Y-threshold)**.** *The error-correction threshold for the surface code with pure Y noise is* 50%*. This error threshold can be achieved by a polynomial-time decoding algorithm.*

In Sec. 3.3.5, we show that the above corollary also applies to rotated surface codes, with odd linear dimensions. A numerical demonstration of the 50% threshold of the surface code with pure Y noise is given in Sec. 3.4.1.

3.3.4 Y-type logical operators of the standard surface code

The structure of standard surface codes with pure Y noise, described in Sec. 3.3.2, also manifests itself in the structure and, consequently, the minimum weight and count of Y-type logical operators, i.e., logical operators consisting only of Y and identity single-qubit Paulis. In this section, we give explicit formulas for the minimum weight and count of Y-type logical operators. Highlighting the cases of coprime and square codes, as well as comparing the formulas to those for X- and Z-type logical operators, we remark on how the minimum weight and count of Y-type logical operators contribute to the performance advantage with pure Y noise and Y-biased noise seen in Ref. [7] and Sec. 3.5.1, for surface codes, in general, and in Secs. 3.4.1 and 3.5.2, for coprime and rotated codes, in particular.

Logical operator minimum weight

We show that the minimum-weight Y-type logical operator on standard surface codes is comparatively heavy. The X-distance d_X of a code is the weight of the minimum-weight X-type logical operator. Clearly, the minimum-weight X-type logical operator on a $j\times k$ code is a full column of X operators on horizontal edges, and, hence, $d_X = j$; similarly, $d_Z = k$. It is also clear that the minimum-weight Y-type logical operator on a square $j\times j$ code is a full diagonal of Y operators, and, hence, $d_Y = 2j - 1$. From the proof of Theorem 3.1, it is apparent that, in the case of pure Y noise, a $j\times k$ surface code can be viewed as a tiling of jk/g^2 copies of a square $g\times g$ code, where $g = \gcd(j,k)$. Therefore, the Y-distance of a $j\times k$ surface code is given by the following corollary.

Corollary 3.2 (Y-distance)**.** *For a standard $j\times k$ surface code, the weight of the minimum-weight Y-type logical operator, and, hence, the distance of the code to pure Y noise, is*

$$d_Y = \frac{(2g-1)jk}{g^2}$$

where $g = \gcd(j,k)$.

As shown in Sec. 3.3.5, the Y-distance of the rotated $j \times k$ surface code, with j and k odd, is $d_Y = jk$. The distances to pure noise for various surface code families are summarized in Table 3.1. We note that, for all code families, Y-distance exceeds X- or Z-distance, which is consistent with the increase in error threshold of surface codes with biased noise seen in Ref. [7] and Sec. 3.5.1. Furthermore, we note that the Y-distance of square codes is $d_Y = O(\sqrt{n})$ while that of coprime and rotated codes is $d_Y = O(n)$, where n is the number of physical qubits. This feature of near-optimal and optimal Y-distance contributes to the significant improvement in logical failure rate of coprime and rotated codes over square codes with pure Y noise and Y-biased noise, see Secs. 3.4.1 and 3.5.2.

Table 3.1: Distances to pure noise for $j \times k$ surface code families. (d_P refers to the distance to pure P noise, where $P \in \{X, Y, Z\}$.)

Code family	d_X	d_Y	d_Z
Square	j	$2j-1$	j
Coprime	j	jk	k
$\gcd(j,k) = g$	j	$(2g-1)(jk/g^2)$	k
Rotated (j, k odd)	k	jk	j

Logical operator count

We show that Y-type logical operators on standard surface codes are comparatively rare. The number c_X of X-type logical operators is equal to the number of ways the logical $\overline{X}$ operator can be deformed by X-type stabilizer generators. The number of X-type stabilizer generators (i.e., vertices) on a $j \times k$ surface code is $j(k-1)$, and, hence, $c_X = 2^{j(k-1)}$; similarly, $c_Z = 2^{(j-1)k}$. From the proof of Theorem 3.1, it is apparent that the g basis codewords of the Y-code correspond to a single logical operator and a full set of $g-1$ linearly independent Y-type stabilizers of a $j \times k$ surface code, where $g = \gcd(j,k)$. Therefore, the number of Y-type logical operators of a $j \times k$ surface code is given by the following corollary.

Corollary 3.3 (Y-count)**.** *For a standard $j \times k$ surface code, the number of Y-type logical operators is*

$$c_Y = 2^{g-1}$$

where $g = \gcd(j,k)$. The number of Y-type stabilizers is also c_Y.

As shown in Sec. 3.3.5, the number of Y-type logical operators on the rotated $j \times k$ surface code, with j and k odd, is $c_Y = 1$. The counts of pure noise logical operators for various surface code families are summarized in Table 3.2. We note that, for all code families, the number of logical operators for pure Y noise is much lower than the number for pure X or Z noise, which is consistent with the increase in error threshold of surface codes with biased noise seen in Ref. [7] and Sec. 3.5.1. Furthermore, we note that the number of Y-type logical operators for square codes is $c_Y = O(2^{\sqrt{n}})$, while for coprime and rotated codes it is $c_Y = O(1)$, where n is the number of physical qubits. This feature, as an extreme example of the role of entropy in error correction [26], contributes to the significant improvement in logical failure rate of coprime and rotated codes over square codes with pure Y noise and Y-biased noise, see Secs. 3.4.1 and 3.5.2.

Table 3.2: Counts of pure noise logical operators for $j \times k$ surface code families. (c_P refers to the number of P-type logical operators, where $P \in \{X, Y, Z\}$.)

Code family	c_X	c_Y	c_Z
Square	2^{j^2-j}	2^{j-1}	2^{j^2-j}
Coprime	$2^{j(k-1)}$	1	$2^{(j-1)k}$
$\gcd(j,k) = g$	$2^{j(k-1)}$	2^{g-1}	$2^{(j-1)k}$
Rotated (j, k odd)	$2^{(j-1)(k+1)/2}$	1	$2^{(j+1)(k-1)/2}$

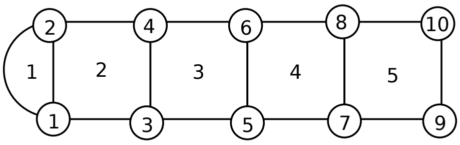
3.3.5 Rotated surface codes

We can relate the results from the previous subsections to rotated surface codes as depicted in Fig. 3.4. In particular, we show that rotated codes, with odd linear dimensions, have similar features to coprime codes as given by Corollaries 3.2 and 3.3; that is, such rotated codes also admit a single Y-type logical operator of weight $O(n)$, where n is the number of physical qubits. Equivalently, the Y-distance of such rotated codes, like coprime codes, is $d_Y = O(n)$; notably, the rotated code is optimal, in that it achieves $d_Y = n$ precisely. Rotated surface codes with even linear dimensions do not share these features, having distance $d_Y = O(\sqrt{n})$ with pure Y noise, and we do not discuss them further. We conclude by showing that the 50% threshold of surface codes with pure Y noise, given by Corollary 3.1, also applies to (odd) rotated codes.

We consider the rotated surface code, with odd linear dimensions, and two-qubit (four-qubit) stabilizer generators on the boundary (in the bulk), as illustrated in Fig. 3.4. The following theorem shows that this version of the surface code is nondegenerate and has a distance of $d_Y = n$ against pure Y noise.

Theorem 3.2 (Rotated code Y-logical)**.** *For a rotated surface code, with odd linear dimensions, $Y^{\otimes n}$ is the only nontrivial Y-type logical operator, where n is the number of physical qubits.*

Proof. It is clear that $Y^{\otimes n}$ is a Y-type logical operator. We show that it is the only nontrivial Y-type operator that commutes with every stabilizer of the code. Let $A = \bigotimes_i Y^{\alpha_i}$ be a Y-type operator. Consider a row of stabilizer generators (checks) and qubits that are adjacent to this row, numbered as shown below:



In order for A to commute with check 1 we require $\alpha_1 = \alpha_2$. With the parity of these checks determined, we then see that, in order for A to commute with check 2, we need $\alpha_3 = \alpha_4$. Continuing along the row, we see that every pair of qubits i, j in the same column must satisfy $\alpha_i = \alpha_j$. The assumption that the code has odd linear dimensions implies that each row and each column of checks includes a weight-two check, as depicted, ensuring that the same argument can equally be applied to every row or column of checks. Therefore, in order for A to commute with all checks, we require $\alpha_1 = \alpha_j$ for all j; i.e., a nontrivial Y-type logical must act as Y on every qubit. □

We note that both the coprime $j\times k$ code and the (odd) rotated $j\times k$ code are nondegenerate against pure Y noise and have Y-distance $d_Y = jk = O(n)$. However, the rotated code is known to be the optimal layout for surface codes in terms of minimum distance [8], and this statement is also true in terms of Y-distance. The rotated code has $d_Y = jk = n$, whereas the coprime code has $d_Y = jk = O(n)$ but contains $n = 2jk - j - k + 1$ physical qubits.

Furthermore, it is clear that the (odd) rotated code with pure Y noise is equivalent to the repetition code and, hence, has a threshold of 50%, in accordance with Corollary 3.1.

3.4 Performance of surface codes with pure Y noise

In Sec. 3.3, we present our analytical results for surface codes with pure Y noise, highlighting features that contribute to the ultrahigh threshold results with Y-biased noise, found in Ref. [7] and improved upon in Sec. 3.5.1. Our analytical results also indicate that coprime and (odd) rotated codes should achieve lower logical failure rates than square codes with pure Y noise.

Here, we present our numerical investigation into the performance of surface codes with pure Y noise. In particular, we present results for square, coprime, and rotated surface code families, confirming the 50% error threshold. We also demonstrate a significant reduction in the logical failure rate for coprime and rotated codes compared with square codes. Specifically, quadratically fewer physical qubits may be used to achieve a target logical failure rate by using coprime or rotated codes compared with square codes.

3.4.1 Advantage of coprime and rotated surface codes with pure Y noise

We investigate the performance of surface codes with pure Y noise. Besides confirming the 50% threshold for the surface code, we demonstrate a significant reduction in logical failure rate for coprime and (odd) rotated surface codes compared to square surface codes such that a target logical failure rate may be achieved with quadratically fewer physical qubits using coprime or rotated codes in place of square codes. That is, we demonstrate that logical failure rate decays exponentially with Y-distance but since, in accordance with Corollary 3.2, the Y-distance of square codes is $O(\sqrt{n})$ and that of coprime and rotated codes is $O(n)$, logical failure rate decays quadratically faster with n for coprime and rotated codes, where n is the number of physical qubits.

In Fig. 3.12, we plot logical failure rate f as a function of physical failure rate p for surface codes belonging to the following families: square, coprime, and rotated codes. For coprime and rotated codes, we see clear evidence of an error threshold at $p_c = 50\%$, consistent with Corollary 3.1. For square codes, the data are consistent with a threshold of $p_c = 50\%$, but the evidence is less definitive. Within a code family, it is expected that smaller codes will perform worse than larger codes below threshold. However, comparing the performance of smaller coprime and rotated codes to square codes, we see a significant improvement in logical failure rate across the full range of physical error probabilities. For example, the 21×21 rotated code, with $n = 441$, and the 20×21 coprime code, with $n = 800$, both clearly outperform the 21×21 square code, with $n = 841$. This result can be seen as a qualitative demonstration of the effect of the features of surface codes with pure Y noise identified in Sec. 3.3.

In Fig. 3.13, we plot logical failure rate f as a function of code distance d_Y to pure Y noise at physical error probabilities p at and below the threshold $p_c = 50\%$ for surface codes belonging to the following families: square, coprime, and rotated codes. For each code family, we see an

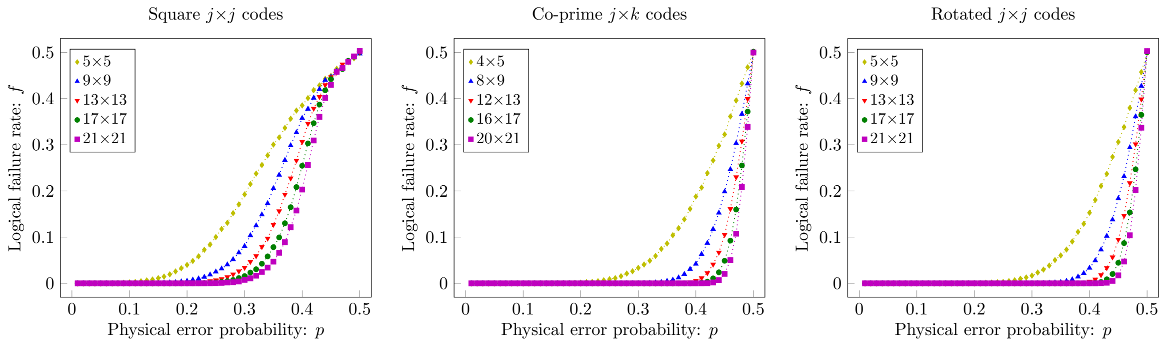


Figure 3.12: Logical failure rate f as a function of physical error probability p for surface code families: square, coprime, and rotated, subject to pure Y noise. Data points are sample means over 60 000 runs using exact maximum-likelihood decoding. Dotted lines connect successive data points for a given code size.

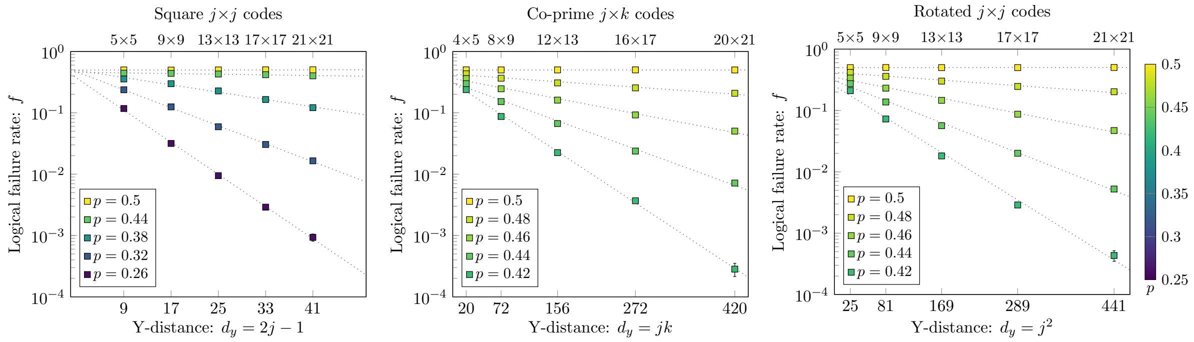


Figure 3.13: Exponential decay of the logical failure rate f with respect to code distance d_Y to pure Y noise in the regime of physical error probability p at and below the error threshold for surface code families: square, coprime, and rotated, subject to pure Y noise. Data points are sample means over 60 000 runs using exact maximum-likelihood decoding. Dotted lines indicate a least-squares fit to data for a given p, and error bars indicate one standard deviation.

exponential decay of the logical failure rate $f \sim \exp(-\alpha d_Y)$, where α is a function of $(p_c - p)$, which is consistent with the threshold $p_c = 50\%$ predicted by Corollary 3.1. Considering $j \times k$ surface codes, according to Corollary 3.2, $d_Y = 2j - 1$ for square codes, $d_Y = jk$ for coprime codes, and $d_Y = j^2$ for rotated codes. That is, $d_Y = O(\sqrt{n})$ for square codes, and $d_Y = O(n)$ for coprime and rotated codes. As a result, based on the observed exponential decay, quadratically fewer physical qubits are required to achieve a target logical failure rate for a given physical error rate by using coprime or rotated codes in place of square codes.

To investigate the performance of different families of surface codes with pure Y noise, we sample the logical failure rate across a full range of physical error probabilities for square, coprime, and rotated codes. For each code family, we use an exact maximum-likelihood decoder. For square and coprime codes, we use the Y-decoder, defined in Sec. 3.B, to avoid the limitations of an approximate [7] or nonoptimal (see Sec. 3.3.2) decoder. For rotated codes, we use the tensor-network decoder, defined in Sec. 3.6, which is exact for pure Y noise. We use code sizes $\{5\times5,\ 9\times9,\ 13\times13,\ 17\times17,\ 21\times21\}$ for square and rotated codes and $\{4\times5,\ 8\times9,\ 12\times13,\ 16\times17,\ 20\times21\}$ for coprime codes and 60 000 runs per code size and physical error probability. In our decoder implementations, we use the Python language with SciPy and NumPy libraries [27, 28] for fast linear algebra and the mathmp library [29] for arbitrary-precision floating-point arithmetic.

3.5 Performance of surface codes with biased noise

Our analytical results (see Sec. 3.3), highlight features of surface codes with pure Y noise that contribute to ultrahigh thresholds with Y-biased noise (see Ref. [7]) and the improvement in logical failure rate achieved by coprime and rotated surface codes (see Sec. 3.4).

Here, we present our numerical investigation into the performance of surface codes with Y-biased noise. In particular, we improve on the results of Ref. [7], providing strong evidence that the threshold of the surface code tracks the hashing bound exactly for all biases. We also demonstrate that the improvement in logical failure rate of coprime and rotated codes with pure Y noise persists with Y-biased noise, such that a *smaller* coprime or rotated code outperforms a square code for a wide range of biases.

3.5.1 Thresholds of surface codes with biased noise

In previous work [7], we show that the surface code exhibits ultrahigh thresholds with Y-biased noise (equivalently, Z-biased noise on the modified surface code of Ref. [7]). The results of Ref. [7] indicate that the threshold error rate of the surface code appears to follow the hashing bound for low to moderate bias; however, it is unclear whether the surface code saturates the hashing bound for all biases.

Here, we improve on the results of Ref. [7], providing strong evidence that the threshold error rate of the surface code saturates the hashing bound exactly for all biases. Our results are summarized in Fig. 3.1, in which threshold estimates for a range of biases are plotted along with the hashing bound. Error bars are one standard deviation relative to the fitting procedure. The threshold estimates are very close to the hashing bound, and any residual differences are likely due to finite size effects and decoder approximation. We estimate the following thresholds: 18.8(2)%, 19.4(1)%, 22.3(1)%, 28.1(2)%, 33.9(2)%, 39.2(1)%, 42.9(2)%, and 45.4(2)%, with $\eta = 0.5$ (standard depolarizing noise), 1, 3, 10, 30, 100, 300, and 1000, respectively. The corresponding hashing bound values are 18.9%, 19.4%, 22.2%, 27.8%, 33.5%, 39.0%, 42.8%, and 45.6%, respectively.

These thresholds are all achieved using a particular tensor-network decoder. The tensor-network decoder of Ref. [9], used in Ref. [7], is an approximate maximum-likelihood decoder tuned via a parameter χ, allowing a trade-off between accuracy and computational cost. In Ref. [7], we use $\chi = 48$ to keep the simulations tractable, but we find the decoder is not completely converged in the intermediate- to high-bias regime. Here, we improve on these results by using a tensor-network decoder, defined in Sec. 3.6, that adapts the decoder of Ref. [9] to rotated codes and achieves a much stronger convergence with biased noise. The convergence of the decoder with bias is summarized in Fig. 3.14, which shows an estimate of the logical failure rate for the rotated 33×33 surface code near threshold as a function of χ for a range of biases. For each bias, the shift in logical failure rate, between the two largest χ shown, is less than half a standard deviation, assuming a binomial distribution.

Our method to numerically estimate the threshold of the surface code with biased noise follows the general approach taken in Ref. [7], with the key difference that we use the tensor-network decoder adapted to rotated codes (see Sec. 3.6) and choose χ such that the decoder

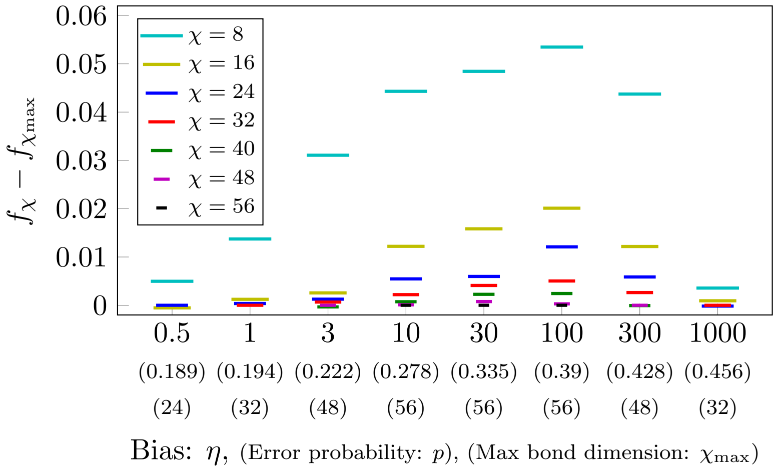


Figure 3.14: Decoder convergence for the rotated 33×33 surface code, represented by shifted logical failure rate $f_\chi - f_{\chi_{\max}}$, as a function of χ at a physical error probability p near the threshold for the given bias η . Each data point corresponds to 30 000 runs with identical errors generated across all χ for a given bias.

more strongly converges. We give a brief summary of the approach here and refer the reader to Ref. [7] for full details. We use rotated surface codes of sizes 21×21, 25×25, 29×29, and 33×33. We estimate the threshold for biases $\eta = 0.5, 1, 3, 10, 30, 100, 300$, and 1000, where $\eta = p_Y/(p_X + p_Z)$ and $p_X = p_Z$ (see Sec. 3.2); we use decoder approximation parameter $\chi = 16, 24, 40, 48, 48, 48, 40$, and 24, respectively, to achieve convergence to within less than half a standard deviation. We run 30 000 simulations per code size and physical error probability. As in Ref. [7], we use the critical exponent method of Ref. [30] to obtain threshold estimates with jackknife resampling over the code distances to determine error bounds.

3.5.2 Advantage of coprime and rotated surface codes with biased noise

In Sec. 3.4.1, we give a demonstration that coprime and rotated surface codes outperform square surface codes with pure Y noise in terms of logical failure rate. It is natural to ask if coprime and rotated codes also outperform square codes with Y-biased noise, i.e., when X and Z errors may also occur. We demonstrate that a significant reduction in logical failure rate against biased noise can be achieved using a rotated $j \times j$ code, containing $n = j^2$ physical qubits, compared to a square $j \times j$ code, containing $n = 2j^2 - 2j + 1$ physical qubits.

Our results are summarized in Fig. 3.2, where we compare the rotated 9×9 code, containing 81 physical qubits, to the square 9×9 code, containing 145 physical qubits. With standard depolarizing noise, i.e., $\eta = 0.5$, and with a low bias, e.g. $\eta = 10$ (where Y errors are 10 times more likely than both X and Z), we see similar performance for the rotated and square codes. In the limit of pure Y noise, we see the very large improvement, across the full range of physical error probabilities, that is already demonstrated in Sec. 3.4.1. Most interestingly, the improvement remains large through the intermediate-bias regime, $\eta = 100$, over a wide range of physical error probabilities, indicating that the advantage of rotated codes over square codes persists with modest noise biases. We note that qualitatively similar results are observed when comparing the coprime 7×8 code to the square 8×8 code (not shown here).

The advantage of rotated codes with biased noise can be explained in terms of the features of surface codes with Y noise identified in Sec. 3.3. The rotated 9×9 code has the same X- and Z-distance ($d_X = d_Z = 9$) as the square 9×9 code. However, the rotated code is much less sensitive to Y noise, having a much larger Y-distance ($d_Y = 81$) than the square code ($d_Y = 17$) and having only one Y-type logical operator ($c_Y = 1$) compared to many more such operators ($c_Y = 2^8 = 256$) on the square code. Therefore, for sufficient bias, we expect rotated $j\times j$ codes to outperform square $j\times j$ codes, despite containing approximately half the number of physical qubits. Also, for a given bias, we expect the relative advantage to increase with code size, as the Y-sensitivity of the rotated code decreases faster than the X- or Z-sensitivity, until low-rate errors become the dominant source of logical failure, at which point high-rate errors are effectively suppressed.

To compare the performance of rotated and square codes with Y-biased noise, we sample the logical failure rate across a full range of physical error probabilities for the square 9×9 code and the rotated 9×9 code with noise biases $\eta \in \{0.5, 10, 100, 1000, 10\,000, \infty\}$. Sample means are taken over 30 000 and 1 200 000 runs per bias and physical error probability for the square and rotated codes, respectively. Since the noise is biased, we cannot use the Y-decoder (see Sec. 3.B) for exact maximum-likelihood decoding. Instead, we use the tensor-network decoder of Ref. [9] for square codes and the tensor-network decoder of Sec. 3.6 for rotated codes, both of which approximate maximum-likelihood decoding. Both decoders are tuned via an approximation parameter χ, which controls the trade-off between efficiency and convergence. As explained in Sec. 3.6, the decoder adapted to rotated codes converges much more strongly with biased noise. We choose $\chi = 48$ for the square code decoder and $\chi = 8$ for the rotated code decoder, such that both decoders converge, at the most challenging bias of $\eta = 100$, to within less than half a standard deviation, relative to $\chi + 8$, assuming a binomial distribution. The use of relatively small codes ensures significant logical failure rates at low physical error probabilities and keeps computational requirements to a reasonable level.

3.6 Improved tensor-network decoding of rotated codes with biased noise

In this section, we describe how tensor-network decoding of the surface code under biased noise can be improved using the rotated surface code layout. We show that the rotated layout allows us to remove certain correlations present in the tensor network used for maximum-likelihood decoding [9], allowing efficient and optimal decoding for pure Y noise. The removal of such correlations greatly improves the efficiency of the decoder in the case of noise strongly biased toward Y, but with a small probability of X and Z errors, a situation previously shown to be challenging using the standard layout [7]. Throughout this section, we refer to surface codes oriented as in Fig. 3.4(b), where shaded and blank faces correspond to X- and Z-type checks, respectively.

We briefly describe the approximate maximum-likelihood decoder proposed by Bravyi, Suchara, and Vargo in Ref. [9]. Maximum-likelihood decoding for stochastic Pauli noise chooses the correction that has the highest probability of successfully correcting the error given an error syndrome, accounting for all errors consistent with that syndrome. If performed exactly it is, by definition, optimal.

The maximum-likelihood decoding algorithm in Ref. [9] is based on mapping coset probabilities to tensor-network contractions. The probability of a coset for an error f is given by

$$\pi(f\mathcal{G}) = \sum_{\alpha,\beta} T(\alpha;\beta), \tag{3.8}$$

where $T(\alpha;\beta)$ is defined as the probability of the Pauli error f times the stabilizer $g(\alpha,\beta) := \prod_v (A_v)^{\alpha_v} \prod_p (B_p)^{\beta_p}$, where $\alpha_v, \beta_p \in \{0,1\}$ and the summation is over all bit strings $\alpha = \alpha_1, \alpha_2, \ldots \alpha_{(n-1)/2}$ and $\beta = \beta_1, \beta_2, \ldots \beta_{(n-1)/2}$. Because of the local structure of the surface code, this summation can be expressed as the contraction of a square-lattice tensor network. While there is some freedom in how the tensor network for a given coset can be defined on both the standard and rotated surface code layouts, we illustrate how a particular definition of the tensor network on the rotated surface code layout can result in significantly more efficient decoding of biased noise.

A complete description of the tensor network that leads to more efficient decoding is provided in Fig. 3.15. We highlight the essential features that give rise to the improved decoding performance. In this layout, every tensor corresponds to a physical qubit, and a horizontal edge between columns i and $i+1$ corresponds to a unique check that acts nontrivially on qubits in both of these columns. We illustrate the correspondence between checks and tensor-network edges on a 5×5 rotated code in Fig. 3.16.

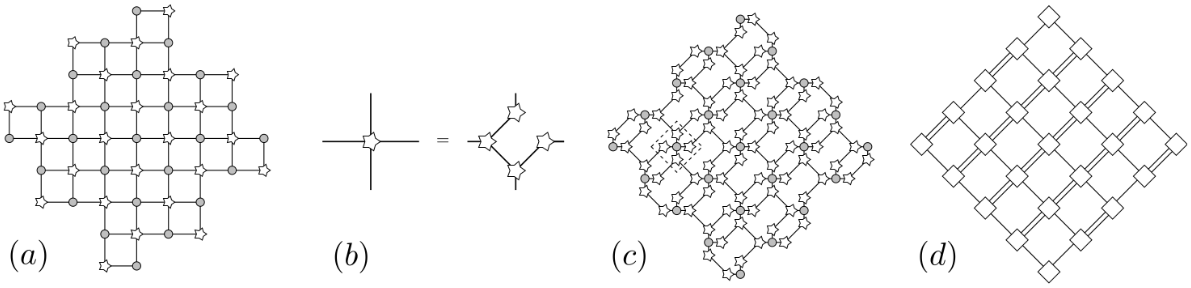


Figure 3.15: (a) Tensor network representing a coset probability for a rotated code. It consists of qubit tensors (circles) and check tensors (stars). The layout in (a) is obtained by applying the construction of Ref. [9] to the rotated code without modification. (b) Splitting a check tensor into multiple check tensors. This splitting is possible because check tensors take the value one if all indices are identical and zero otherwise. (c) A modified tensor network representing a coset probability where a single cell is outlined by a dashed box. This network is obtained from (a) by splitting check tensors. (d) Final modified tensor network obtained by contracting tensors in cells together to form merged tensors (squares). In the discussion of the contraction of this tensor network, we imagine rotating the network anticlockwise by 45° and contracting from left to right. Note that this tensor network is not isotropic: In this rotated frame, the bond dimension is 2 for horizontal edges and is 4 for most vertical edges (except on the boundary).

For certain structured instances of this problem, corresponding to independent X or Z flips, an efficient algorithm for contracting the network is known [9]. However, there is no known efficient algorithm for exact contraction of the network in the case of general local Pauli noise.

In this case, an approximate method for evaluating the tensor-network contraction is used [9]. In this method, the leftmost boundary of the tensor network is treated as a matrix product state (MPS). Columns of the tensor network, which take the form of matrix product operators, are successively applied to the MPS until there are no columns left and the entire lattice is contracted.

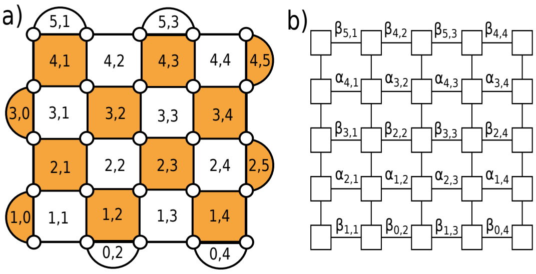


Figure 3.16: (a) Check coordinates are assigned to each check in the rotated layout. (b) The tensor network is defined such that each horizontal edge corresponds to a specific check. The α indices correspond to X checks, and the β indices correspond to Z checks, with the subscripts indicating the check coordinates. Each tensor corresponds to a specific qubit. The bond dimension of horizontal edges is 2, while the bond dimension of the vertical edges is 4.

An approximation is used to keep this calculation tractable. After each column is applied, the singular value decomposition is used to reduce the size of the tensors in the MPS, effectively keeping only the χ largest Schmidt values for each bipartition of the chain and setting the remainder to zero. Without such a truncation, the number of parameters describing the tensors increases exponentially in the number of columns applied to the MPS. The parameter χ can be controlled independently, with larger χ improving accuracy at an increased computational cost. The overall runtime of the algorithm is $\mathcal{O}(n\chi^3)$.

Surprisingly, on the rotated code with the tensor network described above, there is no entanglement in the boundary MPS for pure Y noise. In other words, the MPS decoder is exact for $\chi = 1$, independent of system size. This result is in contrast to the standard layout, where $\chi \sim 48$ is required for a reasonable approximation to coset probabilities on a 21×21 system [7].

As we explain in the following section, this improvement can be attributed to the boundary conditions of the code and the layout of the tensor network. While exact decoding of Y noise can also be performed using methods described in Sec. 3.B, the MPS decoder can be extended easily to noise that is mostly Y noise but with nonzero probability of X and Z errors. Our convergence results (see Sec. 3.5.1) show that there is a substantial improvement in the performance over the standard method when applied to this type of noise.

We observe a similar improvement in performance using the tensor-network decoder described in Ref. [31] when defined on the rotated layout and with an analogous tensor-network layout. Exact decoding is achieved with $\chi = 4$ for pure Y noise, which is not as efficient as the improved MPS decoder described above but substantially more efficient than the MPS decoder on the standard layout.

We remark that, on the standard layout, changing from a square lattice to coprime does little to improve the performance of the MPS decoder. Since the contraction algorithm proceeds column by column, a 21×21 (square) code and a 21×22 (coprime) code have an identical boundary MPS after the first 20 columns are contracted if the same error is applied to qubits in these columns. Thus, we expect the error resulting from the truncation of small singular values during the contraction to be at least as bad for the 21×22 code as the 21×21 code.

3.6.1 Boundary entanglement in MPS decoder

We show that the boundary MPS of the rotated code with the above tensor-network layout is unentangled in the case of infinite bias. The boundary MPS appearing in the contraction algorithm is a (generally approximate) representation of the "boundary state", obtained by contracting all indices of the network up to a given column and leaving the right-going indices of that column uncontracted. More precisely, we define $\psi(\alpha^R;\beta^R)$ to be the contraction of the network up to the $j < L$ column, with the right-going boundary indices set to $\alpha^R;\beta^R$. The L-qubit boundary state is defined as $\left|\psi^R\right\rangle := \sum_{\alpha^R;\beta^R} \psi(\alpha^R;\beta^R)\left|\alpha^R;\beta^R\right\rangle$. We illustrate such a boundary state in Fig. 3.17(a). Let Q_j be the set of qubits in columns up to and including column j. As previously described, each boundary index in $\alpha^R = \alpha_{1,j}, \alpha_{3,j}, \ldots, \alpha_{L-2,j}$ and $\beta^R = \beta_{0,j}, \beta_{2,j}, \ldots, \beta_{L-1,j}$ corresponds to a check acting nontrivially on qubits in columns j and $j+1$, where the check subscripts here are for odd j (for even j, simply add 1 to every row index).

We call checks that act only nontrivially on qubits contained in Q_j bulk checks and refer to them using the indices $\alpha^B;\beta^B$, with superscript B. We refer to a specific $\alpha^R;\beta^R$ as a boundary configuration and a specific $\alpha^B;\beta^B$ as a bulk configuration. We define $\mathcal{G}' \subseteq \mathcal{G}$ to be the set of stabilizer elements that act nontrivially only on Q_j and $g(\alpha^B;\beta^B) \in \mathcal{G}'$ to be the stabilizer element corresponding to the bulk configuration $\alpha^B;\beta^B$. We define $h(\alpha^R;\beta^R)$ to be the product of boundary checks corresponding to the boundary configuration $\alpha^R;\beta^R$ whose action is restricted to qubits in Q_j (so the action on qubits in column $j+1$ is ignored). The fact that $\psi(\alpha^R;\beta^R)$ is calculated by contracting all indices to the left of column j means that all bulk configurations $\alpha^B;\beta^B$ are summed over, like in Eq. (3.8) but restricted to checks in the first j columns. So we can write

$$\psi(\alpha^R;\beta^R) = \pi'(f'\mathcal{G}') = \sum_{\alpha^B,\beta^B} T'(\alpha^B;\beta^B;\alpha^R;\beta^R), \tag{3.9}$$

where π', f', and T', respectively, correspond to versions of π, f, and T that are restricted to Q_j. Specifically, f' is the Pauli error f restricted to Q_j, and $T'(\alpha^B;\beta^B;\alpha^R;\beta^R)$ is the probability of the Pauli error $f'g(\alpha^B;\beta^B)h(\alpha^R;\beta^R)$ on qubits in Q_j. We illustrate an example error f', bulk configuration $\alpha^B;\beta^B$, and boundary configuration $\alpha^R;\beta^R$ in Fig. 3.17. The coset probability π' is likewise restricted to qubits Q_j, with the boundary checks fixed.

In the case of pure Y noise, the summation on the right-hand side of Eq. (3.9) simplifies dramatically. In fact, for any given choice of boundary variables $\alpha^R;\beta^R$ and error f', there is at most one choice of α^B and β^B such that $T'(\alpha^B;\beta^B;\alpha^R;\beta^R)$ is nonzero. So, given $\alpha^R;\beta^R$ and f', either $\psi(\alpha^R;\beta^R)$ is zero or there exists a unique α^B,β^B such that

$$\psi(\alpha^R;\beta^R) = T'(\alpha^B;\beta^B;\alpha^R;\beta^R). \tag{3.10}$$

For a given f' and α^R,β^R, we say that a qubit is "satisfied" for a given check configuration $\alpha^B;\beta^B$ if $f'g(\alpha^B;\beta^B)h(\alpha^R;\beta^R)$ acts on every qubit as either I or Y and not X or Z. For pure Y noise, in order for $T'(\alpha^B;\beta^B,\alpha^R;\beta^R)$ to be nonzero, all qubits in Q_j must be satisfied. We can solve for a bulk configuration $\alpha^B;\beta^B$ that satisfies all qubits, if one exists, by fixing check variables to satisfy qubits one at a time, starting from the qubit adjacent to the two-qubit boundary check in column j. There is only one bulk check adjacent to this qubit; therefore, only one choice for the corresponding check variable will satisfy that qubit. This fixes the first bulk check. We then proceed down this column to fix every check variable in the same

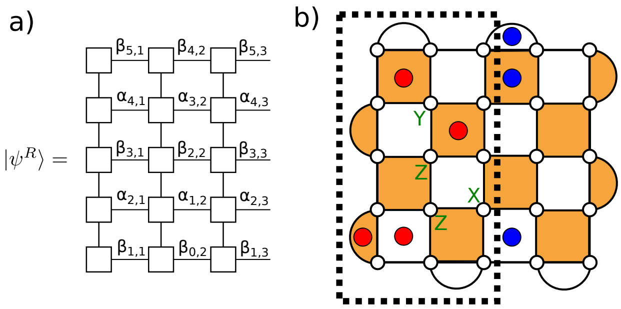


Figure 3.17: (a) A boundary state obtained by contracting the network up to a given column, and leaving the right-going indices of that column uncontracted. (b) An example check and error configuration illustrated for calculating the boundary state for the third column $j = 3$ of a 5×5 code with rotated layout. A bulk configuration $\alpha^B; \beta^B$ is represented by the red dots, and a boundary configuration $\alpha^R; \beta^R$ is represented by the blue dots (where a dot on a check indicates that the check variable $\alpha_{i,k}$ or $\beta_{i,k}$ is 1). An error f' is represented by green letters. Note that for this calculation we consider only the action of the checks and error on qubits in the first three columns Q_3, inside the dashed box. So the action on the boundary checks on qubits outside the box is ignored. The quantity $T'(\alpha^B; \beta^B; \alpha^R; \beta^R)$ is the probability of the product of all dotted checks and the error f' in the dashed box. In the example configuration depicted above, this product contains four X, six Y and two Z errors, giving $T'(\alpha^B; \beta^B; \alpha^R; \beta^R) = p_X^4 p_Y^6 p_Z^2 p_I^3$. In order to calculate the boundary state, all possible configurations of bulk checks must be summed over. In the special case of pure Y noise, where $p_X = p_Z = 0$, only at most one term in this sum is nonzero for any boundary configuration.

manner. With the check configuration in column j determined, we then solve for checks in columns $j-1$, $j-2$, etc., in the same way until all check variables are determined, thereby solving for the bulk configuration $\alpha^B; \beta^B$.

Note that, for certain f' and $\alpha^R; \beta^R$, there may be no configuration of bulk checks that will satisfy all qubits, which implies that the f' and $\alpha^R; \beta^R$ are not compatible with pure Y noise, i.e., $\psi(\alpha^R, \beta^R) = 0$. In fact, only a few special boundary configurations are compatible with pure Y noise. We describe the boundary configurations $\alpha^R; \beta^R$ that are compatible with a given f, starting with the case of the trivial coset $f = I$. We show that the allowed bulk and boundary configurations consist of horizontal strings which terminate at two-qubit X checks on the left code boundary, as shown in Fig. 3.18. Other cosets (with $f \neq I$) follow straightforwardly from this.

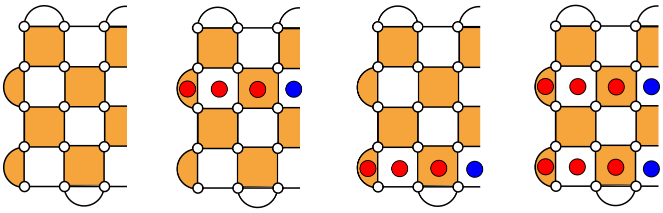


Figure 3.18: All allowed bulk and boundary configurations for Y noise illustrated for the boundary state for the third column $j = 3$ of a 5×5 code on the rotated layout for the trivial coset $f' = I$. The product of the dotted checks must result in only I and Y errors on Q_3 (and no X or Z errors). Blue dots represent the boundary configuration, while red dots represent the corresponding bulk configuration. Blue dots must be connected to a two-qubit check on the left boundary by a string of red dots. The fact that these strings never overlap and are absorbed at the left boundary implies that the boundary variables are uncorrelated, and, therefore, there is no entanglement in the boundary state.

We start from the left-hand side of the code and try to find bulk configurations that satisfy all qubits. We work our way up the column, finding relations between checks. We use the convention that qubit (i,k) refers to the qubit on the bottom-left vertex of the check (face) with coordinates (i,k), as in Fig. 3.16. First, in order to satisfy qubit $(1,1)$, we require $\alpha_{1,0} = \beta_{1,1}$. With the parity of these checks fixed, in order to satisfy qubit $(2,1)$, we need $\alpha_{1,1} = 0$. Then, to satisfy qubit $(3,1)$ we require $\alpha_{3,0} = \beta_{3,1}$. Continuing up the column, we see that $\alpha_{i,0} = \beta_{i,1}$ for i odd and $\alpha_{i,1} = 0$ for i even, and the two-qubit Z check at the end of the column satisfies $\beta_{(L+1)/2,1} = 0$. We can then solve for checks in the next column, finding $\alpha_{i,2} = \beta_{i,1}$ for odd i, $\beta_{i,2} = 0$ for even i, and $\beta_{2,0} = 0$ for the two-qubit check on the lower boundary.

Proceeding in this manner, we can solve for all the checks up to any given column. For the trivial coset, the bulk and boundary configurations satisfy the following:

- All check variables in a given row must be take the same value.
- Check variables in rows terminated by a two-qubit X check (odd rows) may take values 0 or 1. The remaining checks must take the value 0.

We can easily calculate the probability of each satisfying check configuration. First, the trivial configuration $\alpha^R;\beta^R = 0;0$ corresponding to the bulk configuration $\alpha^B;\beta^B = 0;0$, i.e., with all bulk and boundary check variables set to 0, has probability $(p_I)^{jL}$. Flipping any odd boundary check flips the corresponding row of checks in the bulk and introduces $2j$ Y errors, changing the probability by a factor of $(p_Y/p_I)^{2j}$. The fact that the weight introduced by flipping any row does not depend on which other rows are flipped implies that the boundary variables are independent and $\left|\psi^R\right\rangle$ is a product state which can be explicitly written as

$$\left|\psi^R\right\rangle = |0\rangle_{\text{end}} \bigotimes_{k\,\text{even}} |0\rangle_k \bigotimes_{l\,\text{odd}} |\theta\rangle_l \,, \tag{3.11}$$

where $|\theta\rangle = p_I^{2j}|0\rangle + p_Y^{2j}|1\rangle$ and $|0\rangle_{\text{end}}$ corresponds to the two-qubit Z-check at the end of the column. Since the boundary state for the trivial coset $f' = I$ is completely unentangled, the tensor network corresponding to this coset can be contracted exactly with $\chi = 1$.

The case of a nontrivial coset with $f' \neq I$ is analogous to the case of a trivial coset. Starting from any satisfying bulk and boundary configuration, we obtain all other satisfying bulk and boundary configurations by flipping odd rows of checks, as in the trivial coset. If we assume there exist satisfying bulk and boundary configurations $\alpha^{R'};\beta^{R'}$ and $\alpha^{B'};\beta^{B'}$, respectively, for a given error f', the boundary state can be explicitly written as

$$\left|\psi^R\right\rangle = \left|\beta^{R'}{}_{\text{end}}\right\rangle \bigotimes_{k\,\text{even}} \left|\gamma^{R'}{}_k\right\rangle \bigotimes_{l\,\text{odd}} |\theta(l)\rangle_l \,, \tag{3.12}$$

where $|\theta(l)\rangle = p_Y^{N(l)} p_I^{2j-N(l)}|0\rangle + p_Y^{2j-N(l)} p_I^{N(l)}|1\rangle$, $N(l)$ is the number of qubits on which Y is applied in the rows adjacent to the lth row of checks when the boundary variable for row l is 0 and where $\gamma^{R'} = \alpha^{R'}$ for odd j and $\gamma^{R'} = \beta^{R'}$ for even j, and $\beta^{R}_{\text{end}}{}'$ corresponds to the two-qubit check at the end of the column.

Therefore, using the tensor-network layout described above, any coset can be calculated exactly using the MPS decoder with $\chi = 1$, which is a particular property of the physical boundary conditions of the code. In this case described above, starting from a vacuum (with all checks unflipped), flipping a boundary check results in a line of checks being flipped through the bulk, which is absorbed by a two-qubit check on the boundary. We call such a line of flipped check variables a "lineon".

While we can define the tensor network analogously for the standard surface code layout, the boundary state does not have the same product state form. We find that the three-qubit boundary checks result in long-range correlations in the boundary state, which is because the three-qubit checks reflect lineons rather than absorb them, as illustrated in Fig. 3.19. This result means that separated pairs of boundary checks must be flipped together. Also, when distinct lineons travel next to each other or cross, there is a cancellation of Y errors. The consequence of this cancellation is that the probability of a particular lineon depends on whether other lineons are present, which results in correlations between boundary variables and entanglement in the boundary state. The rotated layout with two-qubit checks does not suffer from these problems. The lineons never cross; they are always separated by a row and are absorbed at the boundary.

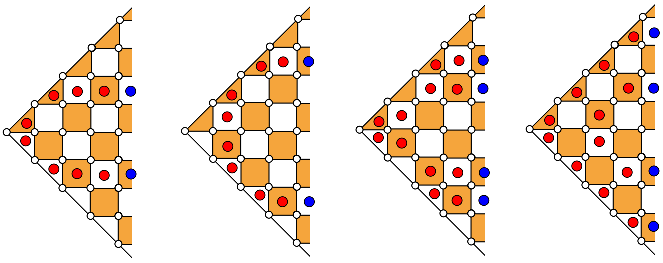


Figure 3.19: Some examples of boundary and bulk configurations for the standard layout of the surface code with three-qubit checks on the boundary for the trivial coset $f' = I$. The lineons travel in straight lines through the four-qubit bulk checks, but the three-qubit boundary checks have the effect of reflecting them by 90°, such that they emerge on the right boundary on the exact opposite side. Therefore, each boundary variable is perfectly correlated with the boundary variable on the exact opposite side. Separate lineons can also cross paths, resulting in a cancellation of the bulk variables. Also, for neighboring pairs of lineons, Y on the qubits shared between them cancel. These all result in correlations between the boundary variables and, therefore, entanglement in the boundary MPS.

To summarize this section, we show that the MPS decoder adapted to the rotated layout is exact with $\chi = 1$ for pure Y noise. This result is due to the fact that many correlations in the tensor network are eliminated in this case, making contraction of the tensor network much more efficient. This decoder can also take into account finite bias (i.e., nonzero p_X and p_Z), and the improvement in efficiency also carries over to this case, as the numerical results of Sec. 3.5.1 show.

3.7 Discussion

In this paper, we describe the structure of the surface code with pure Y noise and show that this structure implies a 50% error threshold and a significant performance advantage in terms of logical failure rate with coprime and rotated codes compared to square codes. Furthermore, we provide numerics confirming our analytical results with pure Y noise and demonstrating the performance advantage of rotated codes with Y-biased noise. It is important to note that our results apply equally to pure Z noise, i.e., dephasing noise, and the Z-biased noise prevalent in many quantum architectures, through the simple modification [7] of the surface code that exchanges the roles of Z and Y operators in stabilizer and logical operator definitions. We, therefore, identify and characterize the features of surface codes that contribute to their ultrahigh thresholds with Z-biased noise and to the improvements in logical failure rate with coprime and rotated codes demonstrated in this paper.

In the limit of pure Y noise, we show that the standard surface code is equivalent to a concatenation of classical codes: a single top-level cycle code and a number of bottom-level repetition codes. We show that this implies the surface code with pure Y noise has a threshold of 50% and, for $j \times k$ surface codes with small $g = \gcd(j, k)$, the more effective repetition code dominates, leading to a reduction in logical failure rate. In terms of logical operators, we show that Y-type logical operators are rarer and heavier than X- or Z-type equivalents, and coprime codes, in particular, have only one Y-type logical operator, and its weight is $O(n)$. We also show that rotated codes, with odd linear dimensions, are closely related to coprime codes, admitting a single Y-type logical operator of optimal weight n.

We confirm, numerically, the 50% error threshold of the surface code with pure Y noise and demonstrate that coprime and rotated codes with pure Y noise significantly outperform similar-sized square codes in terms of logical failure rates such that a target logical failure rate may be achieved with quadratically fewer physical qubits using coprime and rotated codes. Furthermore, we demonstrate that this advantage persists with Y-biased noise. In particular, we find that a *smaller* rotated code, with approximately half the number of physical qubits, outperforms a square code, over a wide range of physical error probabilities, for biases as low as $\eta = 100$, where Y errors are 100 times more likely that X or Z errors. We argue that, for a given bias, the relative advantage of coprime and rotated codes over square codes increases with code size, until low-rate errors become the dominant source of logical errors and high-rate errors are effectively suppressed.

Leveraging features of the structure of rotated codes with pure Y noise, we define a tensor-network decoder that achieves exact maximum-likelihood decoding with pure Y noise and converges much more strongly with Y-biased noise than the decoder of Ref. [9], from which it is adapted. With this decoder, we are able to improve upon the results of Ref. [7] and provide strong evidence that the threshold error rate of surface codes tracks the hashing bound exactly for all biases, addressing an open question from Ref. [7]. Saturating this bound is a remarkable result for a practical topological code limited to local stabilizers.

Although our analytical results focus on features of the surface code with pure Y noise, it is interesting to put our observations of the performance of surface codes with biased noise in the context of other proposals to adapt quantum codes to biased noise [4, 11–22]. Several proposals have been made for constructing asymmetric quantum codes for biased noise from classical codes [11–14] (see Ref. [13] for an extensive list of references), but of particular interest here are approaches that can be applied to topological codes. A significant increase in threshold with biased noise has been demonstrated by concatenating repetition codes at the bottom level with another, possibly topological, code at the top level [4, 15, 16]; interestingly, this construction mirrors the structure we find to be inherent to the surface code. Performance improvements with biased noise have also been demonstrated by modifying the size and shape of stabilizers in Bacon-Shor codes [17–19] and surface and compass codes [20], by randomizing the lattice of the toric code [21] or by concatenating a small Z-error detection code to the surface code [22]. These approaches are distinct from the use of coprime or rotated codes (with the modification of Ref. [7]), which maintain the size and locality of surface code stabilizer generators, and so they could potentially be combined to yield further performance improvements.

Looking forward, the identified features of surface codes and the insights behind them suggest several interesting avenues of research. For the surface code, specifically, different geometries may be more robust to logical errors than coprime and rotated codes in the high-bias regime, where a few well-placed X and Z errors can combine with strings of Y errors to produce more common, lower-weight logical operators. Similarly, certain geometries of surface code used to encode multiple qubits [32] may or may not maintain the high performance of simple surface codes with biased noise. For topological codes, more generally, one can ask which codes exhibit an increase in performance with biased noise and what are the family traits of such codes; we have seen, for example, that the standard triangular 6.6.6 color code does not exhibit an increase in performance. (Although this color code is equivalent, in some sense, to a folded surface code [33], the mapping that relates the two does not preserve the biased noise model.)

Finally, although this paper focuses on features of surface codes with Y or Y-biased noise rather than the issue of fault-tolerant decoding, our numerical results motivate the search for fast fault-tolerant decoders for the surface code with biased noise. The highly significant question of whether the high performance of surface codes with biased noise can be preserved in the context of fault-tolerant quantum computing, is addressed in a forthcoming paper [34], where a fast but suboptimal decoder for tailored surface codes achieves fault-tolerant thresholds in excess of 5% with biased noise. Investigating the optimal fault-tolerant thresholds with biased noise and the performance well below threshold remain important avenues of research.

Acknowledgements

This work was supported by the Australian Research Council (ARC) via the Centre of Excellence for Engineered Quantum Systems (EQUS) Project No. CE170100009 and Discovery Project No. DP170103073. S.B. acknowledges support from the IBM Research Frontiers Institute. A.S.D. was supported by JST, PRESTO Grant No. JPMJPR1917, Japan. Access to high-performance computing resources was provided by the National Computational Infrastructure (NCI), which is supported by the Australian Government, and by the Sydney Informatics Hub, which is funded by the University of Sydney. Some of the numerical computation in this work was carried out at the Yukawa Institute Computer Facility.

3.A Color-code thresholds with biased noise

We demonstrate that the threshold of the triangular 6.6.6 color code [10] decreases when the noise is biased. This result is in stark contrast to the surface code, which exhibits a significant increase in threshold with biased noise [7]. Our results are summarized in Fig. 3.20, in which, we contrast our results for the color code with those for the surface code, reproduced from Sec. 3.5.1. From statistical physics arguments, the optimal error threshold for the unmodified surface code with pure Z noise is estimated to be 10.93(2)% [3, 35], and with depolarizing noise it is estimated to be 18.9(3)% [36]. The color code has similar error thresholds [36, 37] to the surface code with pure Z noise and depolarizing noise. Our results for the color code, using an approximate maximum-likelihood decoder, reveal a decrease in threshold with Y-biased noise: 18.7(1)% with standard ($\eta = 0.5$) depolarizing noise, 13.3(1)% with bias $\eta = 3$, 11.4(2)% with bias $\eta = 10$, 10.6(2)% with bias $\eta = 100$, and 10.5(2)% in the limit of pure Y noise. In contrast, our results for the surface code, from Sec. 3.5.1, reveal a significant increase in threshold with Y-biased noise: 18.8(2)% with standard ($\eta = 0.5$) depolarizing noise, 22.3(1)% with bias $\eta = 3$, 28.1(2)% with bias $\eta = 10$, 39.2(1)% with bias $\eta = 100$, and the analytically proven 50% threshold in the limit of pure Y noise; see Sec. 3.3.3. Our decoder implementation and numerics are described below. The features of surface codes that contribute to their exceptional performance with biased noise are discussed in the body of the paper.

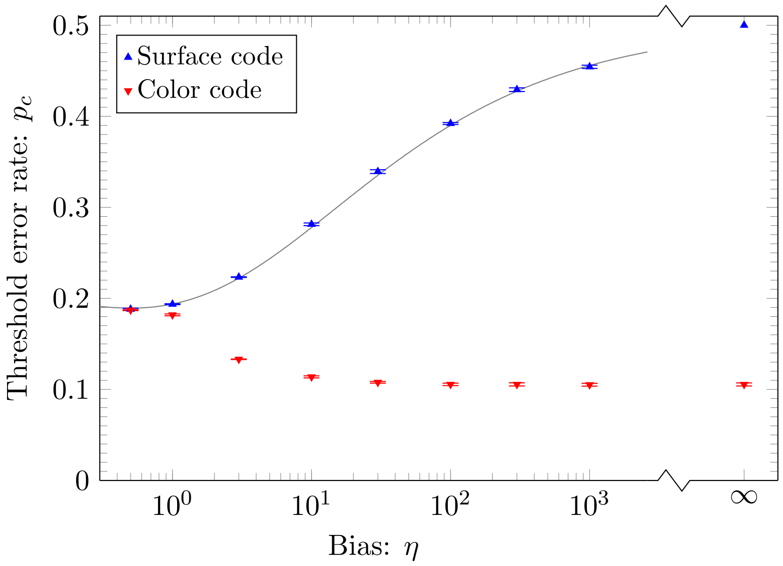


Figure 3.20: Threshold error rate p_c as a function of bias η. Red inverted triangles show threshold estimates for the triangular 6.6.6 color code. For comparison, blue triangles show threshold estimates for the surface code (reproduced from Sec. 3.5.1), with the point at infinite bias, i.e., only Y errors, indicating the analytically proven 50% threshold. Error bars indicate one standard deviation relative to the fitting procedure. The gray line is the hashing bound for the associated Pauli error channel.

3.A.1 Decoder

In order to take account of correlations between X- and Z-type stabilizer syndromes, we implement a tensor-network approximate maximum-likelihood decoder for triangular 6.6.6 color codes following the same principles as the tensor-network decoder of Ref. [9] used in Ref. [7] for surface codes.

Consider a color code with n physical qubits and m independent stabilizer generators. Let $\mathcal{P}$ denote the group of n-qubit Pauli operators, let $\mathcal{G}$ denote the stabilizer group, and recall that the centralizer of $\mathcal{G}$ is given by $\mathcal{C}(\mathcal{G}) = \{f \in \mathcal{P} : fg = gf \ \forall \ g \in \mathcal{G}\}$. If the result of measuring the stabilizer generators is given by syndrome $s \in \{0,1\}^m$ and $f_s \in \mathcal{P}$ is some fixed Pauli operator with syndrome s then the set $f_s\mathcal{C}(\mathcal{G})$ of all Pauli operators with syndrome s is the disjoint union $f_s\mathcal{C}(\mathcal{G}) = f_s\mathcal{G} \cup f_s\overline{X}\mathcal{G} \cup f_s\overline{Y}\mathcal{G} \cup f_s\overline{Z}\mathcal{G}$, where $\overline{X}$, $\overline{Y}$ and $\overline{Z}$ are the logical operators on the encoded qubit.

For a given syndrome s and probability distribution π on the Pauli group, the maximum-likelihood decoder can be implemented by constructing a candidate recovery operator f_s consistent with s and returning $\arg\max_f \pi(f\mathcal{G})$ where $f \in \{f_s, f_s\overline{X}, f_s\overline{Y}, f_s\overline{Z}\}$ and $\pi(f\mathcal{G}) = \sum_{g\in\mathcal{G}} \pi(fg)$.

By analogy with the decoder of Ref. [9] for the surface code, we define a tensor network whose exact contraction yields the coset probability $\pi(f\mathcal{G})$ for the color code. Figures 3.21(a) and 3.21(b) illustrate a distance-5 color code, whereas Figure 3.21(c) illustrates a tensor network with the same layout of qubits and stabilizers. Bonds have dimension 4. Stabilizer tensors are defined such that each element has a value of 1 if all indices are identical and a value of 0 otherwise. Qubit tensors are defined such that each element has the single-qubit probability π of the product of the restriction of f to that qubit with the Paulis associated with bond indices where indices map to Paulis as $0 \mapsto I$, $1 \mapsto X$, $2 \mapsto Y$, and $3 \mapsto Z$. In this way, all possible combinations of stabilizers are applied to f, and the exact contraction of such a tensor network yields the coset probability $\pi(f\mathcal{G})$.

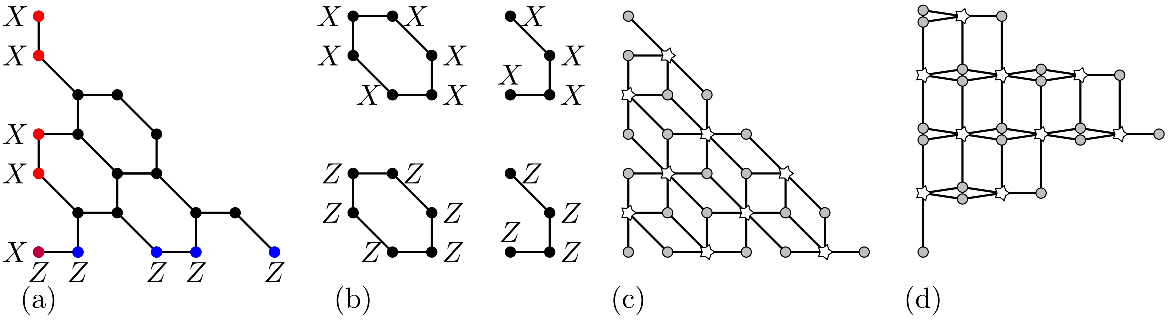


Figure 3.21: (a) Distance-5 triangular 6.6.6 color code with logical operators given by a product of Z along the bottom edge and a product of X along the left edge. (b) Color-code stabilizers. (c) Tensor network corresponding to the coset probability of a distance-5 color code; gray disks represent qubit tensors; white stars represent stabilizer tensors; and lines represent bonds. (d) Equivalent tensor network as a square lattice.

The exact contraction of the tensor network is inefficient with a runtime exponential in the number of qubits n. However, by merging neighboring qubit tensors in pairs, the tensor network can be transformed into a square lattice [see Fig. 3.21(d)] so that techniques, used in the decoder of Ref. [9], can be applied to efficiently approximate the coset probability. The approximation is controlled by a parameter χ which defines the maximum bond dimension retained as the tensor network is contracted. We refer the reader to Ref. [9] for full details of the approximate contraction algorithm. We find that the performance of the decoder converges well for $\chi = 36$ across all noise biases, see below.

3.A.2 Numerics

We follow the general approach taken in Ref. [7]; we give a brief summary here and refer the reader to Ref. [7] for full details. We use triangular 6.6.6 color codes of distances $d = 7, 11, 15$, and 19. We estimate the threshold for biases $\eta = 0.5, 1, 3, 10, 30, 100, 300, 1000, \infty$, where $\eta = p_Y/(p_X + p_Z)$ and $p_X = p_Z$, such that $\eta = 0.5$ corresponds to standard depolarizing noise and $\eta = \infty$ corresponds to pure Y noise (see Sec. 3.2). We approximate maximum-likelihood decoding using the decoder, described above, with approximation parameter $\chi = 36$. The decoder converges well (generally better than in Ref. [7]) across the full range of biases with the weakest convergence in the low-bias regime, see Fig. 3.22. We run 30 000 simulations per code distance and physical error probability. As in Ref. [7], we use the critical exponent method of Ref. [30] to obtain threshold estimates with jackknife resampling over the code distances to determine error bounds.

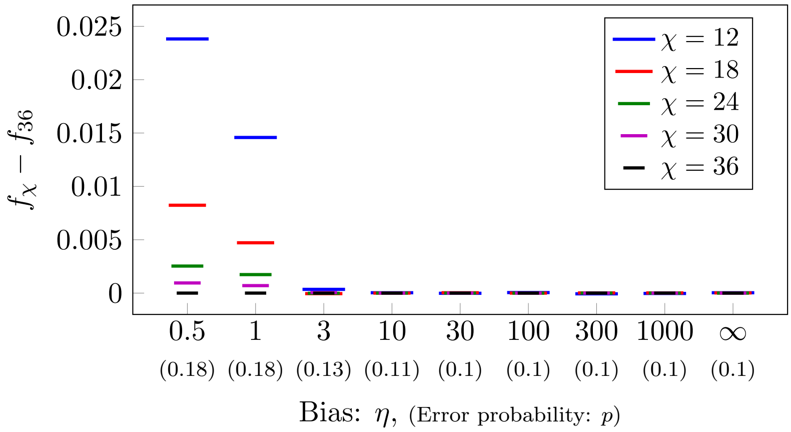


Figure 3.22: Decoder convergence for the distance $d = 19$ triangular 6.6.6 color code, represented by shifted logical failure rate $f_\chi - f_{36}$, as a function of χ at a physical error probability p near the threshold for the given bias η . Each data point corresponds to 60 000 runs with identical errors generated across all χ for a given bias.

3.B Exact optimal Y-decoder

Here, we define the exact optimal decoder for pure Y noise that we use in our numerical simulations of Sec. 3.4.1. As mentioned in Sec. 3.3.2, it is possible to decode Y noise on the planar code by treating it as the concatenation of a cycle code and repetition codes and decoding level by level. However, while efficient, such a decoder is not necessarily optimal. Also, as mentioned in Sec. 3.3.3, the performance of the approximate maximum-likelihood decoder [9] used in previous studies [7] is found to saturate with pure Y noise when tuned for efficiency. Here, we explicitly define an exact maximum-likelihood decoder for the surface code with pure Y noise that is efficient for $j\times k$ surface code families with small $\gcd(j, k)$, such as coprime codes, and tractable for moderate-sized square codes.

Consider a surface code with n physical qubits and m independent vertex and plaquette stabilizer generators. In the case of pure Y noise, the only possible error configurations are Y-type Pauli operators, i.e. operators consisting only of Y and identity single-qubit Paulis. Let $\mathcal{P}_Y$ denote the group of n-qubit Y-type Pauli operators, let $\mathcal{G}_Y$ denote the group of Y-type stabilizers, and define the centralizer of $\mathcal{G}_Y$ as $\mathcal{C}(\mathcal{G}_Y) = \{f \in \mathcal{P}_Y : fg = gf \ \forall\ g \in \mathcal{G}_Y\}$. If the result of measuring the vertex and plaquette stabilizer generators is given by syndrome $s \in \{0, 1\}^m$ and $f_s \in \mathcal{P}_Y$ is some fixed Y-type Pauli operator with syndrome s then the set $f_s\mathcal{C}(\mathcal{G}_Y)$ of all Y-type Pauli operators with syndrome s is the disjoint union $f_s\mathcal{C}(\mathcal{G}_Y) = f_s\mathcal{G}_Y \cup f_s\overline{L}\mathcal{G}_Y$, where $\overline{L}$ is one of the single class of logical operators possible with pure Y noise.

For a given syndrome s and probability distribution π on the Pauli group, the maximum-likelihood decoder for pure Y noise can be implemented by constructing a candidate Y-type recovery operator f_s consistent with s and returning $\arg\max_f \pi(f\mathcal{G}_Y)$ where $f \in \{f_s, f_s\overline{L}\}$ and $\pi(f\mathcal{G}_Y) = \sum_{g\in\mathcal{G}_Y} \pi(fg)$.

On a $j\times k$ surface code, the size of the group of Y-type stabilizers is $|\mathcal{G}_Y| = c_Y = 2^{g-1}$, where $g = \gcd(j, k)$; see Corollary 3.3. Therefore, for surface codes with small g, such as coprime codes, the Y-decoder is efficient, provided that a candidate Y-type recovery operator f_s, the group of Y-type stabilizers $\mathcal{G}_Y$, and logical operator $\overline{L}$ can be constructed efficiently. In the next two subsections, we describe these constructions.

3.B.1 Constructing Y-type stabilizers and logical operators

The construction of Y-type stabilizers and logical operators for a $j \times k$ code is illustrated in Fig. 3.23. A minimum-weight Y-type logical operator is constructed by applying Y operators along a path starting at the top-left corner of the lattice and descending diagonally to the right, reflecting at boundaries, until another corner is encountered from within the lattice. We construct Y-type stabilizers similarly, starting at each of the next $\gcd(j, k) - 1$ qubits of the top row and reflecting until the path cycles. Together, these stabilizers generate the full group of 2^{g-1} Y-type stabilizers, and combine with the minimum-weight logical operator to give the 2^{g-1} Y-type logical operators of the $j \times k$ code.

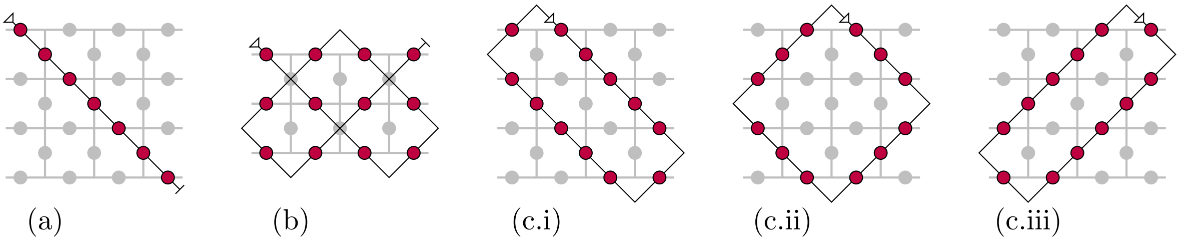


Figure 3.23: Examples of Y-type stabilizer and logical operator construction by applying Y operators along the indicated path until a corner is encountered or the path cycles. Minimum-weight Y-type logical operators (a) and (b) for square 4×4 and coprime 3×4 codes, respectively, are constructed by starting at the top-left qubit. Generators of the group of Y-type stabilizers (c) for the square 4×4 code are constructed by starting at each of the next $\gcd(j, k) - 1 = 3$ qubits of the top row. (For coprime codes, there are no Y-type stabilizers other than the identity.)

3.B.2 Constructing candidate Y-type recovery operators

The construction of a candidate Y-type recovery operator, consistent with a given syndrome, depends on whether the code is coprime, square, or neither.

For coprime codes, it is possible to construct an operator, consisting only of Y and identity single-qubit Paulis, that anticommutes with any single syndrome location. We refer to such operators as Y-type destabilizers. Given a complete syndrome, a candidate Y-type recovery operator is then simply constructed by taking the product of Y-type destabilizers for each syndrome location. One way to construct Y-type destabilizers for coprime codes is illustrated in Fig. 3.24. For a given syndrome location, a partial recovery operator is constructed by applying seed Y operators along a path starting directly below the syndrome location and descending diagonally to the right until a boundary is encountered; further Y operators are applied along paths descending diagonally to the left of each of these seed Y operators, reflecting at boundaries, until the bottom boundary is encountered. The partial recovery operator then anticommutes with the original syndrome location and residual syndrome locations on the bottom boundary. A residual recovery operator is constructed for each residual syndrome location by applying Y operators along a line starting directly to the right of the syndrome location and ascending diagonally to the right, reflecting at boundaries, until a corner is encountered from within the lattice. The residual recovery operators then anticommute with the residual syndrome locations. The destabilizer for the original syndrome location is then simply the product of the partial and residual recovery operators.

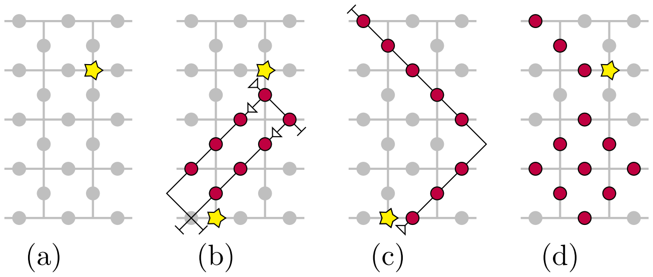


Figure 3.24: Example of Y-type destabilizer construction for a coprime code. (a) Single syndrome location. (b) A partial recovery operator is constructed by applying Y operators, from below the syndrome location along a diagonal to any boundary, then from that diagonal along perpendicular diagonals, until the bottom boundary is encountered. (c) Residual recovery operators are constructed by applying Y operators, from right of each residual boundary syndrome location along a diagonal away, until a corner is encountered. (d) The destabilizer is a product of partial and residual recovery operators.

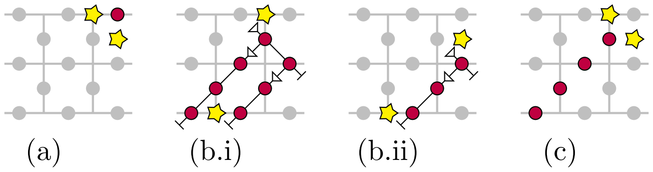


Figure 3.25: Example of candidate Y-type recovery operator construction for a square code using partial recovery operators. (a) Original error and complete syndrome. (b) Partial recovery operators with residual boundary syndrome locations. (c) The candidate recovery operator is the product of all partial recovery operators, since residual boundary syndrome locations cancel in the case of square codes.

For square codes, Y-type destabilizers do not exist, in general, and, hence, a different approach to constructing a candidate Y-type recovery operator must be adopted. Given a complete syndrome for a square code, a candidate Y-type recovery operator can be constructed by taking the product of partial recovery operators for each syndrome location, since the residual boundary syndrome locations cancel in the case of square codes; see Fig. 3.25.

For surface codes that are neither coprime nor square, a candidate Y-type recovery operator is constructed by dividing the lattice into a coprime region and square regions. Partial recovery operators are constructed for each region leaving residual syndrome locations only on plaquettes between regions. Residual syndrome locations can then be moved off the lattice using Y-type stabilizers on the square regions.

References

[1] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," arXiv:quant-ph/9811052, (1998).

[2] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, **87**, pp. 307–346, arXiv:1302.3428, (2015).

[3] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, **43**, 9, pp. 4452–4505, arXiv:quant-ph/0110143, (2002).

[4] P. Aliferis, F. Brito, D. P. DiVincenzo, J. Preskill, M. Steffen, and B. M. Terhal, "Fault-tolerant computing with biased-noise superconducting qubits: a case study," *New J. Phys.*, **11**, 1, p. 013061, arXiv:0806.0383, (2009).

[5] M. D. Shulman, O. E. Dial, S. P. Harvey, H. Bluhm, V. Umansky, and A. Yacoby, "Demonstration of entanglement of electrostatically coupled singlet-triplet qubits," *Science*, **336**, 6078, pp. 202–205, arXiv:1202.1828, (2012).

[6] D. Nigg, M. Müller, E. A. Martinez, P. Schindler, M. Hennrich, T. Monz, M. A. Martin-Delgado, and R. Blatt, "Quantum computations on a topologically encoded qubit," *Science*, **345**, pp. 302–305, arXiv:1403.5426, (2014).

[7] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, "Ultrahigh error threshold for surface codes with biased noise," *Phys. Rev. Lett.*, **120**, p. 050505, arXiv:1708.08474, (2018).

[8] H. Bombin and M. A. Martin-Delgado, "Optimal resources for topological two-dimensional stabilizer codes: Comparative study," *Phys. Rev. A*, **76**, p. 012305, arXiv:quant-ph/0703272, (2007).

[9] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Phys. Rev. A*, **90**, p. 032326, arXiv:1405.4883, (2014).

[10] H. Bombin and M. A. Martin-Delgado, "Topological quantum distillation," *Phys. Rev. Lett.*, **97**, p. 180501, arXiv:quant-ph/0605138, (2006).

[11] L. Ioffe and M. Mézard, "Asymmetric quantum error-correcting codes," *Phys. Rev. A*, **75**, p. 032345, arXiv:quant-ph/0606107, (2007).

[12] P. K. Sarvepalli, A. Klappenecker, and M. Rötteler, "Asymmetric quantum codes: constructions, bounds and performance," *Proc. R. Soc. A*, **465**, 2105, pp. 1645–1672, (2009).

[13] G. G. La Guardia, "On the construction of asymmetric quantum codes," *Int. J. Theor. Phys.*, **53**, pp. 2312–2322, (2014).

[14] A. Robertson, C. Granade, S. D. Bartlett, and S. T. Flammia, "Tailored codes for small quantum memories," *Phys. Rev. Applied*, **8**, p. 064004, arXiv:1703.08179, (2017).

[15] P. Aliferis and J. Preskill, "Fault-tolerant quantum computation against biased noise," *Phys. Rev. A*, **78**, p. 052331, arXiv:0710.1301, (2008).

[16] A. M. Stephens, W. J. Munro, and K. Nemoto, "High-threshold topological quantum error correction against biased noise," *Phys. Rev. A*, **88**, p. 060301, arXiv:1308.4776, (2013).

[17] A. M. Stephens, Z. W. E. Evans, S. J. Devitt, and L. C. L. Hollenberg, "Asymmetric quantum error correction via code conversion," *Phys. Rev. A*, **77**, p. 062335, arXiv:0708.3969, (2008).

[18] J. Napp and J. Preskill, "Optimal Bacon-Shor codes," *Quantum Inf. Comput.*, **13**, pp. 0490–0510, arXiv:1209.0794, (2013).

[19] P. Brooks and J. Preskill, "Fault-tolerant quantum computation with asymmetric bacon-shor codes," *Phys. Rev. A*, **87**, p. 032310, arXiv:1211.1400, (2013).

[20] M. Li, D. Miller, M. Newman, Y. Wu, and K. R. Brown, "2D compass codes," *Phys. Rev. X*, **9**, p. 021041, arXiv:1809.01193, (2019).

[21] B. Röthlisberger, J. R. Wootton, R. M. Heath, J. K. Pachos, and D. Loss, "Incoherent dynamics in the toric code subject to disorder," *Phys. Rev. A*, **85**, p. 022313, arXiv:1112.1613, (2012).

[22] X. Xu, Q. Zhao, X. Yuan, and S. C. Benjamin, "A high threshold code for modular hardware with asymmetric noise," arXiv:1812.01505, (2018).

[23] P. Webster, S. D. Bartlett, and D. Poulin, "Reducing the overhead for quantum computation when noise is biased," *Phys. Rev. A*, **92**, p. 062309, arXiv:1509.05032, (2015).

[24] J. Tillich and G. Zémor, "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Trans. Inf. Theory*, **60**, pp. 1193–1202, arXiv:0903.0566, (2014).

[25] G. Zémor, "On iterative decoding of cycle codes of graphs," in *Codes, Systems, and Graphical Models* (B. Marcus and J. Rosenthal, eds.), 123, pp. 311–326, Springer New York, NY, (2001).

[26] M. E. Beverland, B. J. Brown, M. J. Kastoryano, and Q. Marolleau, "The role of entropy in topological quantum error correction," *J. Stat. Mech.: Theory Exp.*, **2019**, p. 073404, arXiv:1812.05117, (2019).

[27] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," (2001–). https://www.scipy.org/.

[28] T. E. Oliphant, A guide to NumPy, 1. (2006). https://www.numpy.org/.

[29] F. Johansson *et al.*, *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.0)*, (2017). http://mpmath.org/.

[30] C. Wang, J. Harrington, and J. Preskill, "Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory," *Annals of Physics*, **303**, 1, pp. 31 – 58, arXiv:quant-ph/0207088, (2003).

[31] A. S. Darmawan and D. Poulin, "Linear-time general decoding algorithm for the surface code," *Phys. Rev. E*, **97**, p. 051302, arXiv:1801.01879, (2018).

[32] N. Delfosse, P. Iyer, and D. Poulin, "Generalized surface codes and packing of logical qubits," arXiv:1606.07116, (2016).

[33] A. Kubica, B. Yoshida, and F. Pastawski, "Unfolding the color code," *New J. Phys.*, **17**, 8, p. 083026, arXiv:1503.02065, (2015).

[34] D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, "Fault-tolerant thresholds for the surface code in excess of 5% under biased noise," arXiv:1907.02554, (2019).

[35] F. Merz and J. T. Chalker, "Two-dimensional random-bond Ising model, free fermions, and the network model," *Phys. Rev. B*, **65**, p. 054425, arXiv:cond-mat/0106023, (2002).

[36] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, "Strong resilience of topological codes to depolarization," *Phys. Rev. X*, **2**, p. 021004, arXiv:1202.1852, (2012).

[37] H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, "Error threshold for color codes and random three-body Ising models," *Phys. Rev. Lett.*, **103**, p. 090501, arXiv:0902.4845, (2009).

4 Fault-tolerant thresholds for the surface code in excess of 5% under biased noise

DOI: 10.1103/PhysRevLett.124.130501, arXiv: 1907.02554

© 2020 American Physical Society

Preamble

In this chapter, we introduce an efficient approach to decoding that exploits symmetries of a code with respect to a dominant noise model. Applying this approach, with the insights from Chapter 3, we define a decoder for the tailored surface code with Z-biased noise. We use this decoder to extend the ultrahigh threshold results of Chapter 2 to the fault-tolerant context, i.e. the regime directly relevant to experiments where measurements are unreliable.

Abstract

Noise in quantum computing is countered with quantum error correction. Achieving optimal performance will require tailoring codes and decoding algorithms to account for features of realistic noise, such as the common situation where the noise is biased towards dephasing. Here we introduce an efficient high-threshold decoder for a noise-tailored surface code based on minimum-weight perfect matching. The decoder exploits the symmetries of its syndrome under the action of biased noise and generalizes to the fault-tolerant regime where measurements are unreliable. Using this decoder, we obtain fault-tolerant thresholds in excess of 6% for a phenomenological noise model in the limit where dephasing dominates. These gains persist even for modest noise biases: we find a threshold of $\sim 5\%$ in an experimentally relevant regime where dephasing errors occur at a rate 100 times greater than bit-flip errors.

4.1 Introduction

The surface code [1, 2] is among the most promising quantum error-correcting codes to realize the first generation of scalable quantum computers [3–5]. This is due to its two-dimensional layout and low-weight stabilizers that help give it its high threshold [2, 6, 7], and its universal set of fault-tolerant logical gates [2, 8–11]. Ongoing experimental work [12–15] is steadily improving the surface code error rates. Concurrent work on improved decoding algorithms [6, 7, 16–19] is leading to higher thresholds and lower logical failure rates, reducing the exquisite control demanded of experimentalists to realize such a system.

Identifying the best decoder for the surface code depends critically on the noise model. Minimum-weight perfect matching (MWPM) [20, 21] is near optimal in the case of the standard surface code with a bit-flip error model [2] and for a phenomenological error model with unreliable measurements [6]; see [22, 23]. More recently, attention has turned to tailoring the decoder to perform under more realistic types of noise, such as depolarizing noise [16, 18, 19, 24, 25] and correlated errors [26–28]. Of particular note is noise that is biased towards dephasing: a common feature of many architectures. With biased noise and reliable measurements, it is known that the surface code can be tailored to accentuate commonly occurring errors such that an appropriate decoder will give substantially increased thresholds [29, 30]. However, these high thresholds were obtained using decoders with no known efficient implementation in the realistic setting where measurements are unreliable and the noise bias is finite.

In this Letter we propose a practical and efficient decoder that performs well for both finite bias and noisy measurements, demonstrating that the exceptional gains of the tailored surface code under biased noise extend to the fault-tolerant regime. We use the MWPM algorithm together with a recent technique to exploit symmetries of a given quantum error-correcting code [31]. Rather than using the symmetries of the code, we generalize this idea and use the symmetries of the entire system. Specifically, we exploit the symmetries of the syndrome *with respect to its incident error model.* Applied to pure dephasing noise, our decoder exploits the one-dimensional symmetries of the system by pairing the defects of each symmetry separately. Crucially, our approach readily extends to the situation where measurements are unreliable, as well as the finite-bias regime where some low-rate errors violate the symmetries we rely on. We demonstrate that our approach leads to fault-tolerant thresholds exceeding 6% for infinite bias, with these substantial gains persisting to modest biases. Comparing with the *optimal* threshold of 3.3% [22, 23] for conventional decoders that correct the bit-flip and dephasing errors of the same noise model separately, our results represent a very significant improvement in the level of noise that can be tolerated in practical quantum technologies.

4.2 Surface code tailored for dephasing

We define the surface code in a rotated basis with X- and Y-type stabilizers, $S_v \in \mathcal{S}$, to provide additional syndrome information about Z errors; see Fig. 4.1 and its corresponding caption. We consider errors $E \in \mathcal{E}$ drawn from a subgroup of the Pauli group $\mathcal{E} \subseteq \mathcal{P}$. We define the syndrome as a list of the locations of defects. For a given error, defects lie on vertices v such that $S_v E|\psi\rangle = (-1)E|\psi\rangle$ for code states $|\psi\rangle$ satisfying $S_v|\psi\rangle = |\psi\rangle$ for all v.

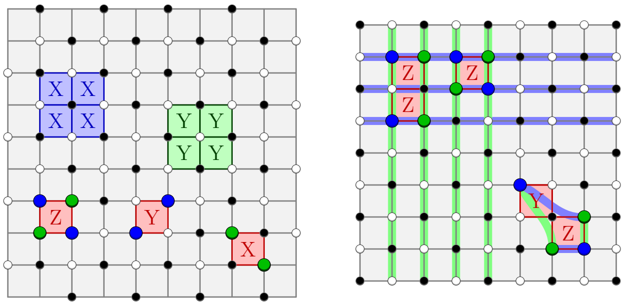


Figure 4.1: (Left) The surface code with qubits on the faces of a square $d \times d$ lattice. The vertices v are bicolored such that stabilizer generators $S_v = \prod_{\partial f \ni v} X_f$ ($S_v = \prod_{\partial f \ni v} Y_f$) lie on black (white) vertices, and $\partial f \ni v$ denotes the faces f touching v. Examples are shown at the top of the figure. The syndrome patterns for Pauli X, Y and Z errors are shown at the bottom of the figure. (Right) The surface code with periodic boundary conditions. Our noise model is such that Z errors occur at a higher rate than Pauli X or Y errors. The syndromes of Z errors, shown at the top left of the figure, respect one-dimensional symmetries, shown as blue and green lines. We can therefore consistently match vertices along the rows and columns of the lattice. The edges returned from each MWPM subroutine reproduce the boundary of the faces that support the error. Lower-rate nondephasing errors may violate the symmetries of the system (bottom, right).

4.3 Decoding with symmetry

We first consider the infinite bias (pure-dephasing) error model generated by only Z errors, $\mathcal{E}^Z = \langle Z_f \rangle$. Errors drawn from this model respect one-dimensional symmetries of the lattice, as in Fig. 4.1. A single Z error generates two defects on each of its adjacent rows and columns. Up to boundary conditions, any error drawn from $\mathcal{E}^Z$ will respect a defect parity conservation symmetry on each of the rows and columns of the lattice.

Let us make this notion of a symmetry rigorous. A symmetry is specified by a subgroup of the stabilizer group $\mathcal{S}_{\text{sym}} \subseteq \mathcal{S}$. Elements $S \in \mathcal{S}_{\text{sym}}$ are defined with respect to an error model $\mathcal{E}$ such that they satisfy $SE|\psi\rangle = (+1)E|\psi\rangle$ for all $E \in \mathcal{E}$ and code states $|\psi\rangle$. This generalizes Ref. [31] where symmetries are defined for the special case where $\mathcal{E} = \mathcal{P}$; the symmetry is now a function of the combined system of both the code and the error model.

For general Pauli error models, the surface code has global symmetries [1]; $\prod_{v \in \mathcal{G}} S_v = 1$ with $\mathcal{G}$ the set of either black or white vertices where we briefly assume periodic boundary conditions to illustrate this point. Under pure dephasing noise, the same model has a much richer set of one-dimensional symmetries. Observe that $S_{\mathcal{L}} = \prod_{v \in \mathcal{L}} S_v$, with $\mathcal{L}$ the set of vertices on a row or column, is a product of Pauli Z matrices. As such, the one-dimensional stabilizers $S_{\mathcal{L}}$ commute with errors drawn from $\mathcal{E}^Z$ and are therefore symmetries. The set of all such $S_{\mathcal{L}}$ generate $\mathcal{S}_{\text{sym}}$ with respect to $\mathcal{E}^Z$.

Now consider what this symmetry implies for an arbitrary syndrome in our error model. A direct consequence of the definition of a symmetry $\mathcal{S}_{\text{sym}}$ is that, for pure dephasing noise, there will always be an even number of defects measured by the subsets of stabilizers whose product gives elements of $\mathcal{S}_{\text{sym}}$. We can design a decoder that exploits this property of these subsets. Specifically, we can consistently pair the defects detected by the stabilizers of these subsets using, say, MWPM, or another suitable pairing algorithm such as that of Ref. [32]. Collections of defects that are combined with pairing operations on sufficiently many symmetries can be neutralized with a low-weight Pauli operator. We say that such a collection is locally correctable [31]. For the surface code under pure dephasing noise, by performing pairing over the one-dimensional lattice symmetries, the edges returned from MWPM form the boundary of the error; see Fig. 4.1(right). The interior of the boundary determines the correction.

Such a decoder is readily extended to the fault-tolerant setting where measurements are unreliable and may give incorrect outcomes. A single measurement error will violate the defect symmetries of the two-dimensional system. Following the approach of Ref. [2], we can recover a new symmetry in the fault-tolerant setting in $(2 + 1)$-dimensional spacetime by repeating stabilizer measurements over time, see also Ref. [31]. A symmetry is recovered by taking the parity of pairs of sequential measurement outcomes, with odd parity heralding a defect. This spacetime symmetry is generic to our proposal here. In this situation, up to the lattice boundaries, the symmetries represent constraints among collections of defects lying on $(1 + 1)$-dimensional planes. Curiously, unlike the phenomenological bit-flip noise model for the surface code [6, 33], the biased phenomenological error model considered here is anisotropic in spacetime. We emphasize the importance of checking for temporal logical errors, consisting of strings of sequential measurement errors, as they may introduce logical failures while performing code deformations [34].

The symmetries of the system are altered at lattice boundaries. We can adapt the decoder to account for this, by adding a pair of defects at each time step to all vertices where a stabilizer is not imposed at the boundary; see Fig. 4.1(left). These defects can be paired to other defects within their respective $(1 + 1)$-dimensional planes of symmetry. Otherwise, they can be matched together freely in the case that they do not need to be paired.

4.4 Decoding with finite bias

We next adapt our decoder to deal with low-rate X and Y errors in addition to high-rate Z errors. For simplicity we will describe this modification for the case of periodic boundary conditions and where measurements are reliable. We give a technical description of our decoders in Section 4.A, and a runtime analysis in Section 4.B.

The decoder for infinite bias noise will pair each defect of the system twice: once to a horizontally separated defect and once to a vertically separated defect. Low rate X and Y errors violate the one-dimensional symmetries that enable us to use the strategy described above, but we can weakly adhere to the strategy as follows. In our modified decoder we pair all defects twice: once where we strongly bias the decoder to pair defects horizontally, and a second time where we strongly bias each defect to pair vertically. Unlike in the infinite-bias case, we permit our decoder to pair defects that are not within their same row or column. We penalize such pairings according to the amount of noise bias. This can be achieved in our input into the MWPM algorithm by assigning high weights to edges for pairs of defects that are not aligned on the same row or column, depending on the respective matching.

In the case of finite bias the collections of defects that are connected through the edges returned by pairing may not be locally correctable. We deal with this issue with additional use of MWPM to complete the decoding procedure. One can show that there will be an even number of defects in each collection of defects connected by edges. Therefore, the parity of defects on black and white vertices are equal. We call collections of defects with an even (odd) parity of defects on black and white vertices 'neutral' ('charged'). Neutral clusters can be locally corrected. Remaining charged collections of defects can be made neutral by pairing them with other nearby charged collections of defects. This final pairing ensures the collections of connected defects are locally correctable.

4.5 Biased noise models

We will test our decoder under two scenarios: with a biased noise model and ideal measurements, and with a phenomenological biased noise model with unreliable measurements. At each time step, qubits are subjected to an error with probability p. Pauli Z errors occur at high rate $p_{\text{h.r.}} = p\eta/(\eta+1)$, while X and Y errors occur at a lower rate, $p_{\text{l.r.}} = p/2(\eta+1)$. The phenomenological (ideal-measurement) biased noise model gives an incorrect measurement outcome with probability $q = p$ $(q = 0)$.

It is important to consider whether the phenomenological noise model we introduced is compatible with a noise-bias setting [35]. As we now demonstrate, it is possible to measure stabilizers and maintain the bias. Following the standard approach [2], stabilizers are measured by preparing an ancilla, a, in an eigenstate of X, then applying entangling gates between the ancilla and the qubits that support the stabilizer, and finally measuring the ancilla qubit in the X basis. To measure S_v for black vertices v we apply $\prod_{\partial f \ni v} CX_{a,f}$ where $CX_{a,f} = (1 + Z_a + X_f - Z_a X_f)/2$ is the controlled-not gate. To measure white vertex stabilizers, we replace the $CX_{a,f}$ gates with $CY_{a,f}$ gates. These gates differ by an $\exp(\mathrm{i}\pi Z_f/2)$ rotation.

We can now justify that stabilizer measurements performed this way preserve the noise bias. Specifically, we demonstrate that no steps in the stabilizer circuit cause high-rate errors to introduce X or Y errors to the data qubits of the surface code. The $CX_{a,f}$ commutes with Z errors that act on the ancilla. As such, it will not create high rate X or Y errors on the data qubits. Similarly, the single-qubit rotation that maps $CX_{a,f}$ onto $CY_{a,f}$ commutes with the high-rate errors, and will therefore only map low-rate errors onto other low-rate errors. Ancilla qubits are vulnerable to high-rate Pauli Z errors. This is reflected by the error model that has a high measurement error rate, $q = p$. An additional concern is that the entangling gates such as $CX_{a,f}$ may increase the frequency that low-rate errors occur. This will depend on the physical implementation, and recent proposals have demonstrated that noise-bias-preserving $CX_{a,f}$ gates are indeed possible in some architectures [36].

4.6 Numerical simulations

We simulate the performance of our decoder for the surface code with periodic boundary conditions against the phenomenological biased noise model, using 30 000 trials per code distance and physical error probability. We used the critical exponent method of Ref. [6], fitting to a quadratic model, to obtain threshold estimates with jackknife resampling over code distances to determine error bounds. Because of the anisotropy in spacetime, we might expect the thresholds of logical errors in the spatial and temporal direction to differ. We report a failure if a logical error occurs in either the spatial or temporal direction. Our results are shown in Fig. 4.2. We identify a threshold of 6.32(3)% for pure dephasing, and thresholds of $\sim 5\%$ for biases around $\eta = 100$. Our decoder begins to outperform the optimal values for standard methods, where bit-flip and dephasing errors are corrected separately, at $\eta \sim 5$. These results demonstrate the advantage of using our decoder in the fault-tolerant setting, even if the noise bias is modest.

We have simulated the performance on the surface code with boundaries, yielding similar results. Figure 4.3 demonstrates a threshold using the fault-tolerant decoder for the surface

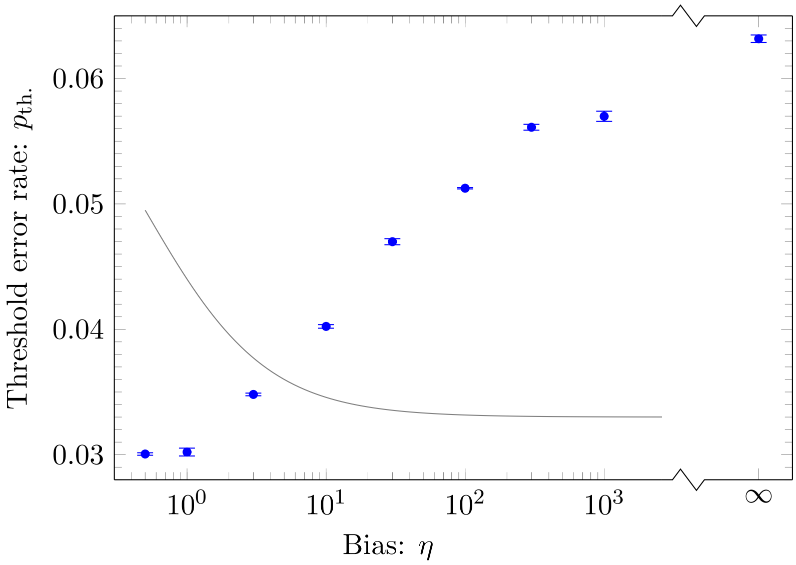


Figure 4.2: Threshold error rates $p_{\text{th.}}$ as a function of noise bias η for both spatial and temporal logical errors for the surface code with periodic boundary conditions. The points show threshold estimates together with 1 standard deviation error bars. The points at smallest and largest bias values correspond to $\eta = 0.5$ (depolarizing noise), and $\eta = \infty$ (pure dephasing), respectively. The solid line represents the optimal performance for the standard surface code with phenomenological noise of a decoder that deals with bit-flip errors and dephasing noise separately. Codes with distance $d = 12, 14, 16, 18, 20$ and $d = 24, 28, 32, 36, 40$ were used for finite and infinite bias threshold estimates, respectively.

code with boundaries where $\eta = 100$. In this case we only measure spatial logical errors because there are no topologically nontrivial temporal errors. Remarkably, the threshold is very similar to the threshold obtained in the case with periodic boundary conditions where we also count logical failures along the temporal direction as well. This is surprising given the anisotropy of the decoding problem in the spatial and temporal directions.

We benchmark our decoder against the optimal performance of the surface code under the biased noise model. In the absence of optimal fault-tolerant thresholds (say, from statistical mechanical arguments [37]), we benchmark using the ideal measurement model. In this case, optimal performance corresponds to the zero-rate hashing bound, which is achievable using a ML decoder [18, 30]. We see in Fig. 4.4 that our decoder underperforms in comparison to the ML decoder, suggesting that there is considerable scope for further improvements. A natural proposal would be to incorporate belief propagation into the MWPM algorithm. Choices of boundary conditions also play a role. We note that our decoder applied to the surface code with boundaries can achieve the optimal threshold of $p_{\text{th.}} \sim 1/2$ for pure dephasing noise. However it underperforms similarly to that shown in Fig. 4.4 at finite biases.

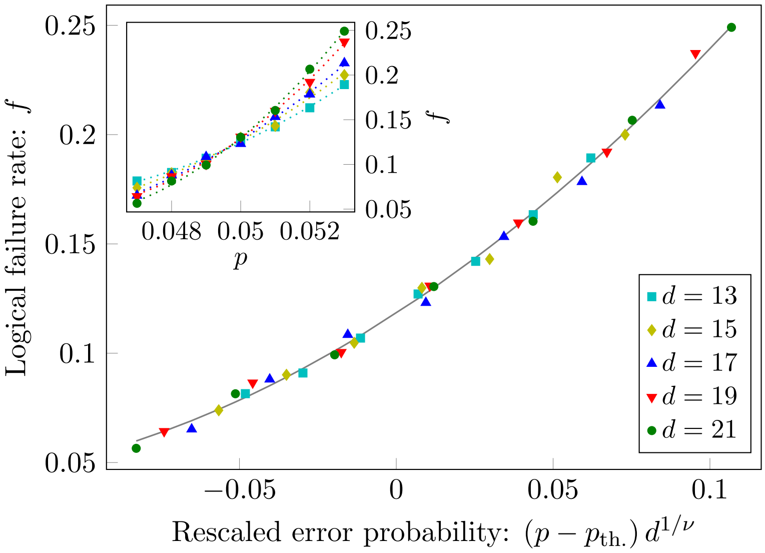


Figure 4.3: Numerical data demonstrating a finite threshold in the fault-tolerant setting. Logical (spatial) failure rate f for the surface code with boundaries shown as a function of the rescaled error rate $x = (p - p_{\text{th.}})d^{1/\nu}$ with bias $\eta = 100$ and $p_{\text{th.}} = 4.96(1)\%$. The solid line is the best fit to the model $f = A + Bx + Cx^2$. The insets show the raw sample means over 30 000 runs for various values of p.

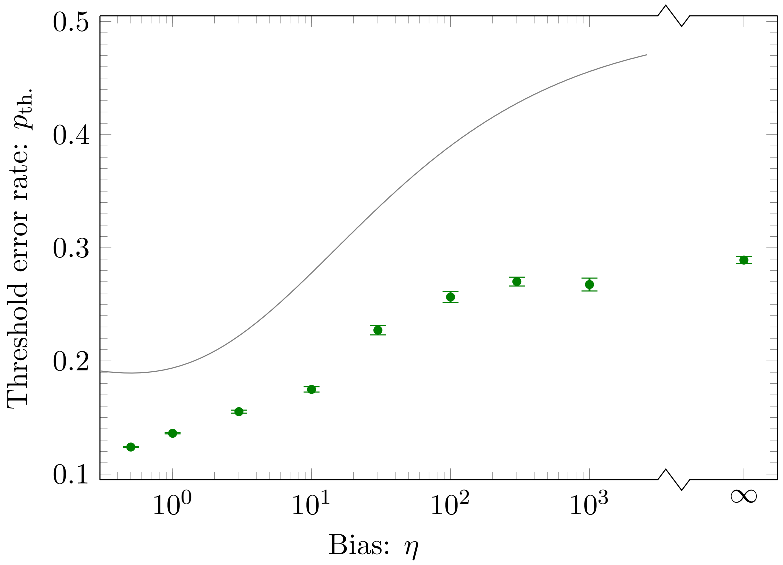


Figure 4.4: Threshold error rates $p_{\text{th.}}$ as a function of noise bias η for the surface code with periodic boundary conditions and ideal measurements. The points show threshold estimates with 1 standard deviation error bars. The points at smallest and largest bias values correspond to $\eta = 0.5$ (depolarizing noise), and $\eta = \infty$ (pure dephasing), respectively. The solid line, which is the zero-rate hashing bound for the associated Pauli error channel, represents threshold error rates that are achievable with ML decoding [30]. Codes with distance $d = 24, 28, 32, 36, 40$ and $d = 48, 56, 64, 72, 80$ were used for finite and infinite bias threshold estimates, respectively.

4.7 Low error rates

The performance of the decoder below threshold will determine the resources required to perform quantum computation. We now speculate on the logical failure rates where the physical error rate is low, specifically $p \ll 1/d$. Using conventional decoding methods the logical failure rate decays as $\mathcal{O}(p^{\delta\sqrt{n}})$ [2, 38] with $n = d \times d$ the code length and δ a constant. The high-threshold at infinite bias is indicative that the decoder can tolerate up to $\sim n/2$ dephasing errors [29, 30]. We may therefore expect that the logical failure rate will decay with improved scaling, $\mathcal{O}({p_{\text{h.r.}}}^{\alpha n})$, for some constant α.

At finite noise bias, the improved scaling in logical failure rate with n can only persist up to some critical system size. Above some system size that depends on η, we expect that the most likely error that will cause a logical failure will be due a string consisting of $\sim \sqrt{n}$ low-rate errors. Up to constant factors, this will occur for some n where ${p_{\text{h.r.}}}^{\alpha n} \ll {p_{\text{l.r.}}}^{\delta\sqrt{n}}$. Nevertheless, given high bias, the decoder will vastly improve logical error rates in the regime where the most likely failure mechanisms are due to long strings of low-rate error events.

We contrast this with bias nullification schemes by concatenation [39, 40]. These approaches increase the effective rate that uncommon errors act on the surface code by a factor equal to the number of qubits of each repetition code, leading to worse performance at low error rates. Moreover, they can only tolerate at most $\propto \sqrt{n}$ high-rate errors.

Extending again to the fault-tolerant case, temporal-logical errors are caused by strings of measurement errors that occur at a high rate. We should consider increasing the number of repetitions T of the error-correction cycle between code deformations to reduce the likelihood of temporal logical failures. Choosing $T \sim 2\delta\sqrt{n}\log p_{\text{l.r.}}/\log p$ will ensure temporal errors will occur at a rate similar to spatial logical errors, $\sim {p_{\text{l.r.}}}^{\delta\sqrt{n}}$, where we have assumed a temporal logical error occurs with likelihood $\sim p^{T/2}$. To achieve the target logical failure rate of the system, although the qubits will be occupied for a longer time to decrease the failure rate of temporal logical errors, the associated decrease in the two spatial dimensions will result in a net improvement on resource scaling using our system.

4.8 Discussion

Minimum-weight perfect matching has formed the backbone of topological quantum error correction [2, 6, 17, 31, 33, 41, 42]. The realization that we can design MWPM decoders with knowledge of the symmetries of the code or system opens up a number of new avenues for decoding algorithm design. A multitude of codes have yet to be explored, as well as their interaction with specialized noise models that reflect the errors that occur in the laboratory. Significant improvements in fault-tolerant thresholds obtained though tailored codes and realistic noise models, such as those we have demonstrated here, offer great promise for the realization of practical quantum technologies.

Acknowledgements

We are grateful for helpful and supportive conversations with A. Grimsmo, N. Nickerson and D. Williamson. This work is supported by the Australian Research Council via the Centre of Excellence in Engineered Quantum Systems (EQUS) project number CE170100009. BJB is also supported by the University of Sydney Fellowship Programme. Access to high-performance computing resources was provided by the National Computational Infrastructure (NCI), which is supported by the Australian Government, and by the Sydney Informatics Hub, which is funded by the University of Sydney.

4.A Decoder implementation

The decoder is described conceptually in Sections 4.3 and 4.4. In this section, we provide technical details of the decoding algorithm implementation and some examples of its operation.

Consider the tailored surface codes defined in Section 4.2, with qubits on the faces and X- and Y-type stabilizers on the black and white vertices, respectively. The input to the decoding algorithm is an error syndrome, and some assumed noise parameters. The output of the decoding algorithm is a recovery operator that returns the code to the codespace. The syndrome is generated from repeated stabilizer measurements [2], such that *defects* are identified with vertex locations in time where the parity of successive pairs of stabilizer measurements is odd. We label defects as X- or Y-type depending on whether they are due to X- or Y-type stabilizer checks.

The decoding algorithm, see Algorithm 4.1, exploits symmetries of the code and noise model in the following way. A graph is constructed with two nodes, labeled horizontal (H) and vertical (V), for each defect. Edges are added between similarly oriented nodes and weighted by a distance function, described in more detail below. Nodes of the graph are paired using minimum-weight perfect matching (MWPM) and the defects identified with the nodes are grouped into *clusters* by following the path traced out by matched pairs of nodes. In the case of pure Z noise, these clusters are guaranteed to have an even parity of each type of defect; such even-parity clusters are locally correctable and we refer to them as *neutral.* In the case of finite bias, clusters are guaranteed to have the same parity of each type of defect but this parity may be odd; such odd-parity clusters are not locally correctable and we refer to them as *charged.* Having locally corrected the clusters as far as possible, the decoding algorithm invokes a residual decoding sub-algorithm, see Algorithm 4.3, that again uses MWPM to pair charged clusters, possibly through neutral clusters, ensuring the code is returned to the codespace.

The graph used in the main decoding algorithm is key to exploiting the symmetries of the code and noise model. As mentioned above, the graph contains a pair of nodes, labeled H and V, for each syndrome defect, and edges are added between similarly oriented nodes (i.e. H to H and V to V), with edge weights given by a distance function. (For a code with boundaries, an additional pair of virtual H and V nodes is added and connected with zero weight, at each boundary vertex where a stabilizer is not applied.) The distance function, see Algorithm 4.2, breaks the path between nodes into time, parallel and diagonal steps. Parallel steps are along horizontal (vertical) lines on the lattice between H (V) nodes. Measurement errors displace defects in time steps, high-rate Z errors displace defects in parallel steps, and low-rate X or Y errors displace defects in diagonal steps. Each of these steps is weighted as a function of the noise parameters: bias, qubit error probability and measurement error probability. The minimum distance between nodes is found by minimizing the number of diagonal, parallel and time steps in that order of priority, and then returning the weighted sum over these steps. Figure 4.5 gives some examples of paths between H and V nodes broken into parallel and diagonal steps.

The derivation of step weights as functions of the noise parameters is as follows. Consider the code, with stabilizer measurements repeated over a fixed number of time steps, represented

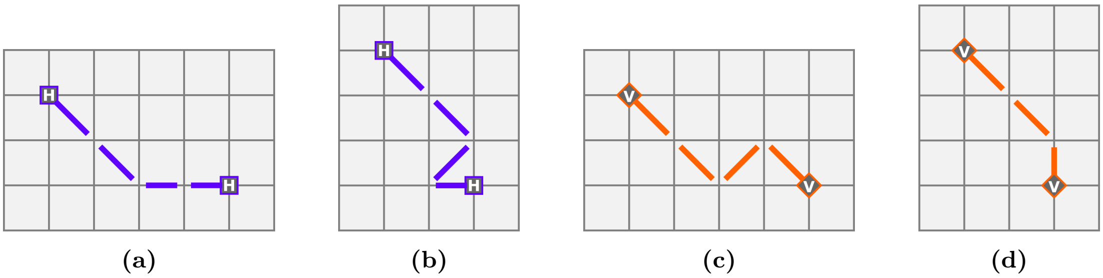


Figure 4.5: Examples of paths between H and V nodes broken into parallel and diagonal steps, minimizing first the number of diagonal steps and then parallel steps; between H (V) nodes parallel steps are horizontal (vertical).

by a (2+1)-dimensional lattice with a total of N qubit locations (one for each physical qubit at each time step) and M stabilizer measurement locations (one for each stabilizer measurement at each time step). The probability of an error configuration, E, consisting of H high-rate qubit errors, L low-rate qubit errors and Q measurement errors, is given by

$$\Pr(E) = (1-p)^{N-H-L} p_{\text{h.r.}}^{H} p_{\text{l.r.}}^{L} (1-q)^{M-Q} q^{Q}$$

where p, $p_{\text{h.r.}}$, $p_{\text{l.r.}}$ and q are the probabilities of any qubit error, a high-rate qubit error, a low-rate qubit error, and a measurement error, respectively. Noting that N and M are constant for a fixed number of time steps, we have

$$\log \Pr(E) = H \log \frac{p_{\text{h.r.}}}{1-p} + L \log \frac{p_{\text{l.r.}}}{1-p} + Q \log \frac{q}{1-q} + k,$$

where k is a constant independent of the error configuration. Recalling, from Section 4.5, that $p_{\text{h.r.}} = p\eta/(\eta+1)$ and $p_{\text{l.r.}} = p/(2(\eta+1))$, where η is the noise bias, we see that the time, parallel and diagonal step weights, as functions of η, p and q can be defined, respectively, as

$$\begin{aligned}
\mu_t &= -\log[q/(1-q)] \\
\mu_p &= -\log[\eta/(\eta+1)] - \log[p/(1-p)] \\
\mu_d &= -\log[1/(2(\eta+1))] - \log[p/(1-p)].
\end{aligned}$$

Examples showing the operation of the main decoding algorithm, assuming reliable measurements, can be seen in Figures 4.6, 4.7 and 4.8. Each figure shows: (a) an error and corresponding syndrome defects; (b) vertical/horizontal nodes added to the graph for the given syndrome defects; (c) matching of nodes resulting from MWPM over the graph; (d) cluster of defects corresponding to the matching; and (e) recovery operator resulting from neutralizing the defects in the cluster. Figure 4.6 shows the successful correction of an infinite bias error on a periodic lattice. Figure 4.7 shows the successful correction of an infinite bias error on a lattice with boundaries. Figure 4.8 shows the successful correction of an error due to finite bias noise on a periodic lattice.

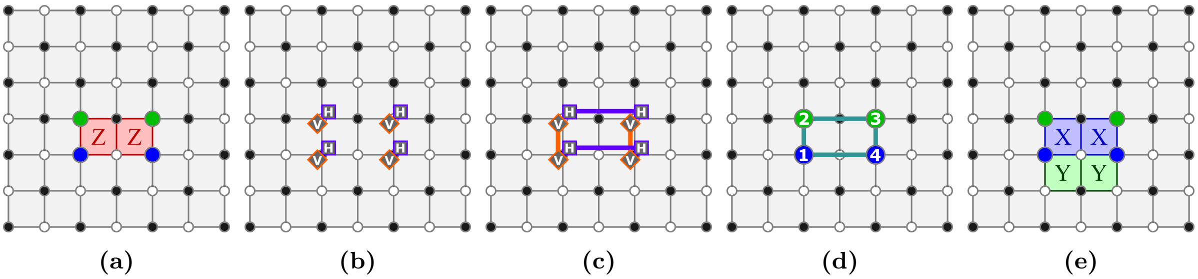


Figure 4.6: Example of successful correction of an infinite bias (i.e. pure Z) error on a periodic lattice. (a) Error and corresponding syndrome defects. (b) Vertical/Horizontal graph nodes; both V and H nodes are added for each syndrome defect. (c) Matching from MWPM; matching is allowed between similarly oriented nodes. (d) Cluster of defects from following matched pairs of nodes. (e) Recovery operator from fusing defects around cluster; the recovery is equivalent to the original error, up to a stabilizer, and so successfully corrects the error.

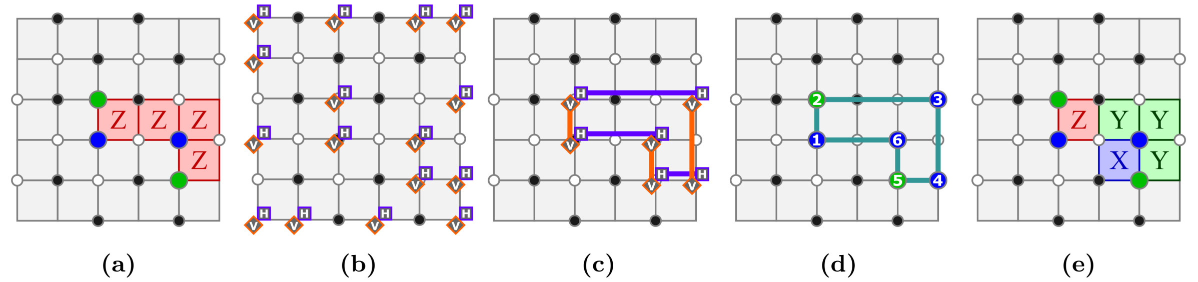


Figure 4.7: Example of successful correction of an infinite bias (i.e. pure Z) error on a lattice with boundaries. (a) Error and corresponding syndrome defects. (b) Vertical/Horizontal graph defects; both V and H nodes are added for each syndrome defect, as well as virtual V and H nodes for each boundary vertex where a stabilizer is not applied. (c) Matching from MWPM; matching is allowed between similarly oriented nodes, and between virtual nodes at the same location (not shown here since such matchings are not used in the construction of clusters). (d) Cluster of defects from following matched pairs of nodes. (e) Recovery operator from fusing defects around cluster; the recovery is equivalent to the original error, up to a stabilizer, and so successfully corrects the error.

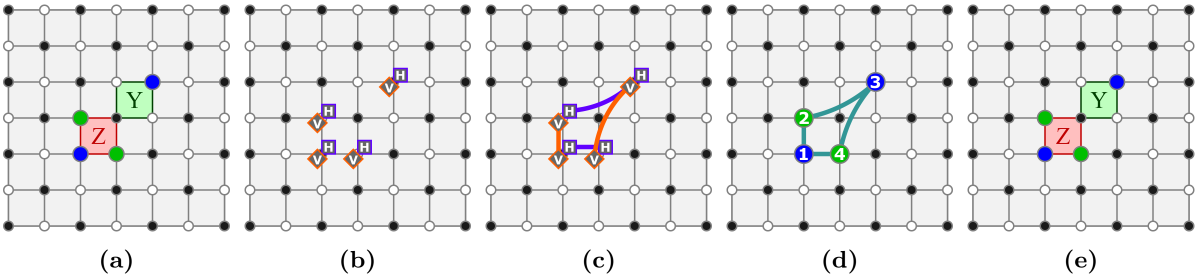


Figure 4.8: Example of successful correction of a finite bias (i.e. high-rate Z and low-rate X or Y) error on a periodic lattice. (a) Error and corresponding syndrome defects. (b) Vertical/Horizontal graph defects; both V and H nodes are added for each syndrome defect, as well as virtual V and H nodes for each vertex where a stabilizer is not applied. (c) Matching from MWPM; matching is allowed between similarly oriented nodes but penalized between distinct rows or columns as a function of the noise parameters. (d) Cluster of defects from following matched pairs of nodes. (e) Recovery operator from fusing defects around cluster; the recovery successfully corrects the original error.

```
Function decode:
    // syndrome is a set of defects where a defect is a (vertex location, type) tuple
    // noise parameters are bias, qubit and measurement error rates, respectively
    Input: syndrome, noise parameters: $\eta$, $p$, $q$
    // recovery is set of Pauli operators with qubit locations
    Output: recovery
    begin construct graph
        foreach defect in syndrome do
            add vertical and horizontal nodes to graph;
        foreach boundary vertex at each time step where no stabilizer is applied do
            add vertical and horizontal virtual nodes to graph connected by a zero
              weight edge;
        foreach vertical node pair, horizontal node pair in graph do
            add edge to graph weighted by distance(node pair, $\eta$, $p$, $q$);
    find matching from graph using MWPM;
    begin construct clusters
        // clusters is a set of clusters where a cluster is an ordered list of defects
        remove matches between virtual nodes with same location from matching;
        // all matches are now pairs of vertical or horizontal nodes
        while matching not empty do
            begin construct cluster
                let $\alpha$ reference any vertical node in matching;
                repeat
                    select and remove vertical node pair ($\alpha$, $\beta$) from matching;
                    add defect corresponding to $\alpha$ to cluster;
                    select and remove horizontal node pair ($\beta$, $\gamma$) from matching;
                    add defect corresponding to $\beta$ to cluster;
                    let $\alpha$ reference $\gamma$;
                until select from matching fails;
                label cluster neutral (charged) if even (odd) parity of single-type defects;
                add cluster to clusters;
    begin construct recovery
        foreach cluster in clusters do
            add shortest path of $Y$ ($X$) operators to recovery between successive $X$-type
              ($Y$-type) defect pairs in cluster (ignoring time dimension);
        Update recovery with output of residual-decode(clusters);
    return recovery;
```

Algorithm 4.1: Main decoding algorithm

```
Function distance:
    // node is a (vertex location, type) tuple with orientation and virtuality labels
    // noise parameters are bias, qubit and measurement error rates, respectively
    Input: node pair ($\alpha$, $\beta$), noise parameters: $\eta$, $p$, $q$
    // weight is a real number
    Output: weight
    // break path between nodes into time, parallel and diagonal steps
    // parallel refers to horizontal (vertical) steps between horizontal (vertical) nodes
    $\delta_t \leftarrow$ minimum number of time steps between $\alpha$ and $\beta$;
    $\delta_p \leftarrow$ minimum number of parallel steps between $\alpha$ and $\beta$;
    $\delta_d \leftarrow$ minimum number of perpendicular steps between $\alpha$ and $\beta$;
    if $\delta_p \geq \delta_d$ then
        $\delta_p \leftarrow \delta_p - \delta_d$;
    else
        $\delta_p \leftarrow (\delta_d - \delta_p) \bmod 2$;
    // define time, parallel and diagonal step weights
    $\mu_t = -\log[q/(1-q)]$;
    $\mu_p = -\log[\eta/(\eta+1)] - \log[p/(1-p)]$;
    $\mu_d = -\log[1/(2(\eta+1))] - \log[p/(1-p)]$;
    return $\delta_t\mu_t + \delta_p\mu_p + \delta_d\mu_d$;
```

Algorithm 4.2: Distance function

```
Function residual-decode:
    // clusters is a set of cluster where each cluster is an ordered list of defects
    Input: clusters
    // recovery is set of Pauli operators with qubit locations
    Output: recovery
    begin construct graph
        foreach cluster in clusters do
            if cluster is charged then
                add cluster to graph;
            if cluster is neutral and contains $X$-type and $Y$-type defects then
                add two copies of cluster to graph connected by a zero weight edge;
        foreach corner vertex at each time step where no stabilizer is applied do
            add virtual cluster to graph;
        foreach cluster pair in graph do
            add edge to graph weighted by minimum Manhattan distance between any
              pairing of defects drawn one from each cluster;
        if odd number of charged clusters then
            add virtual cluster to graph with zero weight edge to each virtual cluster;
    find matching from graph using MWPM;
    begin construct recovery
        remove matches where each cluster is virtual;
        foreach cluster pair in matching do
            add shortest path of $Y$ ($X$) operators to recovery between selected $X$-type
              ($Y$-type) defects from each cluster in pair;
    return recovery;
```

Algorithm 4.3: Residual decoding sub-algorithm

4.B Decoder runtime

In this section, we provide a brief analysis of the decoder runtime. The runtime of the decoder is determined by the runtime of the pairing subroutine. With our choice of minimum-weight perfect matching, a single subroutine has runtime $O(v^3)$, where v is the number of vertices of the input graph. On average, in the main decoding step, we have $v \sim 8pd^2$ for the ideal-measurement case and $v \sim 8pd^3$ for the fault-tolerant case, where p is the error rate and d is the code distance. We include a constant prefactor $8 = 2 \times 4$; the factor 4 is included because each error introduces at most four defects, and the factor 2 is included because for each defect we introduce two vertices to the graph. We neglect the vertices that allow us to pair defects to the boundary; for large system sizes, this contribution is negligibly small compared to the number of vertices that appear in the bulk.

In the finite-bias case, we use minimum-weight perfect matching again to pair charged clusters in a residual decoding step. This also scales like $O(d^2)$ and $O(d^3)$ for the ideal-measurement and fault-tolerant cases, respectively. However, in practice, this residual step is very fast compared to the main decoding step.

We note that we can choose any pairing subroutine to combine defects. For instance, we could use the union-find decoder due to Delfosse and Nickerson [32] to reduce the runtime from $O(v^3)$ to almost v.

With periodic boundary conditions and at infinite noise bias, decoding is readily parallelized into several two-dimensional pairing subroutines [31]. Moreover, in the infinite bias case, the residual decoding step is not required. We can therefore reduce the single pairing problem of $v \sim 8pd^3$ vertices in the fault-tolerant case into $2d$ subroutines of $v \sim 2pd^2$ vertices. One may consider ways to parallelize the decoder in settings where we include boundaries or when the noise bias is finite. We leave this to future work.

References

[1] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, **303**, p. 2, (2003).

[2] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, **43**, p. 4452, (2002).

[3] B. M. Terhal, "Quantum error correction for quantum memories," *Rev. Mod. Phys.*, **87**, p. 307, (2015).

[4] B. J. Brown, D. Loss, J. K. Pachos, C. N. Self, and J. R. Wootton, "Quantum memories at finite temperature," *Rev. Mod. Phys.*, **88**, p. 045005, (2016).

[5] E. T. Campbell, B. M. Terhal, and C. Vuillot, "Roads towards fault-tolerant universal quantum computation," *Nature*, **549**, p. 172, (2017).

[6] C. Wang, J. Harrington, and J. Preskill, "Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory," *Ann. Phys.*, **303**, p. 31, (2003).

[7] R. Raussendorf and J. Harrington, "Fault-tolerant quantum computation with high threshold in two dimensions," *Phys. Rev. Lett.*, **98**, p. 190504, (2007).

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

[9] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, "Surface code quantum computing by lattice surgery," *New J. Phys.*, **14**, p. 123011, (2012).

[10] B. J. Brown, K. Laubscher, M. S. Kesselring, and J. R. Wootton, "Poking holes and cutting corners to achieve Clifford gates with the surface code," *Phys. Rev. X*, **7**, p. 021029, (2017).

[11] B. J. Brown, "A fault-tolerant non-Clifford gate for the surface code in two dimensions," *arXiv:1903.11634*, (2019).

[12] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffry, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, **508**, p. 500, (2014).

[13] A. D. Córcoles, E. Magesan, S. J. Srinivasan, A. W. Cross, M. Steffen, J. M. Gambetta, and J. M. Chow, "Demonstration of a quantum error detection code using a square lattice of four superconducting qubits," *Nat. Comms.*, **6**, p. 6979, (2015).

[14] J. Kelly, R. Barrends, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, I.-C. Hoi, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, A. V. J. Wenner, A. N. Cleland, and J. M. Martinis, "State preservation by repetitive error detection in a superconducting quantum circuit," *Nature*, **519**, p. 66, (2015).

[15] M. Takita, A. D. Córcoles, E. Magesan, B. Abdo, M. Brink, A. W. Cross, J. M. Chow, and J. M. Gambetta, "Demonstration of weight-four parity measurements in the surface code architecture," *Phys. Rev. Lett.*, **117**, p. 210505, (2016).

[16] G. Duclos-Cianci and D. Poulin, "Fast decoders for topological quantum codes," *Phys. Rev. Lett.*, **104**, p. 050504, (2010).

[17] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, **86**, p. 032324, (2012).

[18] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Phys. Rev. A*, **90**, p. 032326, (2014).

[19] A. S. Darmawan and D. Poulin, "Linear-time general decoding algorithm for the surface code," *Phys. Rev. E*, **97**, p. 051302, (2018).

[20] J. Edmonds, "Paths, trees and flowers," *Canad. J. Math.*, **17**, p. 449, (1965).

[21] V. Kolmogorov, "Blossom V: a new implementation of a minimum cost perfect matching algorithm," *Math. Prog. Comp.*, **1**, p. 43, (2009).

[22] T. Ohno, G. Arakawa, I. Ichinose, and T. Matsui, "Phase structure of the random-plaquette Z_2 gauge model: accuracy threshold for a toric quantum memory," *Nucl. Phys. B*, **697**, p. 462, (2004).

[23] A. Kubica, M. E. Beverland, F. Brandão, J. Preskill, and K. M. Svore, "Three-dimensional color code thresholds via statistical-mechanical mapping," *Phys. Rev. Lett.*, **120**, p. 180501, (2018).

[24] J. R. Wootton and D. Loss, "High threshold error correction for the surface code," *Phys. Rev. Lett.*, **109**, p. 160503, (2012).

[25] A. G. Fowler, "Optimal complexity correction of correlated errors in the surface code," *arXiv:1310.0863*, (2013).

[26] A. Hutter and D. Loss, "Breakdown of surface-code error correction due to coupling to a bosonic bath," *Phys. Rev. A*, **89**, p. 042334, (2014).

[27] N. H. Nickerson and B. J. Brown, "Analysing correlated noise in the surface code using adaptive decoding algorithms," *Quantum*, **3**, p. 131, (2019).

[28] A. G. Fowler, A. C. Whiteside, A. L. McInnes, and A. Rabbani, "Topological code autotune," *Phys. Rev. X*, **2**, pp. 041003–, (2012).

[29] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, "Ultrahigh error threshold for surface codes with biased noise," *Phys. Rev. Lett.*, **120**, p. 050505, (2018).

[30] D. K. Tuckett, A. S. Darmawan, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, "Tailoring surface codes for highly biased noise," *Phys. Rev. X*, **9**, p. 041031, (2019).

[31] B. J. Brown and D. J. Williamson, "Parallelized quantum error correction with fracton topological codes," *Phys. Rev. R*, **2**, p. 013303, (2020).

[32] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," *arXiv:1709.06218*, (2017).

[33] R. Raussendorf, J. Harrington, and K. Goyal, "A fault-tolerant one-way quantum computer," *Ann. Phys.*, **321**, p. 2242, (2006).

[34] C. Vuillot, L. Lao, B. Criger, C. G. Almudéver, K. Bertels, and B. M. Terhal, "Code deformation and lattice surgery are gauge fixing," *New J. Phys.*, **21**, p. 033028, (2019).

[35] P. Aliferis and J. Preskill, "Fault-tolerant quantum computation against biased noise," *Phys. Rev. A*, **78**, p. 052331, (2008).

[36] S. Puri, L. St-Jean, J. A. Gross, A. Grimm, N. E. Frattini, P. S. Iyer, A. Krishna, S. Touzard, L. Jiang, A. Blais, S. T. Flammia, and S. M. Girvin, "Bias-preserving gates with stabilized cat qubits," *arXiv:1905.00450*, (2019).

[37] C. T. Chubb and S. T. Flammia, "Statistical mechanical models for quantum codes with correlated noise," *arXiv:1809.10704*, (2019).

[38] M. E. Beverland, B. J. Brown, M. J. Kastoryano, and Q. Marolleau, "The role of entropy in topological quantum error correction," *J. Stat. Mech.:Theor. Exp.*, **2019**, p. 073404, (2019).

[39] A. M. Stephens, W. J. Munro, and K. Nemoto, "High-threshold topological quantum error correction against biased noise," *Phys. Rev. A*, **88**, p. 060301(R), (2013).

[40] X. Xu, Q. Zhao, X. Yuan, and S. C. Benjamin, "High-threshold code for modular hardware with asymmetric noise," *Phys. Rev. Applied*, **12**, p. 064006, (2019).

[41] N. Delfosse, "Decoding color codes by projection onto surface codes," *Phys. Rev. A*, **89**, p. 012317, (2014).

[42] A. Kubica and N. Delfosse, "Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders," *arXiv:1905.07393*, (2019).

5 Conclusion

In this thesis we considered the surface code, one of the most promising and practical quantum error correction codes, in the context of noise biased towards dephasing (Z-biased noise), the dominant noise model in many quantum architectures. Our main analytical result revealed a hidden structure that makes the surface code particularly resilient to specific noise models. We showed how to tailor the code to apply that resilience to Z-biased noise, and how to exploit the structure to define efficient decoders. Our main numerical results demonstrated exceptionally high thresholds and low logical error rates for tailored surface codes with Z-biased noise.

We started with a numerical analysis of the standard surface code with Y-biased noise and demonstrated, in Chapter 2, that, with a small modification the surface code, achieves ultrahigh code-capacity thresholds with Z-biased noise; a result that we further refined in Chapter 3. The optimal code-capacity threshold of the standard surface code with pure X or Z noise is $\sim 11\%$ and with depolarizing noise it is $\sim 19\%$. With a bias of $\eta = 100$, where Z errors are 100 times more likely than X or Y errors, we estimate the optimal code-capacity threshold of the tailored surface code to be $\sim 39\%$ and with pure Z noise to be $\sim 50\%$. We provide strong evidence that the code-capacity threshold of the tailored surface code tracks the hashing bound for all biases. The hashing bound is provably achievable for quantum codes but the proof invokes random codes, so this is a remarkable result for a practical two-dimensional topological code with local stabilizers.

Next, in Chapter 3, we revealed, a hidden structure of the tailored surface code with pure Z noise, or equivalently the standard surface code with pure Y noise, that is responsible for these ultrahigh thresholds. We showed that, in the limit of pure Z noise, the tailored surface code is equivalent to a concatenation of classical codes: a top-level cycle code and a number of bottom-level repetition codes. As a consequence of this structure, we proved that the code-capacity threshold of the tailored surface code with pure Z noise is 50%. As a further consequence, we showed that the distance of the tailored surface code to Z errors, and the number of failure modes, can be tuned by modifying the boundary of the code. In particular, for coprime codes, whose linear dimensions are relatively prime, and closely-related rotated codes, there is a single Z-type logical operator and it has weight $O(n)$; by contrast, for square codes, the number of Z-type logical operators is $O(2^{\sqrt{n}})$ and they have minimum weight $O(\sqrt{n})$, where n is the number of physical qubits. We demonstrated that these characteristics enable tailored coprime and rotated codes to achieve significant improvements in terms of logical error rate with pure Z and Z-biased noise in comparison to square codes. We also defined an improved tensor-network decoder for biased noise and performed a comparative numerical analysis indicating that the color code does not exhibit an increase in threshold with bias.

Finally, in Chapter 4, we introduced an approach to decoding that exploits symmetries of a code with respect to a dominant noise model. Although the approach is suboptimal, it is efficient, using weighted matching with respect to the symmetries, and readily extends to the fault-tolerant context, where measurements are unreliable. We used the approach to define a decoder for the tailored surface code with Z-biased noise and demonstrated exceptionally high fault-tolerant thresholds. The optimal phenomenological fault-tolerant threshold of the standard surface code with pure X or Z noise is $\sim 3.3\%$. Using our decoder, we estimate a threshold for the tailored surface code of $\sim 5\%$ with bias $\eta = 100$, and exceeding 6% with pure Z noise. We noted that there is scope to achieve further improvements in fault-tolerant threshold with a more refined version of the decoder.

Looking forward, our work opens up many interesting avenues of research, some of which we briefly outline below.

In Chapters 2 and 3, we used two-dimensional tensor-network decoders with approximate contraction to estimate the optimal code-capacity thresholds of the tailored surface code with biased noise. With new developments in tensor network methods [1], it may be possible to contract three-dimensional tensor networks in a reasonable time with sufficient accuracy to estimate optimal fault-tolerant thresholds. If this is possible, it would provide insight into the inherent performance of the code in a fault-tolerant setting, as well as providing target threshold values for efficient heuristic decoders.

In Chapter 3, we identified previously unknown features of one of the most studied codes with biased noise. These insights, and the success we had in exploiting them, encourage investigation for analogous features with different noise models, such as spatial correlations, and in different codes, such as subsystem [2, 3] and single-shot codes [4, 5]. If we consider quantum architectures that support non-local stabilizers then constant-rate LDPC codes [6–9] are particularly interesting and may well have symmetries, with specific noise models, that could be exploited.

In Chapter 4, we found phenomenological fault-tolerant thresholds for the tailored surface code with biased noise using a suboptimal decoder. These results assume the availability of bias-preserving syndrome extraction circuits. Since the publication of our first paper, on ultrahigh thresholds with biased noise, the bias-preserving gates required for such circuits have been proposed for some architectures [10]. This motivates an investigation to establish circuit-based fault-tolerant thresholds with biased noise for specific physical implementations.

The research presented in this thesis is very much aligned with the inspiring, and hopefully not too distant [11], dream of realizing large-scale universal quantum computers. However, the motivation for quantum information research goes beyond this dream. Insights from the field of quantum information, and the subfield of quantum error correction, are advancing our understanding of fundamental physics [12, 13]. In my opinion, understanding quantum information is key to understanding the physical world, and I echo the words of David Deutsch [14]: “The quantum theory of computation must in any case be an integral part of the world-view of anyone who seeks a fundamental understanding of reality”.

References

[1] S.-J. Ran, E. Tirrito, C. Peng, X. Chen, L. Tagliacozzo, G. Su, and M. Lewenstein, "Lecture notes of tensor network contractions," arXiv:1708.09213, (2017).

[2] D. W. Kribs, R. Laflamme, D. Poulin, and M. Lesosky, "Operator quantum error correction," *Quantum Info. Comput.*, **6**, pp. 382–399, arXiv:quant-ph/0504189, (2006).

[3] D. Poulin, "Stabilizer formalism for operator quantum error correction," *Phys. Rev. Lett.*, **95**, p. 230504, arXiv:quant-ph/0508131, (2005).

[4] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, **43**, 9, pp. 4452–4505, arXiv:quant-ph/0110143, (2002).

[5] H. Bombín, "Single-shot fault-tolerant quantum error correction," *Phys. Rev. X*, **5**, p. 031043, arXiv:1404.5504, (2015).

[6] J. Tillich and G. Zémor, "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Trans. Inf. Theory*, **60**, pp. 1193–1202, arXiv:0903.0566, (2014).

[7] A. A. Kovalev and L. P. Pryadko, "Improved quantum hypergraph-product LDPC codes," in *2012 IEEE International Symposium on Information Theory Proceedings*, pp. 348–352, (2012), arXiv:1202.0928.

[8] M. H. Freedman and M. B. Hastings, "Quantum systems on non-k-hyperfinite complexes: A generalization of classical statistical mechanics on expander graphs," *Quantum Info. Comput.*, **14**, pp. 144–180, arXiv:1301.1363, (2014).

[9] M. B. Hastings, "Decoding in hyperbolic spaces: Quantum LDPCs codes with linear rate and efficient error correction," *Quantum Info. Comput.*, **14**, pp. 1187–1202, arXiv:1312.2546, (2014).

[10] S. Puri, L. St-Jean, J. A. Gross, A. Grimm, N. E. Frattini, P. S. Iyer, A. Krishna, S. Touzard, L. Jiang, A. Blais, S. T. Flammia, and S. M. Girvin, "Bias-preserving gates with stabilized cat qubits," arXiv:1905.00450, (2019).

[11] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, **2**, p. 79, arXiv:1801.00862, (2018).

[12] R. Jozsa, "Illustrating the concept of quantum information," *IBM Journal of Research and Development*, **48**, pp. 79–85, arXiv:quant-ph/0305114, (2004).

[13] A. Almheiri, X. Dong, and D. Harlow, "Bulk locality and quantum error correction in AdS/CFT," *Journal of High Energy Physics*, **2015**, p. 163, arXiv:1411.7041, (2015).

[14] D. Deutsch, The fabric of reality. Penguin UK, 1st ed., (1998). [Ch. 9, p. 219].