New methods in quantum error correction and fault-tolerant quantum computing

by

Christopher Chamberland

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

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Stephen Bartlett Professor of Physics, University of Sydney
Supervisor:	Raymond Laflamme Professor, Dept. of Physics and Astronomy, University of Waterloo
Committee Member:	Daniel Gottesman Professor, Perimeter Institute for Theoretical Physics
Internal Examiner:	Richard Cleve Professor, David R. Cheriton School of Computer Science, University of Waterloo
Committee Member:	Matteo Mariantoni Professor, Dept. of Physics and Astronomy, University of Waterloo

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

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

Abstract

Quantum computers have the potential to solve several interesting problems in polynomial time for which no polynomial time classical algorithms have been found. However, one of the major challenges in building quantum devices is that quantum systems are very sensitive to noise arising from undesired interactions with the environment. Noise can lead to errors which can corrupt the results of the computation. Quantum error correction is one way to mitigate the effects of noise arising in quantum devices.

With a plethora of quantum error correcting codes that can be used in various settings, one of the main challenges of quantum error correction is understanding how well various codes perform under more realistic noise models that can be observed in experiments. This thesis proposes a new decoding algorithm which can optimize threshold values of error correcting codes under different noise models. The algorithm can be applied to any Markovian noise model. Further, it is shown that for certain noise models, logical Clifford corrections can further improve a code's threshold value if the code obeys certain symmetries.

Since gates and measurements cannot in general be performed with perfect precision, the operations required to perform quantum error correction can introduce more errors into the system thus negating the benefits of error correction. Fault-tolerant quantum computing is a way to perform quantum error correction with imperfect operations while retaining the ability to suppress errors as long as the noise is below a code's threshold. One of the main challenges in performing fault-tolerant error correction is the high resource requirements that are needed to obtain very low logical noise rates. With the use of flag qubits, this thesis develops new fault-tolerant error correction protocols that are applicable to arbitrary distance codes. Various code families are shown to satisfy the requirements of flag fault-tolerant error correction. We also provide circuits using a constant number of qubits for these codes. It is shown that the proposed flag fault-tolerant method uses fewer qubits than previous fault-tolerant error correction protocols.

It is often the case that the noise afflicting a quantum device cannot be fully characterized. Further, even with some knowledge of the noise, it can be very challenging to use analytic decoding methods to improve the performance of a fault-tolerant scheme. This thesis presents decoding schemes using several state of the art machine learning techniques with a focus on fault-tolerant quantum error correction in regimes that are relevant to near term experiments. It is shown that even in low noise rate regimes and with no knowledge of the noise, noise can be further suppressed for small distance codes. Limitations of machine learning decoders as well as the classical resources required to perform active error correction are discussed.

In many cases, gate times can be much shorter than typical measurement times of quantum states. Further, classical decoding of the syndrome information used in quantum error correction to compute recovery operators can also be much slower than gate times. For these reasons, schemes where error correction can be implemented in a frame (known as the Pauli frame) have been developed to avoid active error correction. In this thesis, we generalize previous Pauli frame schemes and show how Clifford frame error correction can be implemented with minimal overhead. Clifford frame error correction is necessary if the logical component of recovery operators were chosen from the Clifford group, but could also be used in randomized benchmarking schemes.

Acknowledgements

First I would like to acknowledge Prof. Raymond Laflamme for his help, support and guidance throughout my PhD. Ray has provided a very vibrant research environment and allowed me to explore many different research avenues. Without this freedom I would never have been able to learn so many different aspects of quantum error correction and fault-tolerance.

I would like to thank Tomas Jochym-O'Connor for his mentorship at the beginning of my PhD. I learned so many new ideas, especially at an early stage in my career, which were crucial for many of the projects I worked on during my PhD.

I would like to thank David Poulin and Barbara Terhal for inviting me to visit Sherbrooke and Delft for several months to work on research ideas in fault-tolerance. These visits have been very rewarding and the exchanges of ideas pushed me in new directions expanding my understanding of fault-tolerance and error correction.

I would like to thank the QuArC group at Microsoft for giving me the opportunity to do an internship there during the summer of 2017. In particular, I would like to thank Michael Beverland for his support and mentorship during my internship. Michael was incredibly patient and would always take time to discuss new ideas which lead to a paper of which I am very proud of.

Throughout my PhD I was very fortunate to have several scientific and engaging discussions with very talent scientists. In particular, I would like to thank Joel Wallman, Debbie Leung, Nicolas Delfosse, Pavithran Iyer, Pooya Ronagh, Guillaume Verdon-Akzam, Ben Criger, Annie Park, Nikolas P. Breuckmann, Jonathan Conrad, Theerapat Tansuwannont, Thomas O'Brien, Christophe Vuillot, Daniel Weigand, Francesco Battisel, Xiatong Ni, Joel Klassen, Michael Chen, Hemant Katiyar, JunAn Lin, Brandon Buonacorsi, Jason Pye, Theodore Yoder and Ryuji Takagi.

Lastly, I would like to thank my parents for their constant support and encouragement throughout my PhD and for giving me the opportunity to receive the level of education that allowed me to be where I am today.

Dedication

To everyone who is passionate about building a quantum computer.

Table of Contents

List of Tables

List of Figures

Chapter 1

Brief review of error correction and fault-tolerance methods

With the advent of quantum algorithms which show that large numbers can be factored into their primes in polynomial time [8] and searching through large unstructured data sets can be done with a quadratic speedup over the best classical algorithms [9], quantum computers hold the promise to be a very powerful computational tool. However, quantum systems are very sensitive to interactions with their environment which can lead to errors potentially corrupting the computational data. One method that has been developed to mitigate the effects of errors is quantum error correction, which provides means to potentially detect and remove errors when they occur.

Roughly speaking, the threshold of an error correcting code corresponds to the physical error rate below which the logical failure rate is smaller than the physical error rate (more details will be provided in the following chapters). As will be seen in the chapters that follow, fault-tolerant thresholds are on the order of 10^{-4} to 10^{-2}. Furthermore, the best pseudo-thresholds obtained to date (relevant to near term fault-tolerant devices) are closer to the 10^{-4} range. Even if better fault-tolerant protocols are found, it will be very difficult to improve upon the current pseudo-thresholds of the most competitive fault-tolerant protocols. To date, only experiments demonstrating fault-tolerant error detection schemes have been performed, since the pseudo-thresholds for these schemes can be several orders of magnitude higher than pseudo-thresholds for fault-tolerant error correction schemes. Looking forward at the next five to ten years, I believe that the most important breakthroughs will need to occur at the hardware level in order to reduce gate fidelities by one to two orders of magnitude in order to perform the first fault-tolerant error *correction* experiments showing that logical qubits have longer lifetimes than physical qubits.

Asymptotically, the overhead cost of schemes for fault-tolerant quantum computation can be very large (easily above one billion qubits depending on the algorithm one wants to implement). Some progress has been made to reduce the overhead, but there are still many unanswered questions and room for significant improvements at the theoretical level. The flag qubit methods introduced in this thesis (Chapter 3) are designed to be a step forward in this direction. Further, flag qubits have many applications other than fault-tolerant error correction, such as providing the ability to fault-tolerantly prepare magic states using a very small number of qubits. This in turn could be one method to reduce the overhead of fault-tolerant quantum computation, but many other methods remain to be explored.

In order to implement error correction protocols, one needs to measure the error syndrome which gives partial information on the errors afflicting the system. Given the outcome of the syndrome measurements, one of the main challenges of error correction is to efficiently perform the classical processing (decoding) used to find the most likely error that actually occurred. In Chapter 2, we will focus on new decoding algorithms that achieve this goal by taking advantage of knowledge about the type of noise present in the quantum device. Our decoding algorithm is applicable to general Markovian noise channels.

In Sec. 1.1 we will begin by reviewing the stabilizer formalism, which lies at the foundation for all of the error correcting codes presented in this thesis. In Sec. 1.1.1, we will give an example of a stabilizer code, the 7-qubit Steane code, which can implement all logical Clifford gates transversally. We will also discuss the idea of transversal gates and present the Eastin-Knill theorem. In Sec. 1.2 we will introduce fault-tolerant methods to deal with noisy gates and measurements. New results in fault-tolerant error correction and fault-tolerant quantum computation will be introduced in Chapter 3 – Chapter 5.

1.1 Review of the stabilizer formalism

First, we begin by defining the n-qubit Pauli group as follows

Definition 1. *n-qubit Pauli group*

$$\mathcal{P}_n^{(1)} = \sigma\{I, X, Y, Z\}^{\otimes n}, \tag{1.1}$$

where X, Y and Z are the single-qubit Pauli matrices, given by

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

and $\sigma \in \{\pm 1, \pm i\}$.

Throughout this thesis we will sometimes make use of the *Clifford hierarchy* which is defined as

Definition 2. *Clifford hierarchy*

$$\mathcal{P}_n^{(k)} = \{U \,|\, UPU^\dagger \in \mathcal{P}_n^{(k-1)} \,\forall\, P \in \mathcal{P}_n^{(1)}\}. \tag{1.3}$$

For example, the Clifford group $\mathcal{P}_n^{(2)}$ can be shown to be generated by

$$\mathcal{P}_n^{(2)} = \langle H_i, S_i, \mathrm{CNOT}_{ij} \rangle, \tag{1.4}$$

where the Hadamard and Phase gates are given by

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \; S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \tag{1.5}$$

and the subscripts i and j indicate the qubits on which the operators will act.

Next, we define the weight of an error $E \in \mathcal{P}_n^{(1)}$ ($\mathrm{wt}(E)$) as the number of qubits on which it has non-trivial support. For instance, if $E = XIIYYZ$, then $\mathrm{wt}(E) = 4$.

Now, an $[\![n, k, d]\!]$ quantum error correcting code encoding k logical qubits into n physical qubits and that can correct all errors of weight $t = \lfloor (d-1)/2 \rfloor$ is the image space $\mathcal{H}_C$ of the injection $\xi : \mathcal{H}_2^k \to \mathcal{H}_C \subset \mathcal{H}_2^n$ where $\mathcal{H}_2$ is the two-dimensional Hilbert space.

Stabilizer codes are defined as quantum error correcting codes $\mathcal{H}_C$ which form the unique subspace of $\mathcal{H}_2^n$ that is fixed by an Abelian stabilizer group $\mathcal{S} \subset \mathcal{P}_n^{(1)}$ such that for any $s \in \mathcal{S}$ and any codeword $|c\rangle \in \mathcal{H}_C$, $s|c\rangle = |c\rangle$. In other words, the codewords are $+1$ eigenstates of all elements in the stabilizer group.

Any operator $s \in \mathcal{S}$ can be written as $s = g_1^{p_1} \cdots g_{n-k}^{p_{n-k}}$. The operators g_i are called the stabilizer generators which have the property that they mutually commute (since $\mathcal{S}$ is an Abelian group) and satisfy $g_i^2 = I$. Consequently, we can write $\mathcal{S} = \langle g_1, \cdots g_{n-k} \rangle$. We also define $N(\mathcal{S})$ to be the normalizer of the stabilizer group, that is

$$N(\mathcal{S}) = \{N \in \mathcal{P}_n^{(1)} \,|\, NM = MN, \forall\, M \in \mathcal{S}\}. \tag{1.6}$$

We can thus see that any non-trivial logical operator on codewords belongs to the set $N(\mathcal{S}) \backslash \mathcal{S}$. The distance d of an error correcting code is defined as the lowest weight operator $M \in N(\mathcal{S}) \backslash \mathcal{S}$. A more detailed overview of stabilizer codes can be found in [10, 11].

Next we define the notion of stabilizer error correction

$[\![7,1,3]\!]$ Steane code
$g_1 = Z_4Z_5Z_6Z_7$
$g_2 = Z_2Z_3Z_6Z_7$
$g_3 = Z_1Z_3Z_5Z_7$
$g_4 = X_4X_5X_6X_7$
$g_5 = X_2X_3X_6X_7$
$g_6 = X_1X_3X_5X_7$
$\overline{X} = X^{\otimes 7}, \overline{Z} = Z^{\otimes 7}$

Table 1.1: Stabilizer generators of the $[\![7,1,3]\!]$ Steane code. The last row corresponds to a representation for the logical X and logical Z operators.

Definition 3. *Stabilizer error correction*

Given a stabilizer group $\mathcal{S} = \langle g_1, \cdots, g_m \rangle$, we define the syndrome $s(E)$ to be a bit string, with i'th bit equal to zero if g_i and E commute, and one otherwise. Let $E_{min}(s)$ be a minimal weight correction E where $s(E) = s$. We say operators E and E' are logically equivalent, written as $E \sim E'$, iff $E' \propto gE$ for $g \in \mathcal{S}$.

With an error correcting code, the goal of a decoder is to find the most likely error E afflicting a system for a given syndrome measurement $s(E)$. As will be shown in this chapter and the next, decoding by applying the minimum weight error compatible with the measured syndrome s ($E_{\min}(s)$) will not always be optimal.

1.1.1 The $[\![7,1,3]\!]$ Steane code, CSS codes and the Eastin-Knill theorem

To put the above ideas together, in this section we will introduce the $[\![7,1,3]\!]$ Steane code which is part of a family of Calderbank-Shor-Steane (CSS) codes [12, 13]. CSS codes have a lot of useful features which we will discuss. Further, as will be shown, the Steane code can implement all logical gates in the Clifford group transversely. In fact, the Steane code is the smallest color code. Color codes are a family of codes with useful topological properties [5, 14, 15] that can implement all logical Clifford gates transversally (see below for the definition of transversal gates). A more detailed discussion of color codes will be given in Chapter 3.

The stabilizer generators of the Steane code are given in Table. 1.1. One can verify that all stabilizer generators commute amongst each other. The codewords for the Steane code are given by

$$|\overline{0}\rangle = \frac{1}{\sqrt{2^3}}(|0000000\rangle + |0110011\rangle + |1010101\rangle + |1100110\rangle + |0001111\rangle + |0111100\rangle + |1011010\rangle + |1101001\rangle), \tag{1.7}$$

$$|\overline{1}\rangle = \frac{1}{\sqrt{2^3}}(|1111111\rangle + |1001100\rangle + |0101010\rangle + |0011001\rangle + |1110000\rangle + |1000011\rangle + |0100101\rangle + |0010110\rangle). \tag{1.8}$$

One can verify that the states in Eq. 1.8 are +1 eigenstates of all operators in Table. 1.1. One can also verify that any weight-one or weight-two error anti-commutes with at least one of the stabilizers. However, the operators $X_L = X_1X_2X_3$ and $Z_L = Z_1Z_2Z_3$ commute with all the stabilizers and do not belong to the stabilizer group. Thus, the distance of the Steane code is three. Further, one can verify that $X_L|\overline{0}\rangle = |\overline{1}\rangle$, $X_L|\overline{1}\rangle = |\overline{0}\rangle$, $Z_L|\overline{0}\rangle = |\overline{0}\rangle$ and $Z_L|\overline{1}\rangle = -|\overline{1}\rangle$. Thus X_L and Z_L are representatives of the logical X and Z operators.

Next, we give the definition of a transversal operation

Definition 4. *Transversal operation*

A transversal operation

1. *Only applies single-qubit gates to qubits in a codeblock*
2. *Only interacts the i^{th} qubit in one codeblock with the i^{th} qubit in a different codeblock or ancilla block.*

Transversal operations have the property that they are inherently fault-tolerant (see Sec. 1.2 along with Def. 5) making such operations very desirable.

Now, consider the operator $H_L = H^{\otimes 7}$ which by Def. 4 is a transversal operation. Notice that $H_L X_L H_L^\dagger = Z_L$, $H_L Z_L H_L^\dagger = X_L$ and that $H_L g_i H_L^\dagger = g_{i+3}$ for all $i \in \{1, 2, 3\}$. Thus we see that $H^{\otimes 7}$ implements the logical Hadamard gate for the Steane code.

Next consider the operator $S^{\otimes 7}$. We have that $S^{\otimes 7} Z^{\otimes 7} (S^{\otimes 7})^\dagger = Z^{\otimes 7}$ and $S^{\otimes 7} X^{\otimes 7} (S^{\otimes 7})^\dagger = Y^{\otimes 7} = i^7 X^{\otimes 7} Z^{\otimes 7} = -Y_L$. Further one can verify that $S^{\otimes 7}$ maps stabilizers to elements of the stabilizer group. Thus we see that $S^{\otimes 7} = S_L^\dagger$ and not S_L. Conversely, a logical phase gate could be obtained by applying $S_L = (S^\dagger)^{\otimes 7}$.

Lastly, we consider a block-wise transversal CNOT gate. Doing so we have the following transformations

$$X_L \otimes I = (X \otimes I)^{\otimes 7} \to X^{\otimes 7} \otimes X^{\otimes 7} = X_L \otimes X_L \tag{1.9}$$
$$I \otimes X_L = (I \otimes X)^{\otimes 7} \to I \otimes X^{\otimes 7} = I \otimes X_L \tag{1.10}$$
$$Z_L \otimes I = (Z \otimes I)^{\otimes 7} \to Z^{\otimes 7} \otimes I = Z_L \otimes I \tag{1.11}$$
$$I \otimes Z_L = (I \otimes Z)^{\otimes 7} \to Z^{\otimes 7} \otimes Z^{\otimes 7} = Z_L \otimes Z_L \tag{1.12}$$

Since the logical X and Z operators under the application of a transversal CNOT transform the same way as the physical X and Z operators and that elements in the stabilizer group also map to elements in the stabilizer group on both code blocs, we see that $\text{CNOT}^{\otimes 7}$ performs a logical CNOT gate. Hence we conclude that when using the Steane code, all logical Clifford gates can be implemented transversally.

As mentioned above, the $[\![7, 1, 3]\!]$ Steane code is part of a family of CSS codes. CSS codes are constructed as follows. Let C_x be an $[n, k_x, d_x]$ classical linear code with parity check matrix H_x and C_z be an $[n, k_z, d_z]$ classical linear code with parity check matrix H_z and we will assume that $C_x^\perp \subseteq C_z$[1]. Then we can construct a quantum code C with the stabilizers given by the matrix (written in binary symplectic form)

$$S = \left(\begin{array}{c|c} H_x & 0 \\ 0 & H_z \end{array}\right). \tag{1.13}$$

In this case C is a $[\![n, k, d]\!]$ quantum code with $k = k_x + k_z - n$ and $d \geq \min\{d_x, d_z\}$. Note that $C_x^\perp \subseteq C_z$ is required to guarantee that all X-stabilizers commute with all Z-stabilizers. Further, since the number of stabilizer generators is always $n - k$ for an $[\![n, k, d]\!]$ quantum error correcting code, then $k = n - (n - k_x) - (n - k_z) = k_x + k_z - n$.

It turns out that the only codes where transversal CNOT gates performs a logical CNOT are CSS codes [11].

Coming back to the Steane code, we have seen that all logical Clifford gates can be implemented transversally. One could then contemplate the existence of a quantum error correcting code where all logical gates in a universal gate set could be implemented transversally. The Eastin-Knill theorem [16] shows that this is not possible

Theorem 1. *Eastin-Knill theorem*

The set of transversal gates for a given quantum error correcting code generates a finite group. Thus the set cannot be universal for quantum computation.

In Chapter 5, we will briefly discuss non-transversal methods to implement fault-tolerant logical gates.

[1]Here $C_x^\perp$ is the dual to C_x, that is the set of strings $w \in \{0, 1\}^n$ such that $w \cdot v = 0$ for all $v \in C_x$.

1.2 Fault-tolerant quantum error correction

In realistic quantum devices, gate and measurement errors can occur with much higher probability than idle qubit errors. Furthermore, if our circuits used to implement our error correction protocols are not chosen carefully, errors arising from a small number of faults can spread to higher weight errors which could potentially be no longer correctable by the code used to protect the data. Fault-tolerant quantum error correction schemes have been devised which guarantee that errors spread in a controlled way such that a small number of faults will not lead to non-correctable errors. We will give a more rigorous definition of *small* in Sec. 1.2.1.

In Sec. 1.2.1, we will review fault-tolerant error correction (EC) schemes with a particular focus on Shor-EC [17] since the ideas presented there will be useful for understanding the material that follows. In Chapter 3, we will present a new fault-tolerant error correction protocol making use of flag-qubits and which is applicable to arbitrary distance codes. As will be shown, flag qubits allow fault-tolerant error correction schemes to be implemented using very few qubits making them competitive for near-term fault-tolerant implementations. Further, since our flag fault-tolerant schemes uses fewer qubits than Shor-EC (more details will be provided below), our scheme could be useful for large quantum computations with LDPC codes.

1.2.1 Overview of fault-tolerant error correction using Shor-EC

Before describing any fault-tolerant protocol, it is important to have a rigorous definition of fault-tolerance. The one that we will use here follows from [11, 18]:

Definition 5. *Fault-tolerant error correction*

For $t = \lfloor (d-1)/2 \rfloor$, an error correction protocol using a distance-d stabilizer code C is t-fault-tolerant if the following two conditions are satisfied:

1. *For an input codeword with error of weight s_1, if s_2 faults occur during the protocol with $s_1 + s_2 \leq t$, ideally decoding the output state gives the same codeword as ideally decoding the input state.*

2. *For s faults during the protocol with $s \leq t$, no matter how many errors are present in the input state, the output state differs from a codeword by an error of at most weight s.*

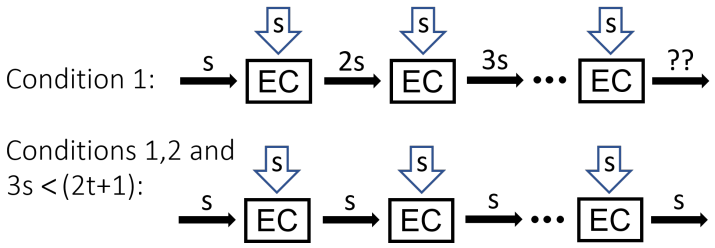


Figure 1.1: An example showing the first fault tolerance condition alone in Def. 5 is not sufficient to imply a long lifetime. We represent s faults occurring during a round of error correction with a vertical arrow, and a state a distance r from the desired codeword with a horizontal arrow with r above. The first condition alone allows errors to build up over time as in the top figure, which would quickly lead to a failure. However provided $s < (2t+1)/3$, both conditions together ensure that errors in consecutive error correction rounds do not build up, provided each error correction round introduces no more than s faults, which could remain true for a long time.

In Def. 5, ideally decoding refers to performing a round of fault-free error correction (where no additional faults can occur). Note that in what follows we also allow the input error to be in a superposition of errors of weight at most s_1. The first condition in Def. 5 guarantees that for codes which can correct errors of weight at most t, if there are at most s_2 faults when the input error is of weight s_1 with $s_1 + s_2 \leq t$, errors can only spread in such a way that the output state of the error correction (EC) protocol cannot differ from the input codeword by an error of weight at most t. In other words, if the input state was $E_{\text{in}}|\overline{\psi}\rangle$ where $\text{wt}(E_{\text{in}}) = s_1 \leq t$ and $|\overline{\psi}\rangle \in \mathcal{H}_C$, then assuming that there are at most s_2 faults during the EC protocol with $s_1 + s_2 \leq t$, the output codeword is $E_{out}|\overline{\psi}\rangle$ with $\text{wt}(E_{\text{out}}) \leq t$.

The second condition in Def. 5 is less trivial but nonetheless important to control how errors spread in between different EC rounds (see the example shown below). In this case, if the input state is given by $E_{\text{in}}|\overline{\psi}\rangle$ where the weight of E_{in} can be arbitrary, then assuming there are at most $s \leq t$ faults during the EC protocol, the output state must be given by $E_{\text{out}}|\overline{\phi}\rangle$ where $\text{wt}(E_{\text{out}}) \leq s$ and $|\overline{\phi}\rangle \in \mathcal{H}_C$. The key here is that $|\overline{\psi}\rangle$ does not necessarily have to be equal to $|\overline{\phi}\rangle$ ($|\overline{\phi}\rangle$ can be any codeword).

To give further motivation as to why the second condition of Def. 5 is important, consider a scenario with s faults introduced during each round of EC, and assume that $t/n < s < (2t+1)/3$ for some integer n (see Fig. 1.1). Consider an error correction protocol in which r input errors and s faults in an EC block leads to an output state with at most

$r + s \leq t$ errors[2]. Clearly condition 1 is satisfied.

With the above considerations, an input state $E_1|\bar{\psi}\rangle$ with $\mathrm{wt}(E_1) \leq s$ is taken to $E_2|\bar{\psi}\rangle$, with $\mathrm{wt}(E_2) \leq 2s$ by one error correction round with s faults. After the jth round, the state will be $E_j|\bar{\psi}\rangle$ with the first condition implying $\mathrm{wt}(E_j) \leq j \cdot s$ provided that $j \leq n$. However, when $j > n$, the requirement of the first condition is no longer satisfied so we cannot use it to upper bound $\mathrm{wt}(E_j)$. Now consider the same scenario but assuming both conditions hold. The second condition implies that after the first round, the input state $E_1|\bar{\psi}\rangle$ becomes $E_2'|\bar{\phi}\rangle = E_2|\bar{\psi}\rangle$, with $\mathrm{wt}(E_2') \leq s$, and where $|\bar{\phi}\rangle$ is a codeword. Therefore the codewords are related by: $|\bar{\phi}\rangle = (E_2'^{\dagger}E_2)|\bar{\psi}\rangle$, with logical operator $(E_2'^{\dagger}E_2)$ having weight at most $3s$, since $\mathrm{wt}(E_2) + \mathrm{wt}(E_2') \leq 3s$. However, the minimum non-trivial logical operator of the code has weight $(2t+1) > 3s$, implying that $|\bar{\psi}\rangle = |\bar{\phi}\rangle$, and therefore that $\mathrm{wt}(E_2) = \mathrm{wt}(E_2') \leq s$. Hence, for the jth round, $\mathrm{wt}(E_j) \leq s$ for all j, i.e. the distance from the codeword is not increased by consecutive error correction rounds with s faults, provided $s < (2t+1)/3$.

Note that it is possible to have less constrained definitions of fault-tolerance. For example, see Ref.[19] which uses a more relaxed version of the first condition in Def. 5. However, in this chapter we will consistently use Def. 5 for all of our fault-tolerant schemes.

Next we will consider the construction of an EC unit which satisfies Def. 5 using the method of Shor-EC. First, consider the circuit of Fig. 1.2 for measuring the operator $P = P_1 \otimes P_2 \otimes \cdots \otimes P_n$. If the error E afflicting the encoded data anti-commutes with P, then the X-basis measurement of the $|+\rangle$ ancilla gives -1 whereas if E commutes with P the measurement yields $+1$. Thus this circuit could in principle be used to measure each stabilizer generator in order to obtain the error syndrome. Notice however that if a single fault occurs on any of the two-qubit gates (apart from the last), this can result in an error of weight greater than one on the encoded data. Hence, the circuit in Fig. 1.2 is clearly not fault-tolerant. There is also the problem that a single Z error occurring somewhere on the ancilla qubit can change the parity of the measurement outcome. Hence, if such a circuit was used to measure all the stabilizer generators, a single fault could result in the wrong error syndrome.

In order to deal with the above issues, we first consider the spread of errors arising from the two-qubit gates. The spread of errors arise from the fact that the two-qubit gates are not transversal. However, if instead of initializing the ancilla qubit in the $|+\rangle$ state it was initialized in an n-qubit cat state (when $P = P_1 \otimes P_2 \otimes \cdots \otimes P_n$), then the two-qubit gates

[2]This is the case for FTEC protocols such as Shor, Steane and Knill EC (which will be introduced later in this thesis) with appropriately verified ancilla states. However the surface code does not satisfy this due to hook errors but nonetheless still satisfies condition 1 of Def. 5.

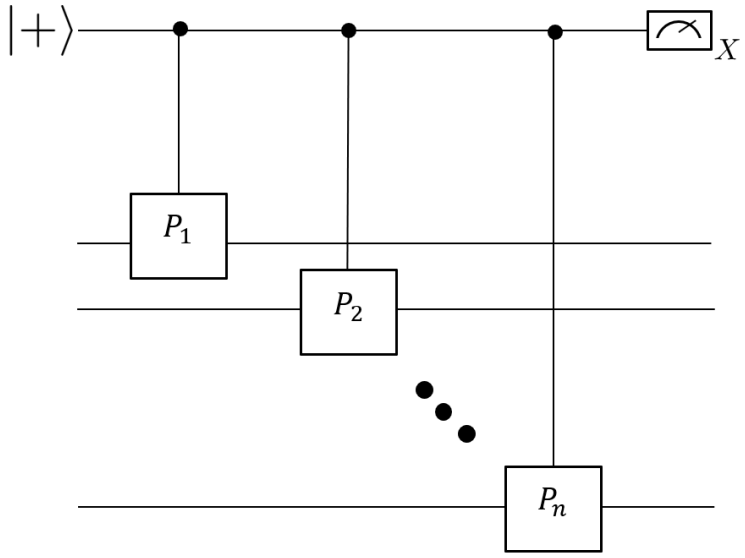


Figure 1.2: Non fault-tolerant circuit for measuring the operator $P = P_1 \otimes P_2 \otimes \cdots \otimes P_n$. The data qubit is a $+1$ eigenstate of P and $|+\rangle$ is the ancilla qubit used to perform the measurement. The j^{th} two-qubit gate is a controlled P_j gate (applies P_j if the control qubit is in the $|1\rangle$ state and applies the identity if the control qubit is in the $|0\rangle$ state).

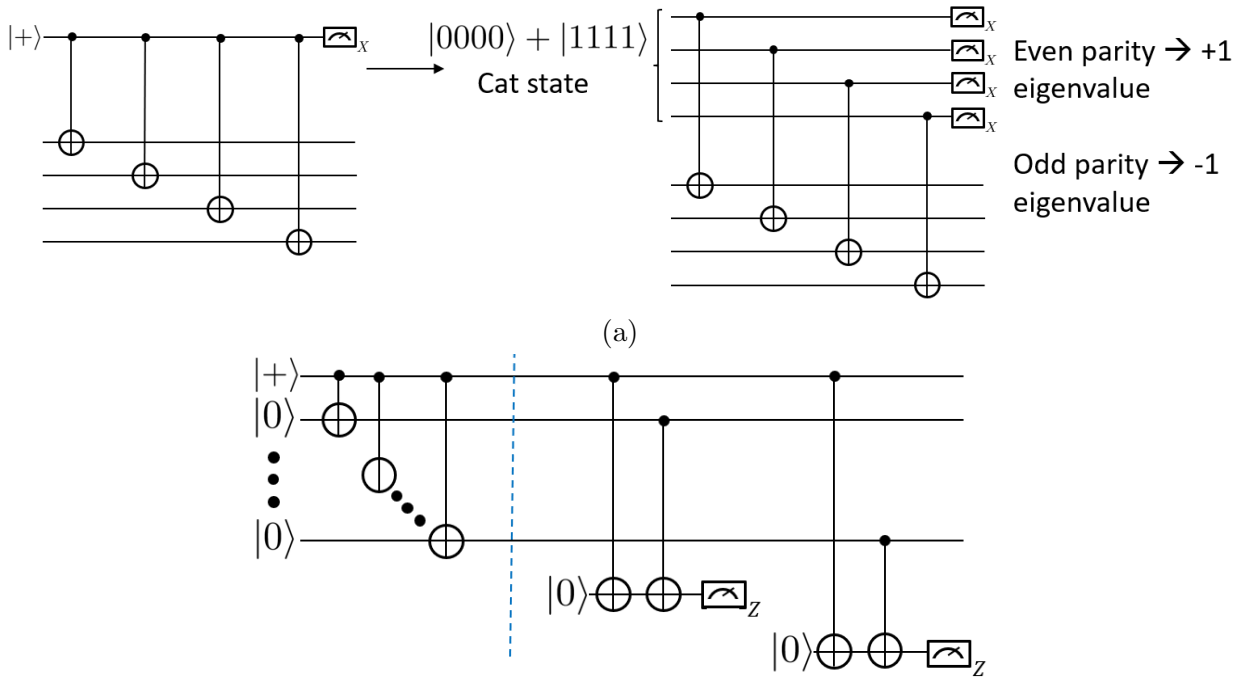


Figure 1.3: In Fig. 1.3a, we consider replacing the circuit in Fig. 1.2 (with $P = X^{\otimes 4}$) by a circuit with an equivalent action but with the $|+\rangle$ ancilla replaced by a four qubit cat state. In Fig. 1.3b, the cat state is prepared in a fault-tolerant way by measuring pairs of qubits in the Z basis. If any of the measurements result in a -1 outcome, the cat state is rejected and the protocol starts anew.

could be implemented transversally as in Fig. 1.3a. In this case, the circuit has the same effect as the one in Fig. 1.2. If the transversal X-basis measurement results in an even parity outcome, then the error afflicting the data qubits must commute with P. Otherwise, the error anti-commutes with P.

The advantage of using the circuit in Fig. 1.3a is that due to the transversal nature of of the two-qubit gates, a single fault at one of the two-qubit gate locations can only cause a data qubit error E with $\text{wt}(E) \leq 1$. In general, if there are s faults occurring at two-qubit gate locations, then the resulting data qubit error E will have weight at most s.

However, there is still the problem that if the cat state is not prepared in a fault-tolerant way, a single fault arising during the preparation of the cat state could potential spread to multiple data qubits (even with the transversal implementation of the two-qubit gates). Several methods have been developed to address this issue. In Ref.[20], the authors design specialized circuits which makes it possible to identify and remove high-weight errors occurring during the preparation of the cat state. This method is referred to as ancilla decoding. Another method is to measure random pairs of qubits in the Z basis as illustrated in Fig. 1.3b. The pairs can be chosen such that if s faults occur during the preparation of the cat state leading to an error of weight greater than s, at least one of the measurement outcomes will be -1 instead of $+1$. In such a case, the cat state is rejected and the process restarts until all measurement outcomes are $+1$. This method will be referred to as ancilla verification.

Lastly, we must deal with potential Z errors occurring on the ancilla qubits that can change the parity of the measurement outcome. As mentioned previously, a single Z error can change the parity of the measurement resulting in the wrong error syndrome. The key insight is to repeat the syndrome measurement enough times to guarantee that if the weight of the input error and number of faults during the EC round does not exceed t, then the conditions of Def. 5 will be satisfied.

Suppose that one obtains the same syndrome measurement s (measurement of all stabilizer generators using the circuits in Fig. 1.3) $t+1$ times in a row. Since we are assuming there can be at most t faults, correcting using the measured syndrome will satisfy Def. 5. To see this, note that if there were measurement errors resulting in the wrong syndrome, at least one of the measured syndromes would be different (and so one would not obtain the same syndrome $t+1$ times in a row). Further, there could in principle be $\leq t$ faults leading to errors which go undetected. However, the syndrome s will include all errors that were detected by the syndrome measurement, and since the errors are additive (due to the transversal nature of the circuit), correcting using s will result in a state that differs from a codeword by an error of weight at most t.

The last thing we need to address is how many syndrome measurement repetitions are required to guarantee that the syndrome is repeated $t+1$ times in a row. The answer is $(t+1)^2$ (although in Chapter 3 we will show that $\frac{1}{2}(t^2+3t+2) = \binom{t+2}{2}$ repetitions are sufficient). To see this, consider the worst case scenario where the same syndrome is repeated t times in a row without faults and a fault occurs on the $t+1$'th syndrome measurement resulting in a new syndrome outcome. We must then repeat the syndrome measurement an additional $t+1$ times. Suppose that again one obtains the same syndrome outcomes t times in a row without faults and a fault occurs on the $t+1$'th syndrome measurement resulting in a new error syndrome. Since we are assuming there can be at most t faults (by Def. 5), this sequence of t consecutive syndromes without faults plus a single fault on the $t+1$'th syndrome measurement can repeat at most t times. Afterwords, by our assumption, there can be no additional faults. Thus the next $t+1$ syndrome measurements are guaranteed to be identical. Hence the total number of times the syndrome was repeated is $t(t+1)+t+1 = (t+1)^2$.

Putting together all of the ideas presented in this section, we can now outline Shor's EC protocol:

Shor-EC Protocol:

1. Use the circuits of Fig. 1.3 to measure all of the stabilizer generators.
2. Repeat the syndrome measurement $(t+1)^2$ times in a row.
3. Use the last syndrome that is repeated $t+1$ times in a row. If no syndrome is repeated $t+1$ times, use that syndrome that was repeated the most times.

In practice there can be more than t faults which occur during Shor's EC protocol. In such as case, if there were no syndromes that repeated $t+1$ times in a row, one would use the syndrome that is repeated the largest number of times in order to have a complete protocol. Note that if more than t faults occur, we cannot expect the fault-tolerant protocol to succeed since there are some errors of weight greater than t that the underlying quantum error correcting code cannot correct.

We conclude this section with the remark that the number of qubits used in the cat states of Fig. 1.3 is equal to the weight of the stabilizer that is being measured. In Chapter 3 we will present a new fault-tolerant EC protocol which uses fewer qubits than Shor-EC.

Chapter 2

Hard decoding algorithm for optimizing thresholds under general Markovian noise

The material of this chapter is based on the journal article of Ref. [21], copyrighted by the American Physical Society. In this work, Joel Wallman and myself were the primary investigators of the research. We did most of the numerical simulations, developed the theoretical ideas and I wrote a large portion of the manuscript. Stefanie Beale contributed to the numerical analysis and all four authors contributed to the analysis of the results and the editing of the manuscript.

2.1 Introduction and motivation

In Sec. 1.1 we briefly discussed how the error syndrome can be used to remove potential errors afflicting a quantum system. In this chapter we will consider this problem in great detail.

Given an error syndrome, a recovery operation is performed in order to correct the error(s) most likely to have occurred. A decoding algorithm is an algorithm for determining a good recovery operation for an observed syndrome [22]. Note that decoding is elsewhere used to refer to the different process of transferring information from logical to physical qubits.

By the Gottesman-Knill theorem [23], Pauli channels can be efficiently simulated on a classical computer when the underlying quantum circuits contain only gates from the Clifford group, qubits prepared in computational basis states and measurements that are performed in the computational basis [24]. Simulating non-Pauli channels in fault-tolerant architectures is computationally demanding and has been done only for small codes [25, 26, 27, 4]. Assuming perfect error correction, that is, perfect preparations of encoded states and syndrome measurements, Rhan *et al.* introduced a technique to obtain the effective noise channel after performing error correction [28]. The technique, based on the process matrix formalism, is applicable to general completely positive and trace-preserving (CPTP) noise. Rhan *et al.* also showed how to efficiently compute threshold values for concatenated codes under a fixed decoder when each qubit is afflicted by CPTP noise. However, the recovery protocols were suboptimal, that is, they did not achieve the best error suppression.

Concatenated codes are formed by encoding each physical qubit of an error correcting code into another code, and the procedure can be repeated recursively. One could obtain the optimal recovery operator by measuring the error syndrome of the full concatenated code. For a code encoding one logical qubit into n physical qubits, the number of syndroms grows as $2^{cn^{l-1}}$ for l levels (c is a constant that depends on the code) making it computationally unfeasible to keep track of all of them [29].

In order to find optimal recovery operators without having to measure the syndromes of the full concatenated code, soft-decoding algorithms were implemented in Refs. [29, 30] under the perfect error correction assumptions. In Ref. [29], the entire list of probabilities for all possible recoveries conditioned on the observed syndrome were retained and passed on to the next level of concatenation in order to implement the optimal recovery operation. The method was applied to study thresholds for depolarizing noise. In a message passing simulation, the total number of syndromes that must be retained grows exponentially with increasing concatenation levels. Therefore, keeping track of all syndromes is inefficient.

In Ref. [31], again using the perfect error correction assumptions, Darmawan and Poulin developed a tensor-network algorithm to compute threshold values for the surface code under arbitrary local noise. The algorithm allowed for the simulation of higher-distance surface codes compared to work done in Ref. [4] and resulted in competitive threshold values for the studied noise models. However, the algorithm does not use non-Pauli transversal gates to its advantage.

In both the soft-decoding and tensor network approaches, the number of syndromes grows exponentially when increasing the code's distance. Therefore, rather than considering all syndrome values, syndromes are sampled from a distribution, leading to statistical

fluctuations in the reported thresholds. It is possible that certain unsampled syndromes could change the behavior of the effective noise at the next level in a significant way.

Hard decoding algorithms apply recovery operations independently at each concatenation level based on the measured syndrome (see Sec. 2.2.5). Syndrome information from previous levels are not used to update the recovery maps. This will generally result in a sub-optimal recovery protocol. However, hard decoding has the advantage of being linear in the number of concatenation levels even when considering *all* syndrome measurements, meaning that the required computational resources to compute the recovery operation remain linear even as the code distance increases exponentially with the number of concatenation levels.

This section focuses on developing a hard decoding algorithm capable of optimizing threshold values and lowering error rates of an error correcting code compared to traditional hard decoding schemes. If the code has non-Pauli single-qubit transversal gates, our algorithm can lead to even further improvements in the computed threshold values and error rates. We assume that error correction can be done perfectly so that additional errors are not introduced during the encoding and decoding protocols.

Our hard decoding algorithm is implemented using the process matrix formalism and can be applied to noise models described by general CPTP maps. The noise behaviour can change between different concatenation levels, even for depolarizing noise models. Therefore, for a particular syndrome measurement, the best choice of a recovery operator can differ from level to level.

We show that our hard decoding algorithm can still lead to reduced error rates even when applied to noise that differs slightly from the noise used to optimize the recovery maps. This indicates that our decoding scheme is robust to perturbative deviations from the assumed noise model.

For codes with transversal Clifford gates, we show that applying a Pauli twirl to a coherent noise channel results in lower threshold values and higher error rates than those obtained when applying our hard decoding algorithm to the original channel. However, if we only optimize over Pauli recovery maps, the Pauli twirl improves the threshold.

This chapter is structured as follows. In Sec. 2.2, we review some preliminary concepts such as the process matrix formalism and how it is used with stabilizer codes, logical noise resulting from independent and correlated physical noise, logical noise in concatenated codes and threshold hypersurfaces for general noise models. In Sec. 2.3 we describe our hard decoding algorithm for optimizing threshold values of Markovian noise models. In Sec. 2.4 we describe how to numerically calculate threshold hypersurfaces for both symmetric decoders and decoders obtained from our hard decoding algorithm.

We then present the results of numerical simulations of the 5-qubit code, Steane's 7-qubit code, Shor's 9-qubit code and the surface-17 code. For each code (excluding the surface-17 code), thresholds and infidelities using our hard decoding optimization protocol are computed for amplitude-phase damping noise (Sec. 2.5) and coherent noise (Sec. 2.6). The concept of infidelity is defined in Eq. 2.35. For the same noise models, we consider level-1 infidelities of the surface-17 code. We also consider thresholds and infidelities for the 7-qubit code where the noise model was described by two-qubit correlated dephasing noise (Sec. 2.7). In Sec. 2.8, we compute thresholds of the Steane code for a coherent error noise channel and its Pauli twirled counterpart (using both logical Clifford corrections and Pauli only corrections). Lastly, we study the robustness of our decoding algorithm to small unknown perturbations of a noise channel (Sec. 2.9).

The amplitude-phase damping threshold and infidelity plots can be found in Fig. 2.4. Applying our optimized hard decoding algorithm can more than double thresholds and reduce infidelities by more than two orders of magnitude.

For coherent error noise, threshold plots are given in Fig. 2.6 and infidelity plots are given in Fig. 2.7. For certain rotation axes, our optimized hard decoding algorithm results in errors that are correctable for all rotation angles. In some regimes, infidelities can be reduced by several orders of magnitude. For all the aforementioned noise models, the 5-qubit code consistently achieves higher thresholds and lower error rates compared to the 7 and 9-qubit codes. There is one exception where the Hadamard transform of the 9-qubit code outperforms the 5-qubit code for amplitude-phase damping noise in a small regime. For most studied noise models, the 7-qubit code achieves higher threshold values and lower error rates than the 9-qubit code, with the exception of rotations near the y-axis due to the asymmetries in the Shor codes stabilizer generators.

The threshold plot comparing a coherent error noise channel to its Pauli twirled counterpart is shown in Fig. 2.9. By performing Clifford corrections, the coherent noise channel outperforms its Pauli twirled counterpart for all sampled rotation axes. Lastly, plots showing the robustness of our decoding algorithm to small unknown perturbations are shown in Fig. 2.10. In certain regimes, applying our decoding algorithm results in lower logical failure rates even if the noise model is not perfectly known.

2.2 Stabilizer codes in the process matrix formalism

We begin by outlining the formalism we use to simulate the performance of concatenated codes under general CPTP noise. After introducing the process matrix formalism for CPTP

maps, we give an alternative description of stabilizer codes to the one discussed in Sec. 1.1 which will be convenient for what follows. Afterwords, we will derive expressions for the process matrix conditioned on observing a specific measurement syndrome for independent single-qubit noise and for two-qubit correlated noise. We then define thresholds for a noise model and define some fixed decoders which will be used in the remainder of this section.

For clarity, we always use Roman font for operators acting on $\mathbb{C}^d$ (e.g., a unitary U), calligraphic font for a channel acting on the operator space (i.e., a superoperator, e.g., $\mathcal{U}(\rho) = U\rho U^\dagger$) and bold calligraphic font for the matrix representation of a channel (e.g., $\boldsymbol{\mathcal{U}}$).

2.2.1 Process matrix formalism for noise at the physical level

A CPTP noise channel $\mathcal{N}$ acting on a state ρ can be written in terms of its Kraus operator decomposition

$$\mathcal{N}(\rho) = \sum_j A_j \rho A_j^\dagger, \tag{2.1}$$

where $\sum_j A_j^\dagger A_j = I$ for trace-preserving channels [32, 33]. Alternatively, Eq. 2.1 can be rewritten as a matrix product using the process matrix formalism. To do so, note that any matrix $M \in \mathbb{C}^{d\times d}$ can be expanded as

$$M = \sum_i M_i B_i, \tag{2.2}$$

where $\boldsymbol{B} = \{B_i\}$ is a trace-orthonormal basis for the space of density matrices, that is, $\mathrm{Tr}(B_i^\dagger B_j) = \delta_{i,j}$, and $M_i = \mathrm{Tr}(B_i^\dagger M)$. We exclusively study multi-qubit channels and so set the B_i to be the normalized Pauli matrices, $\boldsymbol{B} = \boldsymbol{\sigma} = (I, X, Y, Z)/\sqrt{2}$ for a single qubit, and $\boldsymbol{B} = \boldsymbol{\sigma}^{\otimes n}$ for n qubits.

We then define a map $|.\rangle\!\rangle : \mathbb{C}^{d\times d} \to \mathbb{C}^{d^2}$ by setting $|B_j\rangle\!\rangle = \mathbf{e}_j$, where $\{\mathbf{e}_j\}$ is the canonical unit basis of $\mathbb{C}^{d^2}$, and extend the map linearly so that

$$|M\rangle\!\rangle = \sum_j M_j |B_j\rangle\!\rangle = \begin{pmatrix} \mathrm{Tr}[B_1^\dagger M] \\ \vdots \\ \mathrm{Tr}[B_{d^2}^\dagger M] \end{pmatrix}. \tag{2.3}$$

Defining $\langle\!\langle M| = |M\rangle\!\rangle^\dagger$, we have $\langle\!\langle M|N\rangle\!\rangle = \mathrm{Tr}(M^\dagger N)$.

Because quantum channels are linear,

$$\begin{aligned}
|\mathcal{N}(\rho)\rangle\!\rangle &= \sum_j \rho_j |\mathcal{N}(B_j)\rangle\!\rangle \\
&= \sum_{i,j} \rho_i |\mathcal{N}(B_j)\rangle\!\rangle\langle\!\langle B_j|B_i\rangle\!\rangle \\
&= \Big(\sum_j |\mathcal{N}(B_j)\rangle\!\rangle\langle\!\langle B_j|\Big)\Big(\sum_i \rho_i |B_i\rangle\!\rangle\Big) \\
&= \boldsymbol{\mathcal{N}}|\rho\rangle\!\rangle
\end{aligned} \tag{2.4}$$

where $\rho_j = \langle\!\langle B_j|\rho\rangle\!\rangle$ are the expansion coefficients of ρ and we used $\langle\!\langle B_j|B_i\rangle\!\rangle = \mathrm{Tr}(B_j^\dagger B_i) = \delta_{j,i}$. We implicitly defined the matrix representation $\boldsymbol{\mathcal{N}}$ of the quantum channel $\mathcal{N}$.

2.2.2 Stabilizer codes

We now give more details of stabilizer codes which will provide the necessary links for decoding using the process matrix formalism. In this chapter, we only consider the stabilizer codes in Tab. 2.1, and so set $k = 1$.

We assume that states in $\mathcal{H}_2$ can be encoded in $\mathcal{H}_C$ by an encoding map $\mathcal{E}$ and decoded back to $\mathcal{H}_2$ by the adjoint map $\mathcal{E}^\dagger$. We consider the case where the encoding and decoding protocols can be done perfectly without introducing additional errors, so that $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is encoded to $|\overline{\psi}\rangle = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle \in \mathcal{H}_C$ and

$$\mathcal{E}(\rho_{in}) = B\rho_{in}B^\dagger \tag{2.5}$$

with $B = |\overline{0}\rangle\langle 0| + |\overline{1}\rangle\langle 1|$ and some abuse of notation. Since $(1/|\mathcal{S}|)\sum_k \mathcal{S}_k$ acts as the projector onto the codespace ($|\mathcal{S}|$ is the total number of elements in the stabilizer group) and representing $\overline{\tau}$ as the logical version of τ ($\tau \in \boldsymbol{\sigma}$), we define

$$E_\tau = \mathcal{E}(\tau) = \frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}} S\overline{\tau}, \tag{2.6}$$

so that E_τ implements τ in $\mathcal{H}_C$ and vanishes elsewhere.

We also assume that syndrome measurements and recovery maps $\mathcal{R}$ are perfect, so that the only errors are due to a noise process $\mathcal{N}$ acting on $\mathcal{H}$. More details about fault-tolerant encoding and measurements can be found in, for example, Refs. [6, 11, 34, 35, 36].

Suppose a physical Pauli error E occurs on a system in the encoded state $|\overline{\psi}\rangle \in \mathcal{H}_C$. Measuring the stabilizer generators yields the syndrome $l = l_1 l_2 \dots l_{n-k}$ where

$$l_i = \begin{cases} 0 & \text{if } [E, g_i] = 0 \\ 1 & \text{if } \{E, g_i\} = 0, \end{cases} \tag{2.7}$$

$[A, B] = AB - BA$ and $\{A, B\} = AB + BA$. Let Q_l be the set of physical Pauli errors that give the syndrome l, which are all of size $|Q_l| = 2^{2n}/2^{n-k} = 2^{n+k}$. When the syndrome l is measured, a recovery operator $R_l \in Q_l$ is chosen and applied to the state $E|\overline{\psi}\rangle$, returning it to the code space. If $R_l E \in \mathcal{S}$, then the correct state is recovered and the error is removed. Otherwise, $R_l E|\overline{\psi}\rangle$ will differ from $|\overline{\psi}\rangle$ by a logical Pauli operator [11]. The desired outcome of decoding is to find a set of recovery operators which result in the highest probability of recovering the original input state under a given noise model.

As an example, the stabilizer generators for the 3-qubit code protecting against bit-flip errors are $S_1 = Z_1 Z_2$ and $S_2 = Z_2 Z_3$. It can be verified that the errors X_1 and $X_2 X_3$ produce the syndrome $l = 10$ so that $Q_{10} = \{X_1, X_2 X_3\}$. Therefore, if the measured syndrome is $l = 10$, one can either choose X_1 or $X_2 X_3$ to implement the recovery. The particular choice can influence the fidelity of the encoded qubit. For instance, for uncorrelated noise models where single-weight errors are more likely, the better choice for the recovery operator would be $R_{10} = X_1$.

2.2.3 Effective process matrix at the logical level

The process of encoding, applying the physical noise $\mathcal{N}$ to the encoded state, implementing the appropriate recovery maps for the measured syndrome l and decoding yields the effective single-qubit channel

$$\mathcal{G}(\mathcal{N}, R_l) = \mathcal{E}^\dagger \circ \mathcal{R}_l \circ \mathcal{N} \circ \mathcal{E}, \tag{2.8}$$

where $\mathcal{R}_l$ includes the measurement update and the recovery map $R_l \in Q_l$. We now outline how this effective channel can be represented in the process matrix formalism, mostly following Ref. [28] with a straightforward generalization to consider individual syndromes. The states before encoding and after decoding, ρ_{in} and ρ_{out} respectively, are related by

$$|\rho_{\text{out}}\rangle\!\rangle = \boldsymbol{\mathcal{G}}(\mathcal{N}, R_l)|\rho_{\text{in}}\rangle\!\rangle, \tag{2.9}$$

where the process matrix representation of $\boldsymbol{\mathcal{G}}(R_l)$ is

$$\boldsymbol{\mathcal{G}}(\mathcal{N}, R_l) = \sum_{\sigma \in \boldsymbol{\sigma}} |\mathcal{G}(\mathcal{N}, R_l)(\sigma)\rangle\!\rangle\langle\!\langle \sigma| \tag{2.10}$$

5-qubit code	Steane code	Z-Shor code	Surface-17 code
XZZXI	IIIZZZZ	ZZIIIIIII	ZIIZIIIII
IXZZX	IZZIIZZ	ZIZIIIIII	IZZIZZIII
XIXZZ	ZIZIZIZ	IIIZZIIII	IIIZZIZZI
ZXIXZ	IIIXXXX	IIIZIZIII	IIIIIZIIZ
	IXXIIXX	IIIIIIZZI	XXIXXIIII
	XIXIXIX	IIIIIIZIZ	IXXIIIIII
		XXXXXXIII	IIIIXXIXX
		IIIXXXXXX	IIIIIIXXI
$\langle C_{\pi/3}, X, Z\rangle$	$\langle H, S\rangle$	$\langle X, Z\rangle$	$\langle X, Z\rangle$

Table 2.1: Stabilizer generators (top) and the group $\mathscr{L}$ of single-qubit transversal logical operations (bottom) for the 5-qubit code [1], Steane's 7-qubit code [2], Shor's 9-qubit code [3], and the surface-17 code [4], where H and S are the Hadamard and phase gates respectively, $C_{\pi/3} = \exp[i\pi(X+Y+Z)/3\sqrt{3}] \propto SH$, and $\langle . \rangle$ denotes the group generated by the argument. We also consider the X-Shor code, obtained from the Z-Shor code by mapping $X \leftrightarrow Z$. For each code, the logical operators are $X_L = X^{\otimes n}$ and $Z_L = Z^{\otimes n}$. We only consider the surface-17 code at the first level, as surface codes are not scaled up by concatenation. The surface-17 code is so named as it consists of 9 data qubits and 8 ancilla measurement qubits, and is equivalent to the other 2-D configuration with 9 data qubits in Ref. [4] under the assumption of perfect measurements.

by Eq. 2.4. The entries of the process matrix are

$$\begin{aligned} \boldsymbol{\mathcal{G}}_{\sigma\tau}(\mathcal{N}, R_l) &= \langle\!\langle \sigma | \mathcal{G}(\mathcal{N}, R_l)(\tau) \rangle\!\rangle \\ &= \langle\!\langle \mathcal{E}(\sigma) | \mathcal{R}_l \circ \mathcal{N}(E_\tau) \rangle\!\rangle \\ &= \langle\!\langle E_\sigma | \mathcal{R}_l \circ \mathcal{N}(E_\tau) \rangle\!\rangle. \end{aligned} \tag{2.11}$$

By the Born rule, the probability of the syndrome l occurring is $p(l) = \mathrm{Tr}(\mathscr{P}_l \mathcal{N}(\rho_{\mathrm{in}}))$ where the projection operator for the syndrome l is

$$\mathscr{P}_l = \prod_{j=1}^{n-k} \frac{1}{2}(I + (-1)^l g_j). \tag{2.12}$$

Note that from Eq. 2.6, $E_\tau = \mathscr{P}_0 \overline{\tau}$. With the corresponding recovery operator R_l, the transformation on the process matrix can be obtained by implementing the von Neumann-Lüders update rule [37] resulting in

$$\begin{aligned}\boldsymbol{\mathcal{G}}_{\sigma\tau}(\mathcal{N}, R_l) &= \frac{1}{p(l)} \langle\!\langle E_\sigma | R_l \mathscr{P}_l \mathcal{N}(E_\tau) \mathscr{P}_l^\dagger R_l^\dagger \rangle\!\rangle \\ &= \frac{1}{p(l)} \langle\!\langle \mathscr{P}_l^\dagger R_l^\dagger E_\sigma R_l \mathscr{P}_l | \mathcal{N}(E_\tau) \rangle\!\rangle \end{aligned} \tag{2.13}$$

Following Ref. [28], Eq. 2.13 can be further simplified by noting that $R_l^\dagger E_\sigma R_l$ is a map from the space projected by $\mathscr{P}_l$ to itself and vanishes elsewhere so that $\mathscr{P}_l^\dagger R_l^\dagger E_\sigma R_l \mathscr{P}_l = R_l^\dagger E_\sigma R_l$. Defining

$$D_\sigma^{(l)} \equiv R_l^\dagger E_\sigma R_l, \tag{2.14}$$

we arrive at

$$\boldsymbol{\mathcal{G}}_{\sigma\tau}(\mathcal{N}, R_l) = \frac{1}{p(l)} \langle\!\langle D_\sigma^{(l)} | \mathcal{N}(E_\tau) \rangle\!\rangle. \tag{2.15}$$

In the remainder of this section we will assume that the noise is uncorrelated, so that it takes the form $\mathcal{N} = \mathcal{N}^{(1)} \otimes \ldots \otimes \mathcal{N}^{(n)}$ where $\mathcal{N}^{(i)}$ is the process matrix for the physical noise acting on qubit i. As in Eq. 2.2, we can expand E_τ and $D_\sigma^{(l)}$ as

$$E_\tau = \sum_{\mu_i \in \tilde{\boldsymbol{B}}} \alpha^\tau_{\{\mu_i\}} \mu_1 \otimes \ldots \otimes \mu_n, \tag{2.16}$$

$$D_\sigma^{(l)} = \sum_{\nu_i \in \tilde{\boldsymbol{B}}} \beta^\sigma_{\{\nu_i\}}(R_l) \nu_1 \otimes \ldots \otimes \nu_n, \tag{2.17}$$

where $\tilde{\boldsymbol{B}} \subset \boldsymbol{B}$ only has support over products of the stabilizer group and logical operators from Eq. 2.6. For an operator of the form $U = \pm\mu_1 \otimes \ldots \otimes \mu_n$ and using the notation of Ref. [28], we define the function $\phi(U) = \mu_1 \otimes \ldots \otimes \mu_n$ and $a(U) \in \{0, 1\}$ such that $U = (-1)^{a(U)}\phi(U)$. Substituting Eq. 2.6 into Eq. 2.16 gives

$$\alpha^\tau_{\phi(S\overline{\tau})} = \frac{1}{2^{\frac{n}{2}-1}} (-1)^{a(S\overline{\tau})}. \tag{2.18}$$

The α coefficient takes into account the overall sign of the product between elements in the stabilizer group and logical operators (for example, the code with XX and ZZ

stabilizers also has $(XX)(ZZ) = -YY$ as a stabilizer). The factor of $\frac{1}{2^{\frac{n}{2}-1}}$ comes from choosing a trace-orthonormal basis.

The β coefficient can be obtained by substituting Eq. 2.6 into Eq. 2.14, commuting R_l to the left, using $R_l^\dagger R_l = I$ and setting the result equal to Eq. 2.17. Defining $\eta(A, B) = \pm 1$ for $AB = \pm BA$, we obtain

$$\beta^{\sigma}_{\phi(S_k\overline{\sigma})}(R_l) = \alpha^{\sigma}_{\phi(S_k\overline{\sigma})}\eta(R_l, S_k)\eta(R_l, \overline{\sigma}). \tag{2.19}$$

Therefore, for a particular error syndrome l, picking different recovery operators from the set $Q_l^\dagger$ will, in general, yield different values for the coefficient β. This will, in turn, result in different effective noise dynamics. Closed form expressions for α and β are given in Appendix. A.1.

Substituting Eq. 2.17 into Eq. 2.15, we obtain

$$\boldsymbol{\mathcal{G}}_{\sigma\tau}(\mathcal{N}, R_l) = \frac{1}{p(l)} \sum_{\{\mu_i\},\{\nu_i\}} \beta^{\sigma}_{\{\nu_i\}}(R_l)\alpha^{\tau}_{\{\mu_i\}} \prod_{i=1}^{n} \boldsymbol{\mathcal{N}}^{(i)}_{\nu_i\mu_i}, \tag{2.20}$$

where the sum is over all elements in the stabilizer group and $\boldsymbol{\mathcal{N}}^{(i)}_{\nu_i\mu_i} = \mathrm{Tr}[\nu_i \mathcal{N}^{(i)}(\mu_i)]$.

The effective noise channel can be obtained by averaging Eq. 2.20 over all syndrome measurements. Summing over all syndromes, we define $\beta^{\sigma}_{\{\nu_i\}} \equiv \sum_l \beta^{\sigma}_{\{\nu_i\}}(R_l)$, we have

$$\begin{aligned} \boldsymbol{\mathcal{G}}_{\sigma\tau}(\mathcal{N}) &= \sum_l p(l)\boldsymbol{\mathcal{G}}_{\sigma\tau}(\mathcal{N}, R_l) \\ &= \sum_{\{\mu_i\},\{\nu_i\}} \beta^{\sigma}_{\{\nu_i\}}\alpha^{\tau}_{\{\mu_i\}} \prod_{i=1}^{n} \boldsymbol{\mathcal{N}}^{(i)}_{\nu_i\mu_i}, \end{aligned} \tag{2.21}$$

and we will refer to $\mathcal{G}$ as the effective process matrix for the noise channel $\mathcal{N}$. Note that the normalization factor $1/p(l)$ that appears when implementing the von Neumann-Lüders update rule gets cancelled when averaging over all syndrome measurements. For simplicity, and in the remaining sections, when referring to process matrices for individual syndrome measurements as in Eq. 2.20, we will omit the normalization factor.

When considering concatenated codes, it will prove useful to define the coding map Ω^C for a code C as

$$\Omega^C : \mathcal{N} \to \mathcal{G}(\mathcal{N}) = \mathcal{E}^\dagger \circ \mathcal{R} \circ \mathcal{N} \circ \mathcal{E}, \tag{2.22}$$

where the matrix representation of $\mathcal{G}$ is obtained from Eq. 2.21. Note that in Eq. 2.22, $\mathcal{R}$ includes the measurement update and recovery map averaged over all syndrome measurements. The coding map relates the effective noise dynamics at the logical level resulting from the error correction protocol to the noise dynamics occurring at the physical level.

2.2.4 Process matrix for two-qubit correlated noise

In this section we also consider noise models where nearest-neighbor two-qubit correlations occur. More specifically, we will consider a noise channel of the form

$$\begin{aligned}\mathcal{N}(\rho^{\otimes n}) &= \mathcal{N}^{(1)}(\rho)^{\otimes n} + \\ &p_2(\sum_{j=1}^{n-1} \mathcal{N}^{(2)}_{j,j+1}(\rho^{\otimes n}) + \mathcal{N}^{(2)}_{1,n}(\rho^{\otimes n})),\end{aligned} \tag{2.23}$$

where $\mathcal{N}^{(1)}$ corresponds to local uncorrelated noise and with probability p_2, $\mathcal{N}^{(2)}_{j,j+1}(\rho^{\otimes n}) = Z_j Z_{j+1} \rho^{\otimes n} Z_{j+1} Z_j$ applies phase-flip operators to qubits j and $j+1$. For noise models of this form, the process matrix describing the effective noise is given by

$$\begin{aligned}\boldsymbol{\mathcal{G}}_{\sigma\tau}(\mathcal{N}) &= \boldsymbol{\mathcal{G}}_{\sigma\tau}([\mathcal{N}^{(1)}]^{\otimes n}) + \\ &p_2 \sum_{\{\mu_i\},\{\nu_i\}} \beta^\sigma_{\{\nu_i\}} \alpha^\tau_{\{\mu_i\}} (\sum_{j=1}^{n-1} \prod_{\substack{i=1 \\ i \notin \{j,j+1\}}}^{n} \boldsymbol{\mathcal{Z}}^{(2)}_{\nu_j \mu_j} \boldsymbol{\mathcal{Z}}^{(2)}_{\nu_{j+1}\mu_{j+1}} \mathcal{I}_{\nu_i \mu_i} + \\ &\prod_{i=2}^{n-1} \boldsymbol{\mathcal{Z}}^{(2)}_{\nu_1 \mu_1} \boldsymbol{\mathcal{Z}}^{(2)}_{\nu_n \mu_n} \mathcal{I}_{\nu_i \mu_i}),\end{aligned} \tag{2.24}$$

where $\mathcal{Z}(\rho) = Z\rho Z$ in keeping with our standard notation for channels. The contribution from correlated noise appears in the second term of Eq. 2.24.

2.2.5 Effective noise channels for concatenated codes

Concatenation is the process of encoding each of the n physical qubits encoded in an inner code C_1 into an outer code C_2. One can go to arbitrary levels of concatenation by recursively applying this procedure.

More formally, we consider an m-qubit code C^{out} with encoding map $\mathcal{E}^{out}$ which will form the outer code, and an n-qubit code C^{in} with encoding map $\mathcal{E}^{in}$ which will form the inner code. The logical qubit ρ_0 is first encoded using C^{out}, and afterwards each of the m qubits are encoded using the code C^{in}. The composite encoding map is given by

$$\tilde{\mathcal{E}} = (\mathcal{E}^{in})^{\otimes m} \circ \mathcal{E}^{out}. \tag{2.25}$$

Throughout this section we will implement a hard decoding scheme, which applies a recovery operation independently at each concatenation level [11]. Each code block is thereby corrected based on the inner code. The entire register is then corrected based on the outer code. We denote the mn-qubit code with the effective encoding map $\tilde{\mathcal{E}}$ by $C^{out}(C^{in})$. The procedure for choosing a decoding map for a given noise model described by a CPTP map will be addressed in Sec. 2.3.

Let $\mathcal{G}$ describe the effective dynamics of C^{in} where the physical noise dynamics are described by $\mathcal{N}$. To obtain the effective noise dynamics of $\tilde{\mathcal{G}}$ for the code $C^{out}(C^{in})$, we assume that all n-qubit blocks evolve according to $\mathcal{N}$ so that the mn-qubit code evolves according to (assuming uncorrelated noise)

$$\tilde{\mathcal{N}} = \mathcal{N}^{\otimes m}. \tag{2.26}$$

For convenience, we define $\mathcal{E}^{\dagger}_{\mathcal{R}} \equiv \mathcal{E}^{\dagger} \circ \mathcal{R}$ so that $\mathcal{E}^{\dagger}_{\mathcal{R}}$ includes both the recovery and decoding step. In Ref. [28], it was shown that with the above assumptions $\tilde{\mathcal{G}}$ is given by

$$\tilde{\mathcal{G}} = (\mathcal{E}^{\dagger})^{out}_{\mathcal{R}} \circ \mathcal{G}^{\otimes m} \circ \mathcal{E}^{out}. \tag{2.27}$$

From Eq. 2.22, the above equation can be written as

$$\tilde{\mathcal{G}} = \Omega^{C^{out}}(\mathcal{G}) = \Omega^{C^{out}}(\Omega^{C^{in}}(\mathcal{N})). \tag{2.28}$$

For uncorrelated noise, we conclude that the effective channel for the code $C^{out}(C^{in})$ can be computed in the same way that lead to Eq. 2.21 by replacing $\mathcal{N}$ with $\mathcal{G}$ for the code C^{out}. The concatenated code $C^{out}(C^{in})$ can then be described by the composition of maps

$$\Omega^{C^{out}(C^{in})} = \Omega^{C^{out}} \circ \Omega^{C^{in}}. \tag{2.29}$$

The above equation can be easily generalized to the concatenation of codes in the set $\{C_1, C_2, \ldots, C_n\}$ yielding the map

$$\Omega^{C_1(C_2(\ldots C_n))} = \Omega^{C_1} \circ \Omega^{C_2} \circ \ldots \circ \Omega^{C_n}. \tag{2.30}$$

For the particular case where the same code $C_i = C$ ($i \in \{1, 2, \ldots, t\}$) is used at t levels of concatenation, we define

$$\mathcal{G}^{(t)}(\mathcal{N}) = \Omega^{C_1(C_2(\ldots C_t))}(\mathcal{N}). \tag{2.31}$$

For correlated noise, we cannot in general write the map for the code $C^{out}(C^{in})$ as a composition of maps for the code C^{out} and C^{in}. However, in this section we will assume that when the code is concatenated, no correlations occur between different code blocks. Only qubits within each code block undergo correlated noise described by Eq. 2.23. The noise dynamics for each code block of the code C^{out} will thus be described by the effective noise dynamics of Eq. 2.24 and the analysis leading to Eq. 2.30 will also apply in this case. This situation could be realized if the physical qubits in each lowest-level code are contained in individual nodes of a distributed quantum computer and is a good approximation if correlations decay exponentially with the separation between physical qubits.

2.2.6 Thresholds for noise models

A fixed noise process $\mathcal{N}$ is correctable by a concatenated code C if successive levels of concatenation eventually remove the error completely for arbitrary input states, that is, if

$$\lim_{t\to\infty} \boldsymbol{\mathcal{G}}^{(t)}(\mathcal{N}) = I_4, \tag{2.32}$$

where $\boldsymbol{\mathcal{G}}^{(t)}(\mathcal{N})$ is as defined in Eq. 2.31 and I_4 is the 4×4 identity matrix.[1]

A threshold for a code is defined relative to an m-parameter noise model, that is, a family $\mathscr{N} = \{\mathcal{N}_p : p \in [0,1]^m\}$ of noise processes such that $\mathcal{N}_0 = \mathcal{I}$. The $\mathscr{N}$-threshold for a code C is the hypersurface of the largest volume in $[0,1]^m$ containing only correctable noise processes and the origin, with the faces of $[0,1]^m$ removed.

The typical behavior of the diagonal components of the process matrix for a 1-parameter noise model is illustrated in Fig. 2.1. The diagonal components converge to one (zero) below (above) threshold, while the off-diagonal components converge to zero.

2.2.7 Specific decoders

The effective noise acting on a logical qubit is highly dependent upon both the physical noise processes and the choice of recovery operators for each syndrome (cf. Eq. 2.21).

[1] Formally, we could also require the error rate to decrease doubly-exponentially when quantified by an appropriate metric [33], however, we do not verify this requirement.

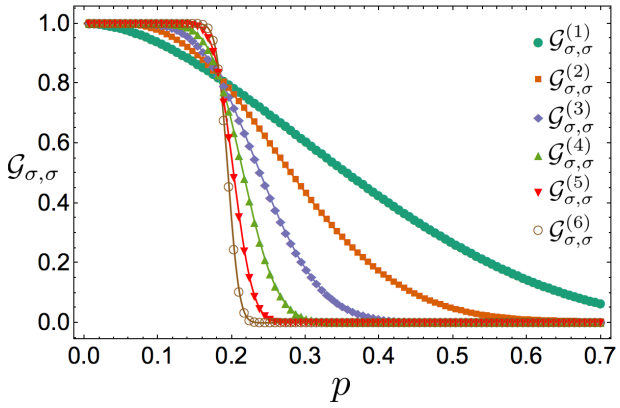


Figure 2.1: Representative plot of the smallest diagonal component $\boldsymbol{\mathcal{G}}_{\sigma,\sigma}$ of the process matrix for a noise model parametrized by p. As functions of p, the diagonal components of the process matrix approach a step-function as the number of concatenation levels approaches infinity. The threshold is the smallest value p_{th} such that $\lim_{t\to\infty} \boldsymbol{\mathcal{G}}^{(t)}(\mathcal{N}_p) = I_4$ for all $p \leq p_{th}$.

One decoder that will be very useful is the symmetric decoder; this decoder associates the measured syndrome with the error that acts on the fewest number of qubits and is consistent with the syndrome. If multiple errors acting on equal numbers of qubits are consistent with a syndrome, one is chosen arbitrarily and used each time that syndrome occurs. However the particular choice could affect the threshold value.

The symmetric decoder for the $[\![5, 1, 3]\!]$ code, for example, associates each syndrome to a unique weight-one Pauli operator. Therefore, all weight-one Pauli operators are corrected. However, one could choose a different decoder for the 5-qubit code. If we consider a noise model where only X-errors occur, a decoder could be chosen which corrects all weight-one and weight-two Pauli X errors. However, this decoder would not be able to correct any Y or Z type Pauli errors. More details will be provided in Sec. 2.5 and Sec. 2.6.

2.3 Hard decoding algorithm for optimizing error-correcting codes

We now present our optimized hard decoding algorithm that determines the choice of recovery operators at each level of concatenation. The goal of the algorithm is to correct the effective noise, that is, to map it to the identity channel $\mathcal{I}$ as quickly as possible. More

formally, let ε be a pre-metric on the space of CPTP maps, that is, a function such that $\varepsilon(\mathcal{N},\mathcal{M}) \geq 0$ with equality if and only if $\mathcal{N} = \mathcal{M}$ and $\varepsilon(\mathcal{N},\mathcal{M}) = \varepsilon(\mathcal{M},\mathcal{N})$ for all CPTP maps $\mathcal{N}$ and $\mathcal{M}$ (this is a pre-metric as ε does not have to satisfy the triangle inequality). The function $\varepsilon(\mathcal{N}) := \varepsilon(\mathcal{N},\mathcal{I})$ defines an 'error rate'.

We will set $1-\varepsilon(\mathcal{N})$ to be the average gate fidelity to the identity. This choice significantly reduces the amount of computational resources required to find the optimal recovery maps. The choice of ε may affect the performance of the decoder, however, we defer an investigation of this to future work.

Our hard decoding optimization algorithm selects recovery operations with the goal of minimizing the logical error rate after the recovery operations have been applied. The flowchart given in Fig. 2.2 applies the hard decoding algorithm to a channel $\mathcal{N}$ and determines whether the effective noise will converge to the identity with concatenation. In Fig. 2.2, $\mathcal{M}$ is a general CPTP map, $\mathcal{R}_{sym}$ is a (not necessarily unique) set of recovery operators for symmetric decoding (see Sec. 2.2.7), $\boldsymbol{\mathcal{G}}(\mathcal{M},k)$ are the distinct elements of $\{\boldsymbol{\mathcal{G}}(\mathcal{M},R) : R \in \mathcal{R}_{sym}\}$, $m(k)$ is the number of instances of $\boldsymbol{\mathcal{G}}(\mathcal{M},k)$, and L_g is a set of transversal logical operators. The optimized physical recovery maps are

$$R \to T(L_g^\dagger)R \tag{2.33}$$

for all $R \in \mathcal{R}_{sym}$, where $g = m(k)\mathcal{G}(\mathcal{M},R)$ and $T(L_g^\dagger)$ denotes the transversal implementation of $L_g^\dagger$. As we discuss in Sec. 2.4, the choice for $L_g^\dagger$ may not be unique. The action of Eq. 2.33 is equivalent to finding the set $\boldsymbol{\mathcal{L}}_g = \{L_g \,:\, g \in \mathscr{G}(\mathcal{M})\}$ of transversal logical operators that minimize

$$\varepsilon\Big(\sum_{g\in\mathscr{G}(\mathcal{M})} \boldsymbol{\mathcal{L}}_g g\Big) \tag{2.34}$$

There are 2^{n-1} syndromes, however, step 2 produces only 4, 7, 12, and 67 distinct process matrices for the $[\![5,1,3]\!]$, Steane, Shor, and surface-17 codes respectively, *independently* of the physical noise model. Therefore considering only the distinct $\boldsymbol{\mathcal{G}}(\mathcal{M},k)$ in step 2 reduces the memory and computational requirements by a factor between 4 and 20 for the codes considered in this section. We could also improve performance by setting the off-diagonal terms to zero when they are sufficiently small and recalculating $\mathscr{G}(\mathcal{M})$ for Pauli channels $\mathcal{M}$. Removing the off-diagonal terms corresponds to performing a Pauli twirl by applying a uniformly-random Pauli operator P to each physical qubit before the noise acts and then applying a logical P at the t^{th} concatenation level. However, this step

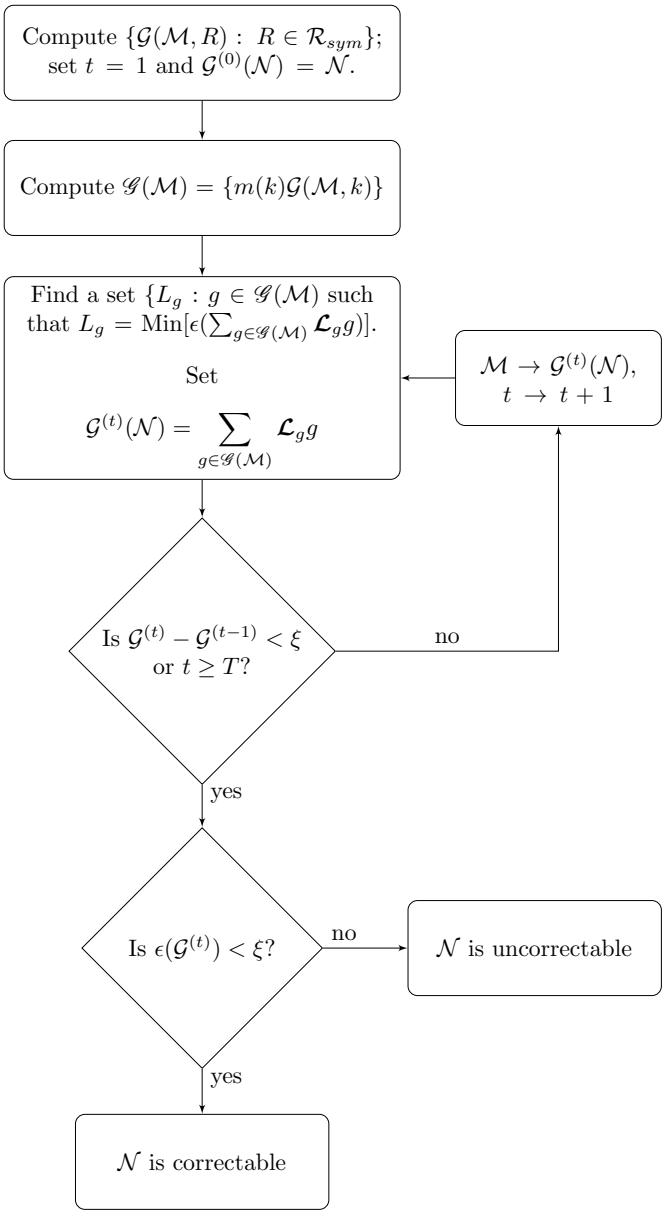


Figure 2.2: Method for selection of recovery operations for a fixed code C and noise channel $\mathcal{N}$. The iterative step that calculates the process matrix at level t, $\mathcal{G}^{(t)}(\mathcal{N})$, is equivalent to setting the recovery maps to $R_{opt}=T(L_g^\dagger)R$ for all $R\in\mathcal{R}_{sym}$, where $m(k)\boldsymbol{\mathcal{G}}(\mathcal{M},R)=g$ and $T(L_g^\dagger)$ denotes the transversal implementation of $L_g^\dagger$. This set may not be unique; see Sec. 2.3.2 for specific examples of when this can occur. We use the l_∞ norm, that is, the maximum of the absolute values of the entries of a matrix, to test whether the process matrix has converged. However, any matrix norm can be used instead.

complicates the algorithm and was not necessary to obtain the results of this section (it typically sped up computations by a factor between 2 and 10).

As we will discuss in Sec. 2.8, the use of a code's non-Pauli transversal gates (note that for any stabilizer code the logical Pauli operators are always transversal) can significantly increase the code's performance. This improvement is obtained when a syndrome measurement results in a logical non-Pauli error with high probability, which can occur even when significantly below threshold. This suggests that using highly symmetric stabilizer codes may provide better performance even at low error rates (in addition to also making non-trivial fault-tolerant computations more viable).

The resources required to find the set of recovery maps which optimally correct a noise model $\mathcal{N}$ are efficient in the number of concatenation levels required because our algorithm is independent of the observed syndromes from previous concatenation levels. The largest contribution to the complexity of our scheme comes from computing the β matrix for each syndrome measurement. From Eq. 2.19, there are $3 \times 2^{n-1}$ operations required to compute a β matrix for a particular syndrome value. The factor of 3 comes from computing the commutation relations (encoded by η) between R_l and the code's logical Pauli operators (R_l always commutes with the identity) and the factor of 2^{n-1} comes from verifying the commutation relations between R_l and all elements in the stabilizer group. As there are 2^{n-1} possible syndrome values, $3 \times 2^{2(n-1)}$ operations are required to compute all the β matrices.

2.3.1 Infidelity-optimized decoding

The average gate infidelity to the identity (hereafter simply the infidelity),

$$r(\mathcal{N}) = 1 - \int \mathrm{d}\psi \langle\psi|\mathcal{N}(|\psi\rangle\langle\psi|)|\psi\rangle, \tag{2.35}$$

is a commonly-used error pre-metric on the space of CPTP maps where the integral is over all pure states according to the unitarily-invariant Fubini-Study metric. The infidelity can be written as

$$r(\mathcal{N}) = \frac{4 - \mathrm{Tr}\,\boldsymbol{\mathcal{N}}}{6} \tag{2.36}$$

in the process matrix formalism for a single qubit [38]. The infidelity is particularly convenient for our algorithm because the trace is a linear function of the channel. Consequently,

to find a set $\{L_g : g \in \mathscr{G}(\mathcal{M})\}$ that minimizes $\varepsilon\Big(\sum_{g\in\mathscr{G}(\mathcal{M})} \boldsymbol{\mathcal{L}_g} g\Big)$ it is sufficient to maximize

$$\operatorname{Tr} \boldsymbol{\mathcal{L}} g \tag{2.37}$$

independently for each $g \in \mathscr{G}(\mathcal{M})$, rather than considering all $|\mathscr{G}(\mathcal{M})|^{|\mathscr{L}|}$ possibilities.

2.3.2 Resolving ties

There is one important caveat in the implementation of our hard decoding algorithm, namely, there may be multiple sets $\{L_g : g \in \mathscr{G}(\mathcal{M})\}$ that minimize the error in $\varepsilon\Big(\sum_{g\in\mathscr{G}(\mathcal{M})} \boldsymbol{\mathcal{L}_g} g\Big)$.

For example, consider the Steane code with $\mathcal{U}_\theta(\rho) = U_\theta \rho U_\theta^\dagger$ and $U_\theta = \cos\theta I_2 + i \sin\theta X$. Then the only two matrices from step 2 of our algorithm for any value of $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ are

$$\begin{aligned} \boldsymbol{\mathcal{R}}_{z1}(\theta) &= \frac{7\cos(8\theta) + 25}{32} \boldsymbol{\mathcal{U}_{\phi/2}} \\ \boldsymbol{\mathcal{R}}_{z2}(\theta) &= \frac{7\sin^2(4\theta)}{16} \boldsymbol{\mathcal{U}_{-3\theta}} \end{aligned} \tag{2.38}$$

for the trivial syndrome and the syndromes that detect X errors respectively, where

$$\begin{aligned} \phi &= \arctan\Big(\frac{(3\cos(4\theta) + \cos(8\theta) + 10)\tan^3(2\theta)}{-3\cos(4\theta) + \cos(8\theta) + 10}\Big) \\ &= 14(\theta^3 + \theta^5) + O(\theta^7). \end{aligned} \tag{2.39}$$

Similar expressions hold for other values of θ with different signs.

For this example, using all transversal gates significantly improves the recovery, as, for example, $U_{\pm\pi/4}$ (the phase gate around the X-axis) is a transversal gate and so $\boldsymbol{\mathcal{R}}_{z2}(\pi/12)$ can be perfectly recovered. Furthermore, there are two logical gates, namely U_0 and $U_{\pi/4}$, that maximize Eq. 2.37 for $g = \boldsymbol{\mathcal{R}}_{z2}(\pi/24)$, and so the choice is ambiguous. When confronted with such ambiguities, we choose the first logical operator that maximizes Eq. 2.37 (in particular, the identity if it is one of the options). As we will discuss further in Sec. 2.5, this ambiguity due to the ordering of logical operators does arise in practical examples without "fine-tuning" any parameters and it can impact performance.

2.4 Numerically calculating threshold hypersurfaces

We now describe our numerical method for calculating threshold hypersurfaces under symmetric (Sec. 2.2.7) and infidelity-optimized decoders (Sec. 2.3). For convenience, we regard a noise channel $\mathcal{N}$ as correctable if there exists some level of concatenation t such that $\mathrm{Tr}\,\boldsymbol{\mathcal{G}}^{(t)} \geq 4-\xi$, or, equivalently, the infidelity of $\mathcal{G}^{(t)}$ is at most $\xi/6$. The value of ξ was set to 0.01.

2.4.1 Symmetric threshold hypersurfaces

The subset of correctable errors for a given noise model is not generically connected. For this reason, a binary search between $p = 0$ and $p = 1$, where p is the noise parameter, is insufficient when calculating a threshold value because this method may miss some uncorrectable regimes. To calculate $p_{sym,thr}(q)$, a threshold value of p (with q fixed) when a symmetric decoder is applied at each level of concatenation, we initialized $p = 0$ and incremented by 0.05 until the noise with $p = p_u$ was uncorrectable, then implemented a binary search between $p = p_u - 0.05$ and $p = p_u$. To find a threshold hypersurface for a noise model with multiple parameters, we iteratively apply this procedure while varying q over a dense mesh.

2.4.2 Threshold hypersurfaces for our infidelity-optimized decoder

To calculate threshold values of a code C afflicted by a general CPTP map using our hard decoding algorithm, we follow the procedure illustrated in Fig. 2.3. Here $t(p)$ is the minimum number of concatenation levels required to correct $\mathcal{N}_{pq}$.

2.5 Thresholds and infidelities for amplitude-phase damping

In the remainder of this section, we show that our hard decoding algorithm leads to significant improvements in threshold values and, in some cases, decreases the infidelity by several orders of magnitude relative to the symmetric decoder. We will also show that

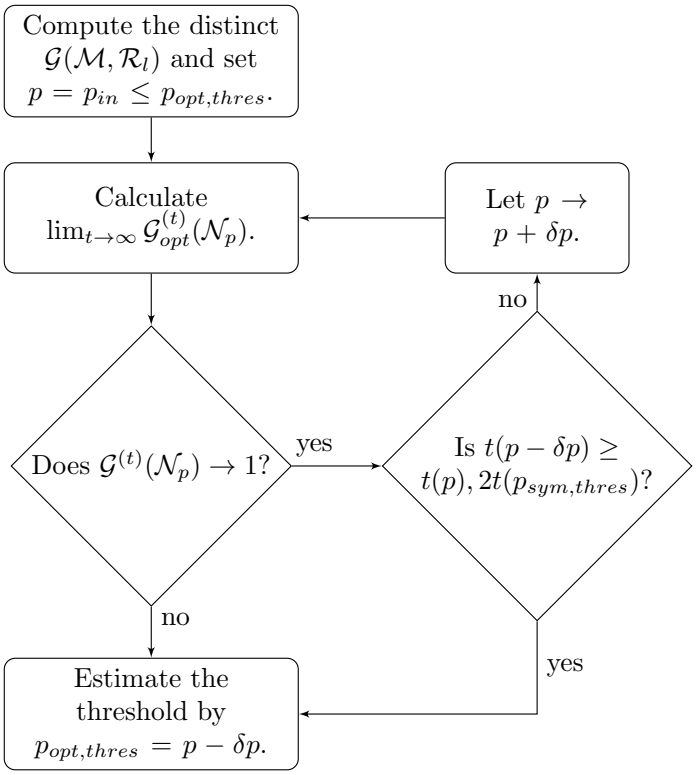


Figure 2.3: Method for lower-bounding the threshold for a one-parameter noise model $\mathcal{N}_p$, where $t(p)$ is the minimum number of concatenation levels required to correct $\mathcal{N}_p$. We begin by setting $p_{in} = p_{sym,thres}$ and $\delta p = 0.01$, then repeating with the new lower bound and $\delta p = 0.001$ and finally $\delta p = 0.0001$. To find the threshold hypersurface for an m-parameter noise model, repeat this procedure for $\mathcal{N}_{pq}$ while iterating through a mesh of points, q.

the performance of our decoder is robust to perturbations in the noise, so that it can be implemented using the necessarily imperfect knowledge of the noise in an experiment.

In this section we consider a physical noise model consisting of both amplitude and phase damping processes. The amplitude damping channel acts on a two-level system at zero temperature. If the system is in the excited state, then a transition to the ground state occurs with probability p. If the system starts in the ground state, it will remain in the ground state indefinitely. A physical example of this scenario would be the spontaneous emission of a photon in a two-level atom. The Kraus operators for the amplitude damping channel are [33]

$$A_{AD}^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix}, \; A_{AD}^{(1)} = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix}. \tag{2.40}$$

We point out that Eq. 2.40 can be generalized to take into account non-zero temperature effects. In this case, when the system is in the ground state, there is a non-zero probability of making a transition to the excited state. In Ref. [39], the performance of the 5-qubit code, Steane code and non-additive quantum codes was estimated for the generalized amplitude damping channel. However, the methods used did not allow for an exact analysis. In the remainder of this section we will only consider the amplitude-damping channel at zero temperature.

Phase damping arises when a phase kick $\exp(i\theta Z)$ is applied to a qubit with a random angle θ. When θ is sampled from a Gaussian distribution, then the Kraus operators are

$$A_{PD}^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\lambda} \end{pmatrix}, \; A_{PD}^{(1)} = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{\lambda} \end{pmatrix}, \tag{2.41}$$

where λ characterizes the width of the distribution of θ. The phase damping channel is also equivalent to the phase-flip channel, that is, applying a Z with probability $\alpha = (1+\sqrt{1-\lambda})/2$.

Combining the amplitude and phase damping channel, we consider the amplitude-phase damping channel given by

$$\mathcal{N}_{APD}(\rho) = \mathcal{N}_{PD}(\mathcal{N}_{AD}(\rho)) = \mathcal{N}_{AD}(\mathcal{N}_{PD}(\rho)). \tag{2.42}$$

As the amplitude-phase damping channel contains two parameters (p and λ), the threshold hypersurface will be a curve below which the process matrix is correctable.

The threshold curves and infidelity at the first and third concatenation levels for the $[\![5,1,3]\!]$, Steane, and Shor codes are illustrated in Fig. 2.4 and Fig. 2.5. The infidelities

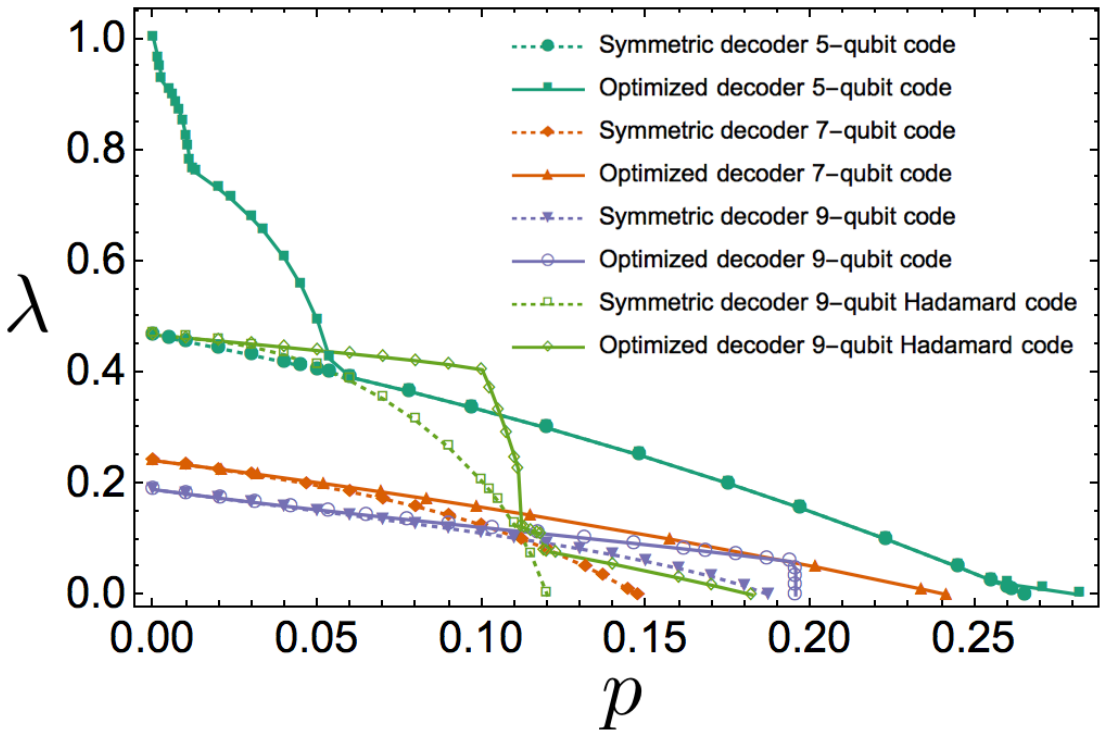


Figure 2.4: Threshold curves for the amplitude-phase damping channel under the $[\![5, 1, 3]\!]$, Steane code and the X and Z Shor code. The symmetrized and optimized curves overlap when the optimized decoder is the symmetric decoder, however, there are many regimes where the optimized decoder improves the threshold. For the $[\![5, 1, 3]\!]$ code, thresholds using the optimized decoder increase by as much as a factor of 2.14 relative to the symmetric decoder. The curves for the fixed symmetric decoders are all smooth, whereas the curves for the optimized decoders have kinks corresponding to points where the decoder changes to exploit asymmetries in the noise.

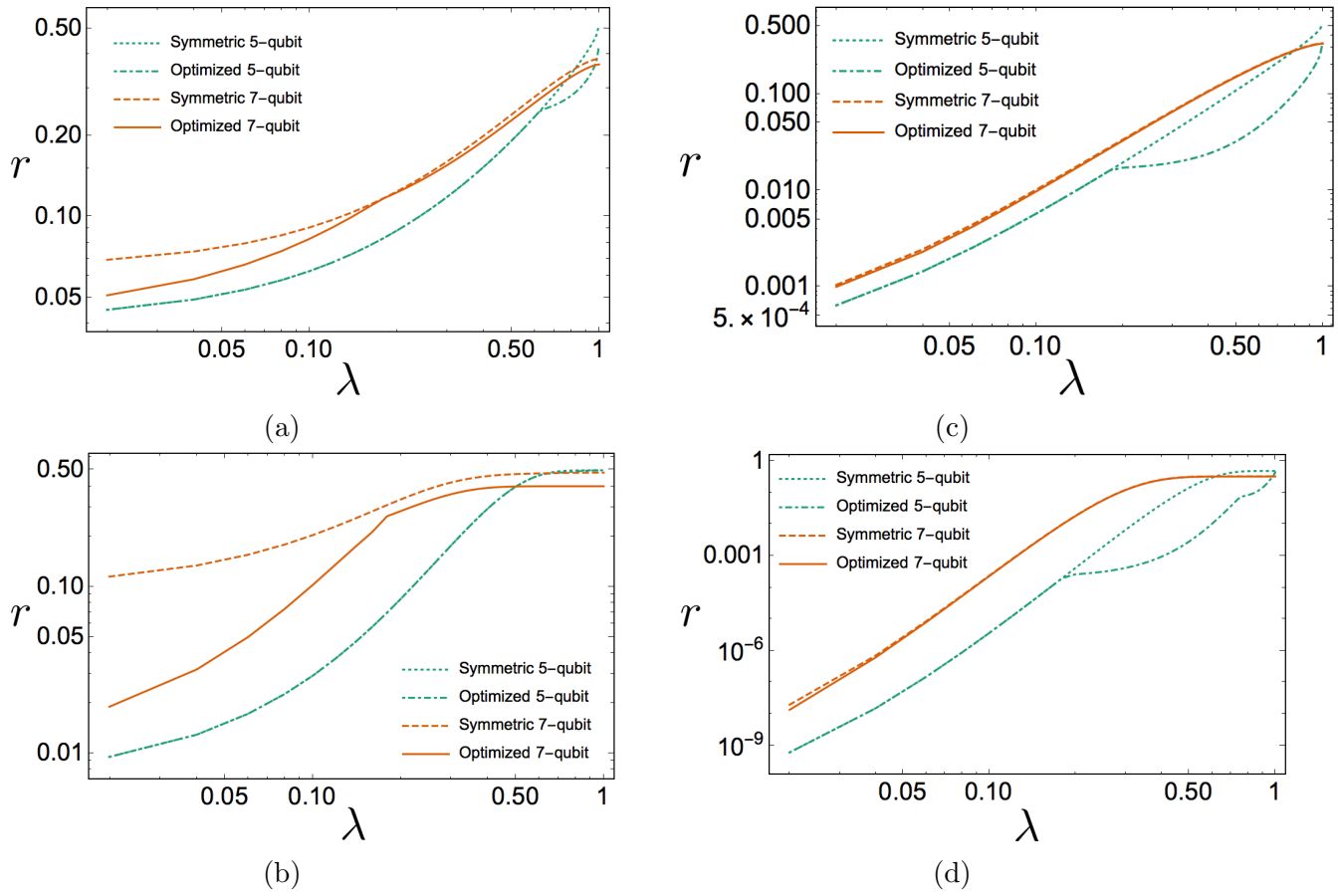


Figure 2.5: Infidelity curves r as functions of the dephasing parameter λ at the first (a), (c) and third (b), (d) concatenation level for the amplitude-phase damping channel under the $[\![5, 1, 3]\!]$ and Steane codes. The amplitude damping rate is fixed at $p = 0.17$ (a), (b) and $p = 0.01$ (c), (d). The symmetrized and optimized curves overlap when the optimized decoder is the symmetric decoder, however, there are many regimes where the optimized decoder improves the infidelity. For the $[\![5, 1, 3]\!]$ code, infidelities are lowered by as much as 2 orders of magnitude. The curves for the fixed symmetric decoders are all smooth, whereas the curves for the optimized decoders have kinks corresponding to points where the decoder changes to exploit asymmetries in the noise. All codes also exhibit improved performance for large values of p relative to λ, that is, when the noise is primarily amplitude damping.

for the Shor and surface-17 codes are not shown since they behave similarly to the Steane code. The $[\![5,1,3]\!]$ code generally outperforms all other codes in terms of logical infidelity and thresholds under both optimized and symmetric decoders, except in an intermediate regime where the optimized decoder exploits the asymmetry in the stabilizers of the X-Shor code.

The optimized decoder coincides with the symmetric decoder for each code in some parameter regimes, although only when $p = 0$ for the Steane and Shor codes. However, the optimized decoder often differs significantly from the symmetric decoder, resulting in substantially improved logical infidelities and thresholds. The optimized decoder changes in different parameter regimes to exploit asymmetries in the noise, producing the kinks in the curves in Fig. 2.4 and Fig. 2.5. The amplitude-phase damping channel is highly biased towards Z errors for small values of p relative to λ. The optimized decoder exploits this for the $[\![5,1,3]\!]$ code by only correcting Z errors for t levels of concatenation until the noise is approximately symmetric, and then switching to the symmetric decoder, with t increasing as p approaches zero. The X-Shor code also performs better in this regime as it has more X-type stabilizers that detect Z-type errors.

The optimized decoder also results in improved thresholds and logical infidelities for high amplitude damping rates for all codes. The noise is significantly different from Pauli noise in this regime and so decoders constructed under the assumption of Pauli noise will be less likely to identify the correct error compared to decoders optimized for amplitude-phase damping.

As discussed in Sec. 2.3.2, multiple sets $\{L_g : g \in \mathscr{G}(\mathcal{M})\}$ maximize Eq. 2.37 for the Steane code with amplitude-phase damping and large values of λ. For example, setting $\lambda = 0.1431$ and choosing the first recovery operator that maximizes Eq. 2.37 gives a threshold of $p_{th} = 0.1032$. However, searching all tuples $\{L_g : g \in \mathscr{G}(\mathcal{M})\}$ that maximize Eq. 2.37 (where the degeneracy only occurs at the first level) for the same value of λ gives a higher threshold of $p_{th} = 0.1150$.

2.6 Thresholds for coherent errors

In this section we illustrate the behavior of coherent errors under error correction. We consider a coherent error noise model where every qubit undergoes a rotation by an unknown angle θ about an axis of rotation $\hat{n}$. The coherent noise channel can thus be written as

$$\mathcal{N}_{\theta,\phi,\gamma}(\rho) = e^{i\theta\hat{n}\cdot\vec{\sigma}}\rho e^{-i\theta\hat{n}\cdot\vec{\sigma}}, \tag{2.43}$$

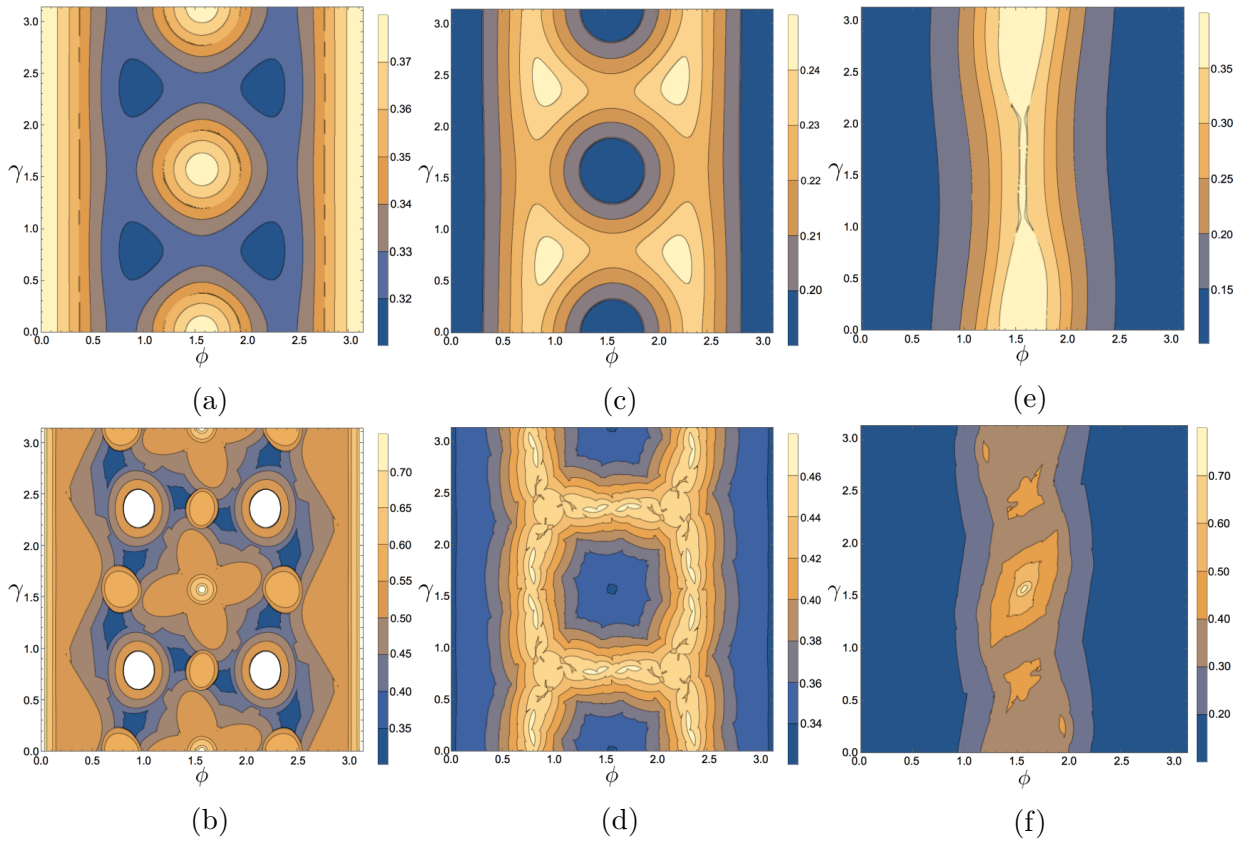


Figure 2.6: Contour plots representing hypersurfaces of the threshold value of θ for rotations around the axis $\hat{n} = (\sin\phi\cos\gamma, \sin\phi\sin\gamma, \cos\phi)$ for (a) the $[\![5,1,3]\!]$ code, (c) the Steane code and (e) the Shor code using the symmetric decoder and (b) the $[\![5,1,3]\!]$ code, (d) the Steane code and (f) the Shor code using optimized decoding. The optimized decoder uses transversal gates to improve the threshold, particularly when the rotation is around an eigenbasis of a transversal Clifford gate (see Table. 2.1). In particular, the $[\![5,1,3]\!]$ code has a transversal $\pi/3$ rotation around $\hat{n}_C = (1,1,1)/\sqrt{3}$ (i.e., $\gamma = \pi/4$, $\phi = \pi/3$), which enables the optimized decoder to correct *arbitrary* rotations around axes close to $\hat{n}_C$, illustrated by the white circular regions in (b). For Steane's code, the lightest colored regions in (d) corresponds to threshold angles $\theta_{th} \approx 0.46$ compared to $\theta_{th} \approx 0.24$ in (c), an improvement by almost a factor of 2. The Shor code has no transversal non-Pauli gates and so the improvements from the optimized decoder are not as substantial. However, for rotations near the y-axis, the Shor code outperforms the Steane code by a factor of at most 2.3.

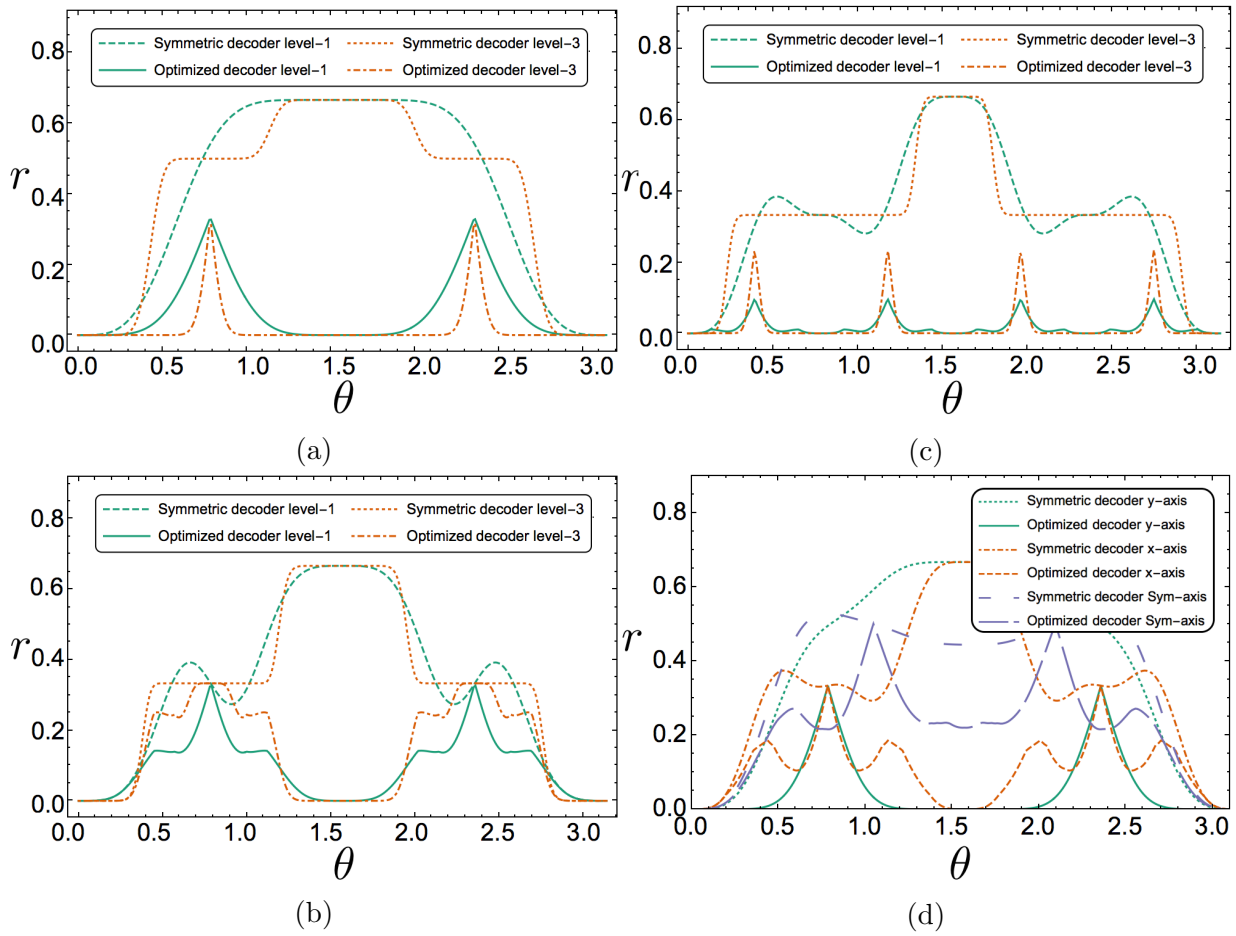


Figure 2.7: In (a), (b) and (c) infidelities r of the $[\![5,1,3]\!]$ code, Shor code and Steane code are plotted at the first and third levels for a rotation about the x-axis. In (a), the optimized infidelity curves are peaked at the code's threshold value $\theta_{th} = \pi/4$. In (b), the peaks of the optimized infidelity curves are centred slightly above the code's threshold value $\theta_{th} = 0.3396$. However, the optimized level-3 infidelity curve intersects the optimized level-1 curve at the threshold value as expected. In (c), the optimized level-1 and level-3 infidelity curves intersect at the threshold value $\theta_{th} = 0.3692$. The peaks of the infidelity curves occur at $\theta = \pi/4$ due to the codes symmetry. In (d), infidelity plots of the surface-17 at the first concatenation level are shown for a rotation about the y-axis, x-axis and the $(1,1,1)/\sqrt{3}$ axis. It can be seen that the infidelity is lowest for rotations about the y-axis. In all 4 plots, it can be seen that applying our hard decoding algorithm reduces the infidelities by, in some cases, orders of magnitude compared to the symmetric decoder.

where $\hat{n} = (\sin\phi\cos\gamma, \sin\phi\sin\gamma, \cos\phi)$. We obtain the threshold hypersurface for the noise model

$$\mathscr{N} = \{\mathcal{N}_{\theta,\phi,\gamma} : \theta \in [0, 2\pi], \gamma, \phi \in [0, \pi]\}, \tag{2.44}$$

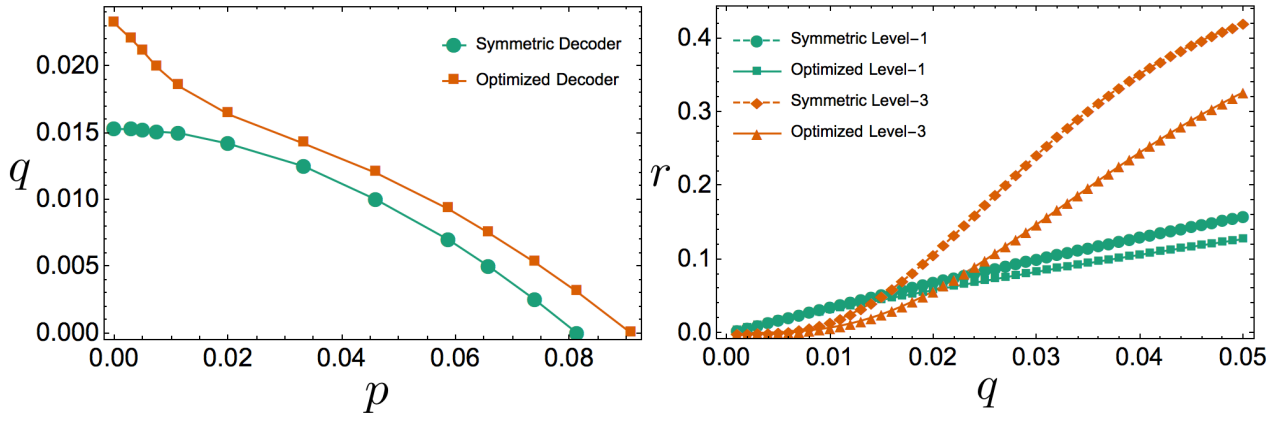
by fixing γ and ϕ and obtaining the threshold for θ.

The threshold hypersurfaces for the $[\![5, 1, 3]\!]$, Steane, and Shor codes are illustrated as contour plots in Fig. 2.6. The infidelities at the first and third concatenation levels for $[\![5, 1, 3]\!]$, Steane, Shor, and surface-17 (1st level only) codes are plotted in Fig. 2.7.

Unlike with amplitude-phase damping noise, the optimized decoder strictly outperforms the symmetric decoder for all rotation axes. With the exception of the Shor code, the threshold hypersurfaces are relatively flat under symmetric decoding, that is, the threshold rotation angle is relatively independent of the rotation axis. The optimized decoder breaks this, giving larger threshold angles for different axes, particularly for rotations about an eigenbasis of the transversal Clifford gates listed in Table. 2.1. In particular, the $[\![5, 1, 3]\!]$ code can correct any rotation about axes $\hat{n}$ that are close to $(\pm, \pm, \pm)/\sqrt{3}$. The Steane code can correct any rotation about the Pauli axes except for angles close to odd integer multiples of $\pi/4$. The performance of the Shor code is generally only modestly improved by the optimized decoder, largely because the Shor code has no transversal non-Pauli gates. However, the optimized decoder is able to exploit the asymmetries in the stabilizer generators to increase the threshold for rotations near the y axis by more than a factor of 3.

The improved threshold angles are reflected in the orders-of-magnitude reduction in the logical infidelities in Fig. 2.7. The infidelities are periodic because transversal gates can be used to counteract the unitary noise. However, as discussed in Sec. 2.8, the transversal gates are useful even when the action of the noise on the codespace is far from a transversal gate. For the $[\![5, 1, 3]\!]$, Steane and Z-Shor codes, the infidelities are the infidelity at the first and third concatenation levels for rotations about the x axis, while for the surface-17 code the infidelities are at the first level for rotations about the x, y and $\frac{1}{\sqrt{3}}(1, 1, 1)$ axes. For the $[\![5, 1, 3]\!]$, Steane and Shor codes, the threshold values of θ correspond exactly to the cross-over points between the level-1 and level-3 curves. At the third level, the infidelity is greatly suppressed below threshold and increased above threshold. The optimized infidelity curves are lower (in some cases by several orders of magnitude) than the infidelities arising by applying the symmetric decoder at all levels.

For the $[\![5, 1, 3]\!]$ code, the only uncorrectable values of θ are odd integer multiples of $\pi/4$. The surface-17 code has a higher threshold against Y errors than against X (or Z) errors. The surface-17 code treats X and Z errors symmetrically. However, since the X



(a) (b)

Figure 2.8: (a) Threshold curves and (b) infidelities r at the first and third concatenation level (with fixed $p = 0.003$) for the $[\![7, 1, 3]\!]$ code. Two-qubit correlated dephasing occurs with probability q and depolarizing noise occurs with probability $1-q$, with a depolarizing noise parameter p (see Eq. 2.45 and Eq. 2.46). For small values of p, the Z errors arising from the two-qubit correlations dominate the noise. Applying our optimized hard decoding algorithm in this regime yields a threshold of $q_{th} = 0.0232$. The contribution from depolarizing noise increases with p until the noise is predominantly depolarizing. In this regime, the optimized decoder implements the standard CSS decoder at all levels. When $q = 0$, the noise is purely depolarizing and $p_{th} = 0.0908$. For all values of p, the threshold q_{th} obtained by implementing our optimized decoder is larger than the threshold obtained by implementing the symmetric decoder. The level-1 and level-3 infidelity curves intersect near the respective thresholds for $p = 0.003$, namely, $q_{th} = 0.0153$ and $q_{th} = 0.0220$ for the symmetric and optimized decoders respectively.

and Z stabilizer generators have support on different qubits, error rates resulting from Y errors will differ from error rates resulting from X and Z errors.

The optimized decoding algorithm gives the greatest improvements for codes with transversal non-Pauli gates, namely, the $[\![5, 1, 3]\!]$ and Steane codes.

2.7 Correlated noise channel

In this section, we study the effect of correlated noise on the logical noise in Steane's code. The correlated noise we consider consists of local depolarizing noise and two-qubit

correlated dephasing errors to all adjacent pairs. The composite noise channel maps an n-qubit state ρ

$$\mathcal{N}(\rho) = (1-q)\mathcal{D}_p^{\otimes n}(\rho) + \frac{q}{n}\sum_{j=1}^{n}\mathcal{Z}_{j,j+_n 1}^{(2)}(\rho), \tag{2.45}$$

where $j +_n 1 = j+1$ if $j < n$ and 1 otherwise (that is, we consider the qubits to be in a ring),

$$\mathcal{D}_p(\tau) = (1-p)\tau + \frac{p}{3}(X\tau X + Y\tau Y + Z\tau Z) \tag{2.46}$$

is the depolarizing channel acting on a single-qubit state τ, and $\mathcal{Z}_{j,j+_n 1}(\rho) = Z_j Z_{j+1}\rho Z_{j+1}Z_j$ applies phase-flip operators to qubits j and $j+1$. The logical process matrix can be computed for the noise model in Eq. 2.45 by Eq. 2.24.

The threshold and infidelity at the first and third concatenation levels of Steane's code are illustrated in Fig. 2.8. In the small p regime, the noise is dominated by the two-qubit correlated dephasing contribution. The optimized decoder corrects a larger amount of Z errors at the first few levels by breaking the symmetry in the syndrome measurements. At higher levels, the decoder corrects in a more symmetric fashion in order to remove the remaining Pauli errors. This improved performance is also illustrated in the reduced optimized logical infidelities shown in Fig. 2.8 (b) as a function of q with $p = 0.003$.

There is an intermediate regime where the local depolarizing noise contribution becomes more relevant, leading to a decrease in the threshold value for q. However, the optimized threshold is still noticeably larger than the symmetric decoder threshold. Finally, when the local depolarizing noise is the dominant source of noise, our optimization algorithm chooses recovery maps consistent with the standard CSS decoder. The standard CSS decoder yields a slightly larger p threshold value compared to the symmetric decoder when $q = 0$.

2.8 The effect of Pauli twirling on thresholds and the benefits of using transversal operations

In Sec. 2.5, Sec. 2.6 and Sec. 2.7, we showed that our hard decoding optimization algorithm could improve threshold values by more than a factor of 2 for amplitude-phase damping noise. For coherent noise there where certain rotation axes where the noise was correctable for arbitrary rotation angles. Infidelities were reduced by orders of magnitudes in certain

regimes. The amplitude-phase damping and coherent noise models are both non-Pauli. Performing a Pauli twirl on a noise channel $\mathcal{N}$ (that is, conjugating it by a uniformly random Pauli channel) maps it to a channel $\mathcal{T}(\mathcal{N})$ that is a Pauli channel and so has a diagonal matrix representation with respect to the Pauli basis [40, 27]. In [27, 26], the effective noise at the first level for the amplitude damping channel was found to be in good agreement to a Pauli twirled approximation of the channel.

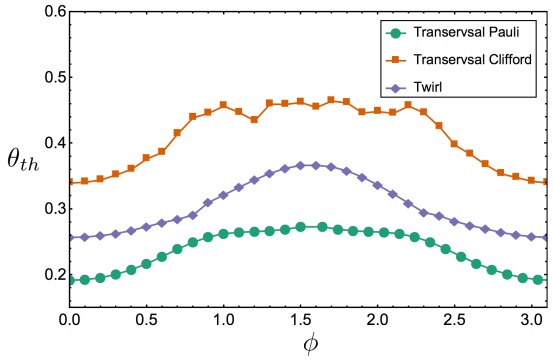


Figure 2.9: Threshold curves for unitary rotations about $(\frac{1}{\sqrt{2}}\sin\phi, \frac{1}{\sqrt{2}}\sin\phi, \cos\phi)$ by an angle θ under three different decoding schemes for the Steane code. For all values of ϕ, an improvement by as much as a factor of 1.7 in the threshold θ_{th} obtained by using our algorithm optimizing over all transversal Clifford gates can be observed relative to optimizing over all Pauli gates. The Pauli-twirl reduces (increases) the threshold when optimizing over all transversal Clifford (Pauli) gates for all values of ϕ.

However, we now show that performing a Pauli twirl on coherent noise and using the Steane code can either reduce or increase threshold values, depending on the particular recovery protocol. We also illustrate the improvements obtained by using all transversal gates in the decoding algorithm, instead of just the Pauli gates. Threshold curves for rotations about $(\frac{1}{\sqrt{2}}\sin\phi, \frac{1}{\sqrt{2}}\sin\phi, \cos\phi)$ by an angle θ under three different decoding schemes for the Steane code are presented in Fig. 2.9. The three schemes we consider are : 1) our optimized decoding algorithm applied to the twirled noise; 2) our optimized decoding algorithm applied to the bare noise using all transversal gates; and 3) our optimized decoding algorithm applied to the bare noise using only transversal Pauli gates. Using transversal Clifford gates in our recovery protocol gives the largest threshold values for all values of ϕ and so Pauli twirling reduces the threshold. However, if only transversal Pauli operators are used, Pauli twirling increases the threshold for all values of ϕ.

The curves in Fig. 2.9 also demonstrate that the threshold can increase by at most a factor of 1.7 when optimizing over all transversal gates for coherent noise compared to

optimizing over all transversal gates for the twirled channel. This advantage arises for two reasons. First, for a known noise model, a transversal gate can be applied to map it to another noise model that may be closer to the identity. Second, syndrome measurements may map coherent errors closer to a non-Pauli unitary. However, both these benefits are lost when the noise is twirled because both the physical noise and the noise for each syndrome is Pauli noise, which is generally far from any non-Pauli unitary.

2.9 Sensitivity and robustness of our hard decoding optimization algorithm to perturbations of the noise model

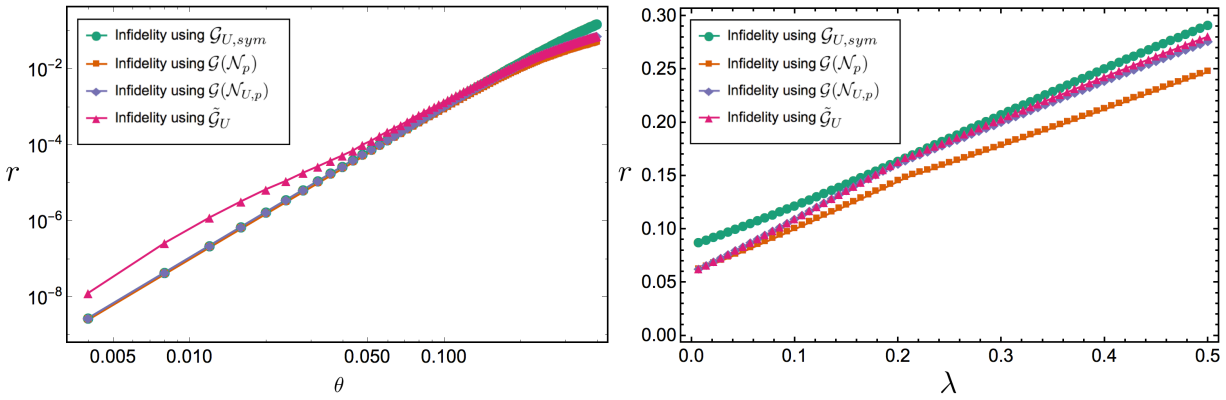
In Sec. 2.3, we presented a hard decoding algorithm for optimizing threshold values of an error correcting code for arbitrary CPTP maps. Our algorithm can therefore be applied to non-Pauli channels, including more realistic noise models that could be present in current experiments. However, the noise afflicting an experimental system is only ever approximately known. Nevertheless, we now demonstrate that applying the decoder obtained by our algorithm for a fixed noise channel $\mathcal{N}$ to a perturbed noise model $\mathcal{N}_p$ retains, in some cases, improvements in error suppression relative to the symmetric decoder.

To study perturbations about a noise channel, let $\mathscr{N} = \{\mathcal{N}_p : p \in [0,1]\}$ be a 1-parameter noise model and

$$\mathcal{N}_{U,p}(\rho) = [1 - f(p)]\mathcal{N}_p + f(p)U\rho U^\dagger \tag{2.47}$$

where U is a random unitary and $f : [0,1] \to [0,1]$ is a function such that $f(p) \ll \|\mathcal{N}_p - \mathcal{I}\|$ for any suitable norm (e.g., the diamond norm). (The generalization to multi-parameter noise families is straightforward.)

We applied our algorithm to $\mathcal{N}_p$ and $\mathcal{N}_{U,p}$, giving the effective process matrices $\mathcal{G}(\mathcal{N}_p)$ and $\mathcal{G}(\mathcal{N}_{U,p})$ respectively. We then applied the symmetric decoder and the decoder optimized for $\mathcal{N}_p$ to the perturbed noise $\mathcal{N}_{U,p}$ to obtain the process matrices $\mathcal{G}_{U,sym}$ and $\tilde{\mathcal{G}}_U$ respectively. The infidelities of these process matrices (at the first concatenation level) are plotted in Fig. 2.10 (a) for the $[\![5,1,3]\!]$ code with coherent noise and $f(\theta) = \sin^2\theta/10$ and in Fig. 2.10 (b) for the Steane code with amplitude-phase damping, $p = 0.2$ and $f(\lambda) = \lambda/10$. For both plots we averaged the values over 100 uniformly random unitaries. These results demonstrate that the significant improvements obtained using the optimized decoder are, in most studied cases, robust to perturbations in the noise.



(a) (b)

Figure 2.10: Averaged infidelity plots over 100 random unitary operators U of the effective process matrices $\mathcal{G}(\mathcal{N}_p)$, $\mathcal{G}(\mathcal{N}_{U,p})$, $\mathcal{G}_{U,sym}$ and $\tilde{\mathcal{G}}_U$. The figure in (a) is obtained using the $[\![5,1,3]\!]$ code for coherent errors using random rotation axes for each random unitary. The perturbation was chosen to have the form $f(\theta) = \sin^2\theta/10$. The figure in (b) is obtained using the Steane code for the amplitude-phase damping channel. The perturbation was chosen to have the form $f(\lambda) = \lambda/10$. In (a), the inset plot shows all infidelities on a log-log scale in the regime where θ is small. As can be seen from the figure, in the regime where $\theta \gtrsim 0.185$, the optimized recovery maps for the unperturbed channel yield a lower infidelity when applied to the perturbed channel than that from applying the symmetric decoder. For smaller rotation angles, the infidelity from $\tilde{\mathcal{G}}_U$ is slightly larger than the infidelity arising from $\mathcal{G}_{U,sym}$. The two differ by at most a factor of 7 in the small θ limit. The infidelity obtained by applying the hard decoding optimization algorithm to the unperturbed channel is lowest for all sampled values of θ. In (b), it can be observed that applying the decoder chosen by our optimization algorithm for the unperturbed channel to the perturbed channel results in a lower infidelity than applying the symmetric decoder to the perturbed channel, for all sampled values of λ. This indicates that our decoding scheme is very robust to small perturbations of the amplitude-phase damping channel.

The one exception we observed is for coherent noise in the $[\![5,1,3]\!]$ code for $\theta \lesssim 0.185$, where the infidelity obtained using the optimized decoder for the unperturbed channel is larger than that obtained using the symmetric decoder by a factor of at most 7.

2.10 Summary and outlook

In this section, we presented an optimized hard decoding algorithm for arbitrary local Markovian noise and numerical techniques to characterize thresholds for noise models. Block-wise two-qubit correlated noise was also considered. Using the analytical tools of Sec. 2.2, we provide numerical results in Sec. 2.5 to Sec. 2.9 which shows substantial improvements obtained by our algorithms compared to a fixed decoder for a variety of noise models, including coherent errors, correlated dephasing and amplitude-phase damping, and codes, namely, the $[\![5, 1, 3]\!]$, Steane, Shor and surface-17 codes. For coherent noise, our optimized decoding algorithm allowed, in some cases, the noise to be corrected for all sampled rotation angles and reduced infidelities at a fixed concatenation level by orders of magnitude.

Our hard decoding algorithm is scalable and efficiently optimizes the recovery operations independently at each concatenation level while taking advantage of a code's transversal gates. At a given concatenation level, all syndrome measurements are considered rather than being sampled from a distribution, so that the performance is exactly characterized rather than containing statistical (and state-dependent) uncertainties.

In contrast to hard decoding, message-passing algorithms [29] can increase thresholds for Pauli noise, in some cases nearing the hashing bound subject to sampling uncertainties. Large codes can also be studied using tensor networks [31], although this requires a tensor-network description of the code and is exponential in the code distance. An interesting and important open problem is to combine the current techniques with those of Refs. [29, 31] to either reduce statistical uncertainties in message-passing algorithms by exploiting symmetries in the code or to treat larger, non-concatenated code families.

Further, we showed that performing a Pauli twirl can increase or decrease the threshold depending on the code and noise properties. In [25], the Pauli twirl was found to have little impact on the performance of amplitude damping, which is known to be "close" to Pauli noise (that is, exhibit similar worst-case errors) [41]. We conjecture that Pauli twirling will generally reduce thresholds for codes that have many transversal gates, but may improve performance for codes with fewer transversal gates.

Lastly, we considered the robustness of our hard decoding optimization algorithm to noise channels that were not perfectly known. We showed that by optimizing our decoder for a channel that was slightly perturbed by a random unitary operator from the actual channel acting on the qubits, it was still possible to obtain improved error rates over the symmetric decoder. However, there are some circumstances where the optimized decoder, while still being robust, is outperformed by the symmetric decoder. Determining the

robustness of decoders is an open problem that will be especially relevant when decoders are used for experimental systems with incompletely characterized noise.

In Refs. [25, 26], the process matrix formalism was used to obtain pseudo-thresholds for the Steane code using the standard CSS decoder. Measurement errors were taken into account, resulting in more accurate pseudo-threshold values. Our methods were developed assuming that the encoding and decoding operations were perfect. The next step in our work will be to generalize our results to include measurement and state-preparation errors.

Chapter 3

Flag fault-tolerant error correction with arbitrary distance codes

The material of this chapter is based on the journal article of Ref. [42], copyrighted by the Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. In this work, myself and Michael Beverland were the investigators of the research. I had the idea for the project, developed most of the framework and theoretical ideas as well as perform all numerical simulations. Both authors contributed to the final scheme which was developed to realize the research goal of the project. Additionally, both authors contributed to writing the manuscript and editing the work for publication.

3.1 Introduction and formalism

There are three general approaches of fault-tolerant error correction applicable to a wide range of stabilizer codes due to Shor [17] (see Sec. 1.2.1), Steane [43], and Knill [44] (we will give detailed descriptions of Steane and Knill-EC in Sec. 4.2.2 and Sec. 4.2.3). There are also a number of promising code-specific FTEC schemes, most notably the surface code with a minimum weight matching error correction scheme [45, 46, 7]. This approach gives very competitive fault-tolerant thresholds and only requires geometrically local measurements (the best fault-tolerance thresholds are obtained from Knill error correction using concatenated error-detecting codes for post-selected ancilla creation). A high threshold [17, 47, 48, 49] implies that relatively imperfect hardware could be used to reliably implement long quantum computations. Despite this, the hardware and overhead requirements

for the surface code are sufficiently demanding that it remains extremely challenging to implement in the lab.

Fortunately, there are reasons to believe that there could be better alternatives to the surface code. For example, dramatically improved thresholds could be possible using concatenated codes if they enjoyed the same level of optimization as the surface code has in recent years [29, 50]. Another enticing alternative is to find and use efficiently-decodable low density parity check (LDPC) codes with high rate [51, 52, 53] in a low-overhead FTEC protocol [54]. For these and other reasons, it is important to have general FTEC schemes applicable to a wide range of codes and to develop new schemes.

Shor EC can be applied to any stabilizer code, but typically requires more syndrome measurement repetitions than Steane and Knill EC. Furthermore, all weight-w stabilizer generators are measured sequentially using w-qubit verified cat states. On the other hand, Steane EC has higher thresholds than Shor EC and has the advantage that all Clifford gates are applied transversally during the protocol. However, Steane EC is only applicable to CSS [12, 43] codes and uses a verified logical $|+\rangle$ state encoded in the same code to simultaneously obtain all X-type syndromes, using transversal measurement (similarly for Z). Knill EC can also be applied to any stabilizer code but requires two additional ancilla code blocks (encoded in the same code that protects the data) prepared in a logical Bell state. The Bell state teleports the encoded information to one of the ancilla code blocks and the extra information from the transversal Bell measurement gives the error syndrome. Knill EC typically achieves higher thresholds than Shor and Steane EC but often uses more qubits [55, 56]. It is noteworthy that for large LDPC codes, in which low weight generators are required to be fault-tolerantly measured, Shor EC is much more favourable than Steane or Knill EC. Many improvements in these schemes have been made. For example, in [20], ancilla decoding was introduced to correct errors arising during state preparation in Shor and Steane EC rather than simply rejecting all states which fail the verification procedure.

In this chapter, we build on a number of recent papers [57, 58, 59] that demonstrate flag error correction for particular distance-three and error detecting codes and present a general protocol for arbitrary distance codes. Flag error correction uses extra ancilla qubits to detect potentially problematic high weight errors that arise during the measurement of a stabilizer. We provide a set of requirements for a stabilizer code (along with the circuits used to measure the stabilizers) which, if satisfied, can be used for flag error correction. We are primarily concerned with extending the lifetime of encoded information using fault-tolerant error correction and defer the study of implementing gates fault-tolerantly to future work. Our approach can be applied to a broad class of codes (including but not limited to surface codes, color codes and quantum Reed-Muller codes). Of the three general schemes described above, flag EC has most in common with Shor EC. Further, flag EC does not

require verified state preparation, and for all codes considered to date, requires fewer ancilla qubits. Lastly, we note that in order to satisfy the fault-tolerant error correction definition presented in Sec. 1.2.1, our protocol applied to distance-three codes differs from [57].

We foresee a number of potential applications of these results. Firstly we believe it is advantageous to have new EC schemes with different properties that can be used in various settings. Secondly, flag EC involves small qubit overhead, hence possibly the schemes presented here and in other flag approaches [57, 58, 59] will find applications in early qubit-limited experiments. Thirdly, we expect the flag EC protocol presented here could potentially be useful for LDPC codes as described in [54].

In Sec. 3.2.1 we provide important definitions and introduce flag FTEC for distance-three and -five codes. In Sec. 3.3 we apply the protocol to two examples: the $[\![19, 1, 5]\!]$ and $[\![17, 1, 5]\!]$ color codes, which importantly have a variety of different weight stabilizers. The general flag FTEC protocol for arbitrary distance codes is given in Sec. 3.4.1. A proof that the general protocol satisfies the fault-tolerance criteria is given in Appendix. B.1. In Sec. 3.4.2 we provide examples of codes that satisfy the conditions that we required for flag FTEC. Flag circuit constructions for measuring stabilizers of the codes in Sec. 3.4.2 are given in Sec. 3.5. We also provide a candidate circuit construction for measuring arbitrary weight stabilizers in Appendix. B.3. In Sec. 3.6, we analyze numerically a number of flag EC schemes and compare with other FTEC schemes under various types of circuit level noise. We find that flag EC schemes, which have large numbers of idle qubit locations, behave best in error models in which idle qubit errors occur with a lower probability than CNOT errors. The remainder of this section is devoted to some definitions and noise model/simulation methods.

3.1.1 Definitions, noise model and pseudo-threshold calculations

We first start with a definition that will be used several times in this section,

Definition 6. *Weight-t Pauli operators*

$$\mathcal{E}_t = \{E \in \mathcal{P}_n | wt(E) \leq t\}, \tag{3.1}$$

where $\mathcal{P}_n$ is the n-qubit Pauli group.

In Sec. 3.6, we perform a full circuit level noise analysis of various error correction protocols. Unless otherwise stated, we use the following depolarizing noise model:

1. With probability p, each two-qubit gate is followed by a two-qubit Pauli error drawn uniformly and independently from $\{I, X, Y, Z\}^{\otimes 2} \setminus \{I \otimes I\}$.
2. With probability $\frac{2p}{3}$, the preparation of the $|0\rangle$ state is replaced by $|1\rangle = X|0\rangle$. Similarly, with probability $\frac{2p}{3}$, the preparation of the $|+\rangle$ state is replaced by $|-\rangle = Z|+\rangle$.
3. With probability $\frac{2p}{3}$, any single qubit measurement has its outcome flipped.
4. Lastly, with probability $\tilde{p}$, each resting qubit location is followed by a Pauli error drawn uniformly and independently from $\{X, Y, Z\}$.

Some error correction schemes that we analyze contain a significant number of idle qubit locations. Consequently, most schemes will be analyzed using three ratios ($\tilde{p} = p$, $\tilde{p} = p/10$ and $\tilde{p} = p/100$) to highlight the impact of idle qubit locations on the logical failure rate.

The two-qubit gates we consider are: CNOT, XNOT= H_1(CNOT)H_1, and CZ= H_2(CNOT)H_2.

Logical failure rates are estimated using an N-run Monte Carlo simulation. During a particular run, errors are added at each location following the noise model described above. Once the error locations are fixed, the errors are propagated through a fault-tolerant error correction circuit and a recovery operation is applied. After performing a correction, the output is ideally decoded to verify if a logical fault occurred. For an error correction protocol implemented using a stabilizer code C and a fixed value of p, we define the logical failure rate

$$p_{\mathrm{L}}^{(C)}(p) = \lim_{N\to\infty} \frac{N_{\mathrm{fail}}^{(C)}(p)}{N}, \tag{3.2}$$

where $N_{\mathrm{fail}}^{(C)}(p)$ is the number of times a logical X *or* logical Z error occurred during the N rounds. In practice we take N sufficiently large to estimate $p_{\mathrm{L}}^{(C)}(p)$, and provide error bars [60, 36].

In this section we are concerned with evaluating the performance of FTEC protocols (i.e. we do not consider performing logical gates fault-tolerantly). We define the pseudo-threshold of an error correction protocol to be the value of p such that

$$\tilde{p}(p) = p_L^{(C)}(p). \tag{3.3}$$

Note that it is important to have $\tilde{p}$ on the left of Eq. 3.3 instead of p since we want an encoded qubit to have a lower logical failure rate than an unencoded idle qubit. From the above noise model, a resting qubit will fail with probability $\tilde{p}$.

3.2 Flag error correction for small distance codes

In this and the next section, we present a t-fault-tolerant flag error correction protocol with distance-$(2t+1)$ codes satisfying a certain condition. Our approach extends that introduced by Chao and Reichardt [57] for distance three codes, which we first review using our terminology below. We then present the protocol for distance five CSS codes which contains most of the main ideas of the general case (which is provided in Sec. 3.4). Lastly, in Sec. 3.3 we provide examples of how the protocol is applied to the $[\![19, 1, 5]\!]$ and $[\![17, 1, 5]\!]$ color codes.

3.2.1 Definitions and Flag 1-FTEC with distance-3 codes

In what follows, we use the term location to refer to a gate, state preparation, measurement or idle qubit where a fault may occur. Note also that a two-qubit Pauli error $P_1 \otimes P_2$ arising at a two-qubit gate location counts as a single fault.

It is well known that with only a single measurement ancilla, a single fault in a blue CNOT of the stabilizer measurement circuit shown in Fig. 3.1a can result in a multi-weight error on the data block. This could cause a distance-3 code to fail, or more generally could cause a distance-d code to fail due to fewer than $(d-1)/2$ total faults. We therefore say the blue CNOTs are *bad* according to the following definition:

Definition 7. *Bad locations*

A circuit location in which a single fault can result in a Pauli error E on the data block with $\mathrm{wt}(E) \geq 2$ will be referred to as a bad location.

As shown in Fig. 3.1b, the circuit can be modified by including an additional ancilla (flag) qubit, and two extra CNOT gates. This modification leaves the bad locations and the fault-free action of the circuit unchanged. However, any single fault leading to an error E with $\mathrm{wt}(E) \geq 2$ will also cause the measurement outcome of the flag qubit to flip [57]. The following definitions will be useful:

Definition 8. *Flags and measurements*

Consider a circuit for measuring a stabilizer generator that includes at least one flag ancilla. The ancilla used to infer the stabilizer outcome is referred to as the measurement qubit. We say the circuit has flagged if the eigenvalue of a flag qubit is measured as -1. If the eigenvalue of a measurement qubit is measured as -1, we will say that the measurement qubit flipped.

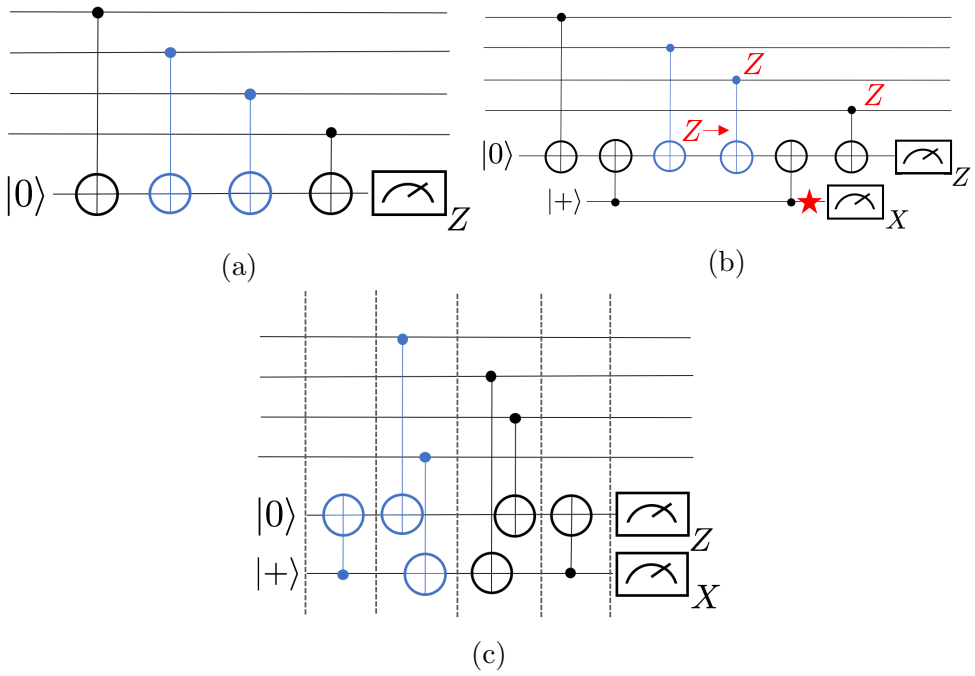


Figure 3.1: Circuits for measuring the operator $ZZZZ$ (can be converted to any Pauli by single qubit Cliffords). (a) Non-fault-tolerant circuit. A single fault IZ occurring on the third CNOT (from the left) results in the error $IIZZ$ on the data block. (b) Flag version of Fig. 3.1a. An ancilla (flag) qubit prepared in $|+\rangle$ and two extra CNOT gates signals when a weight two data error is caused by a single fault. Subsequent rounds of error correction may identify which error occurred. Consider an IZ error on the second CNOT, in the non-flag circuit this would result in a weight two error, but in this case, this fault causes the circuit to flag. (c) An alternative flag circuit with lower depth than (b). All bad locations are illustrated in blue.

The purpose of flag qubits is to signal when high weight data qubit errors result from few fault locations during a stabilizer measurement. Two key definitions are:

Definition 9. *t-flag circuit*

A circuit[1] $C(P)$ *which, when fault-free, implements a projective measurement of a weight-w Pauli P without flagging is a t-flag circuit if the following holds: For any set of* $v \leq t$ *faults in* $C(P)$ *resulting in an error E with* $min(wt(E), wt(EP)) > v$*, the circuit flags.*

Note that a t-flag circuit for measuring a weight-t stabilizer P is also a k-flag circuit for any $k > t$. In Sec. 3.5 we give constructions for some t-flag circuits.

Definition 10. *Flag error set*

Let $\mathcal{E}(g_i)$ *be the set of all errors caused by one fault which caused the circuit* $C(g_i)$ *to flag.*

Note that the flag error set can contain the identity (for instance, when only a measurement error occurs) as well as weight one errors.

Suppose all errors in a flag error set $\mathcal{E}(g)$ for a 1-flag circuit $C(g)$ have distinct syndromes. As $C(g)$ is a 1-flag circuit, a single fault that leads to an error of weight greater than one will cause the circuit $C(g)$ to flag. Moreover, when a flag has occurred due to at most one fault, a complete set of fault-free stabilizer measurements will infer the resulting element of the flag error set which has been applied to the data qubits. In fact, one would only require distinct syndromes for errors in the flag error set that are logically inequivalent, as defined in Def. 3 (see Sec. 3).

As an example, consider the 1-flag circuit in Fig. 3.1b. A single fault at any of the blue CNOT gates can lead to an error E_b with $\text{wt}(E_b) \leq 2$ on the data. The set $\mathcal{E}(Z^{\otimes 4})$ contains all errors E_b which resulted from a fault at a blue CNOT gate causing the circuit $C(Z^{\otimes 4})$ of Fig. 3.1b to flag, i.e., $\mathcal{E}(g) = \{I, Z_{q_3}Z_{q_4}, X_{q_2}Z_{q_3}Z_{q_4}, Z_{q_1}X_{q_2}, Z_{q_4},\ X_{q_3}Z_{q_4}, Y_{q_3}Z_{q_4}\}$ with qubits q_1 to q_4.

With the above definitions, we can construct a fault-tolerant flag error correction protocol for $d = 3$ stabilizer codes satisfying the following condition.

Definition 11. ***Flag* 1*-FTEC condition:***

[1] To avoid confusing the notation of $C(P)$ that represents a circuit and C that represents a code space, we always include the measured Pauli in parenthesis unless clear from context.

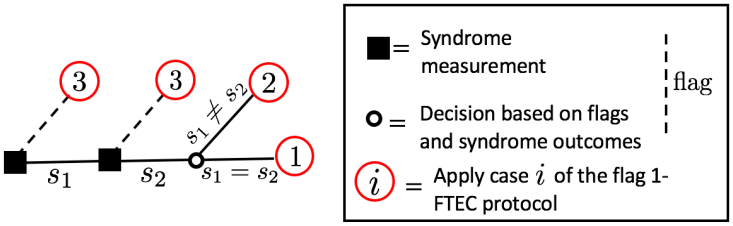


Figure 3.2: Tree diagram illustrating the possible paths of the Flag 1-FTEC Protocol. Numbers enclosed in red circles at the end of the edges indicate which step to implement in the Flag 1-FTEC Protocol. A dashed line is followed when any of the 1-flag circuits $C(g_i)$ flags. Solid squares indicate a syndrome measurement using 1-flag circuits whereas rings indicate a decision based on syndrome outcomes. Note that the syndrome measurement is repeated at most three times.

Consider a stabilizer code $\mathcal{S} = \langle g_1, g_2, \cdots, g_r \rangle$ *and* 1*-flag circuits* $\{C(g_1), C(g_2), \cdots, C(g_r)\}$. *For every generator* g_i*, all pairs of elements* $E, E' \in \mathcal{E}(g_i)$ *satisfy* $s(E) \neq s(E')$ *or* $E \sim E'$.

In other words, we require that any two errors that arise when a circuit $C(g_i)$ flags due to a single fault must be either distinguishable or logically equivalent. For the following protocol to satisfy the FTEC conditions in Def. 5, one can assume there is at most 1 fault. If the Flag 1-FTEC condition is satisfied, the protocol is implemented as follows:

Flag 1-FTEC Protocol:
Repeat the syndrome measurement using flag circuits until one of the following is satisfied:

1. If the syndrome s is repeated twice in a row and there were no flags, apply the correction $E_{\min}(s)$.

2. If there were no flags and the syndromes s_1 and s_2 from two consecutive rounds differ, repeat the syndrome measurement using non-flag circuits yielding syndrome s. Apply the correction $E_{\min}(s)$.

3. If a circuit $C(g_i)$ flags, stop and repeat the syndrome measurement using non-flag circuits yielding syndrome s. If there is an element $E \in \mathcal{E}(g_i)$ which satisfies $s(E) = s$, then apply E, otherwise apply $E_{\min}(s)$.

A tree diagram for the flag 1-FTEC Protocol is illustrated in Fig. 3.2. We now outline

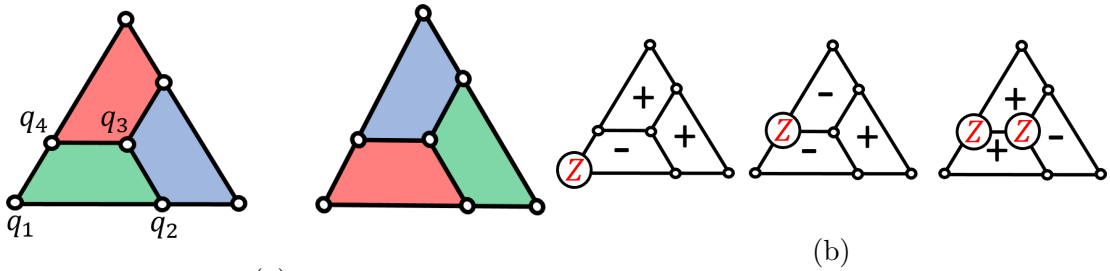


Figure 3.3: (a) A representation of the Steane code where each circle is a qubit, and there is an X- and a Z-type stabilizer generator for each face. Stabilizer circuits are specified from that in Fig. 3.1(a) after rotating the lattice such that the relevant face is on the bottom left. (b) For $g = Z_{q_1}Z_{q_2}Z_{q_3}Z_{q_4}$, the flag error set is $\mathcal{E}(g) = \{I, Z_{q_3}Z_{q_4}, X_{q_2}Z_{q_3}Z_{q_4}, Z_{q_1}X_{q_2}, Z_{q_4}, X_{q_3}Z_{q_4}, X_{q_3}Z_{q_3}Z_{q_4}\}$ which contains all errors arising from a single fault that causes the stabilizer measurement circuit $C(g)$ to flag. Since the Steane code is a CSS code, the X component of an error will be corrected independently allowing us to consider the Z-part of the flag error set $\mathcal{E}_Z(g) = \{I, Z_{q_1}, Z_{q_4}, Z_{q_3}Z_{q_4}\}$. As required, the elements of $\mathcal{E}_Z(g)$ all have distinct syndromes (with satisfied stabilizers represented by a plus).

the proof that the flag 1-FTEC protocol satisfies the fault-tolerance criteria of Def. 5 (a more rigorous proof of the general case is presented in Appendix. B.1). To show that Flag 1-FTEC Protocol satisfies the criteria of Def. 5, we can assume there is at most one fault during the protocol. If a single fault occurs in either the first or second round leading to a flag, repeating the syndrome measurement will correctly diagnose the error. If there are no flags and a fault occurs which causes the syndromes in the first two rounds to change, then the syndrome during the third round will correctly diagnose the error. There could also be a fault during either the first or second round that goes undetected. But since there were no flags it cannot spread to an error of weight-2. In this case applying a minimum weight correction based on the measured syndrome of the second round will guarantee that the output codeword differs from a valid codeword by an error of weight at most one. Note that the above argument applies irrespective of any errors on the input state, hence the second criteria of Def. 5 is satisfied. It is worth pointing out that up to three repetitions are required in order to guarantee that the second criteria of Def. 5 is satisfied (unless the code has the property that all states are at most a weight-one error away from a valid codeword, as in Ref. [57]).

The Steane code is an example which satisfies the Flag 1-FTEC condition with a simple choice of circuits. To verify this, the representation of the Steane code given in Fig. 3.3b

is useful. There is an X- and a Z-type stabilizer generator supported on the four qubits of each of the three faces. First let us specify all six stabilizer measurement circuits. The circuit that measures $Z_{q_1}Z_{q_2}Z_{q_3}Z_{q_4}$ is specified by taking qubits q_1, q_2, q_3, and q_4 to be the four data qubits in descending order in the 1-flag circuit in Fig. 3.1b. The other two Z-stabilizer measurement circuits are obtained by first rotating Fig. 3.3b by 120° and 240° and then using Fig. 3.1b. The X-stabilizer circuit for each face is the same as the Z-stabilizer circuit for that face, replacing CNOT gates acting on data qubits by XNOT gates. The Z component of the flag error set of the circuit in Fig. 3.1b is $\mathcal{E}_Z(Z_{q_1}Z_{q_2}Z_{q_3}Z_{q_4}) = \{I, Z_{q_1}, Z_{q_4}, Z_{q_3}Z_{q_4}\}$. As can be seen from Fig. 3.3b, each of these has a distinct syndrome, thus the measurement circuit for $Z_{q_1}Z_{q_2}Z_{q_3}Z_{q_4}$ satisfies the flag 1-FTEC condition, as do the remaining five measurement circuits by symmetry.

3.2.2 Flag 2-FTEC with distance-5 codes

Before explicitly describing the conditions and protocol, we discuss some of the complications that arise for codes with $d > 3$.

For distance-5 codes, we must ensure that if two faults occur during the error correction protocol, the output state will differ from a codeword by an error of at most weight-two. For instance, if two faults occur in a circuit for measuring a stabilizer of weight greater than four, the resulting error E on the data should satisfy $\mathrm{wt}(E) \leq 2$ unless there is a flag. In other words, all stabilizer generators should be measured using 2-flag circuits.

In another case, two faults could occur during the measurement of *different* stabilizer generators g_i and g_j. If two bad locations fail and are both flagged, and assuming there are no more faults, the measured syndrome will correspond to the product of the error caused in each circuit (which could have weight greater than two). Consequently, one should modify Def. 10 of the flag error set to include these types of errors. One then decodes based on the pair of errors that resulted in the measured syndrome, provided logically inequivalent errors have distinct syndromes.

Before stating the protocol, we extend some of the definitions given above.

Consider a stabilizer code $\mathcal{S} = \langle g_1, g_2, \cdots, g_r \rangle$ and t-flag circuits $C(g_i)$ for measuring the generator g_i.

Definition 12. *Flag error set*

Let $\mathcal{E}_m(g_{i_1}, \cdots, g_{i_k})$ be the set of all errors caused by precisely m faults spread amongst the circuits $C(g_{i_1}), C(g_{i_2}), \cdots, C(g_{i_k})$ which all flagged.

Note that there could be more than one fault in a single circuit $C(g_{i_k})$. Examples of flag error sets are given in Tab. 3.1 where only contributions from Z errors are included (since the considered code is a CSS code). We also define a general t-fault correction set:

$$\tilde{E}_t^m(g_{i_1}, \cdots, g_{i_k}, s) = \begin{cases} \{E \in \mathcal{E}_m(g_{i_1}, \cdots, g_{i_k}) \times \mathcal{E}_{t-m} \\ \text{ such that } s(E) = s\} \\ \{E_{\min}(s)\} \text{ if above set empty.} \end{cases} \tag{3.4}$$

By $E \in \mathcal{E}_m(g_{i_1}, \cdots, g_{i_k}) \times \mathcal{E}_{t-m}$, we are considering the set consisting of products between errors caused by k flags and any error of weight $t - m$.

As will be seen below, the correction set will form a critical part of the protocol by specifying the correction applied based on the measured syndrome and flag outcomes over multiple syndrome measurement rounds. In the case where k t-flag circuits flagged caused by $k \leq m \leq t$ faults, the correction applied to the data block will correspond to an element of $\mathcal{E}_m(g_{i_1}, \cdots, g_{i_k}) \times \mathcal{E}_{t-m}$ if the measured syndrome corresponds to an element in this set (there could also be $t - m$ faults which did not give rise to a flag). However in practice, there could be more than t faults and so the measured syndrome may not be consistent with any element of the set $\mathcal{E}_m(g_{i_1}, \cdots, g_{i_k}) \times \mathcal{E}_{t-m}$. In this case, and for the error correction protocol to satisfy the second criteria of Def. 5, the correction will correspond to $E_{\min}(s)$. These features are all included in the set $\tilde{E}_t^m(g_{i_1}, \cdots, g_{i_k}, s)$.

Definition 13. ***Flag* 2*-FTEC condition:***

Consider a stabilizer code $\mathcal{S} = \langle g_1, g_2, \cdots, g_r \rangle$ *and* 2*-flag circuits* $\{C(g_1), C(g_2), \cdots, C(g_r)\}$. *For any choice of generators* $\{g_i, g_j\}$*:*

1. $E, E' \in \mathcal{E}_2(g_i, g_j) \Rightarrow s(E) \neq s(E')$ *or* $E \sim E'$,
2. $E, E' \in \mathcal{E}_2(g_i) \cup (\mathcal{E}_1(g_i) \times \mathcal{E}_1) \Rightarrow s(E) \neq s(E')$ *or* $E \sim E'$.

In order to state the protocol, we define an update rule given a sequence of syndrome measurements using t-flag circuits for the counters[2] n_{diff} and n_{same} as follows:

[2] n_{diff} tracks the minimum number of faults that could have caused the observed syndrome outcomes. For example, if the sequence s_1, s_2, s_3 was measured, n_{diff} would increase by one since a single measurement fault could give rise to the given sequence (for example, this could be caused by a single CNOT failure which resulted in a data qubit and measurement error). However for the sequence s_1, s_2, s_1, s_2, n_{diff} would increase by two.

Flag 2-FTEC protocol – update rules:
Given a sequence of consecutive syndrome measurement outcomes s_k and s_{k+1}:

1. If n_{diff} didn't increase in the previous round, and $s_k \neq s_{k+1}$, increase n_{diff} by one.
2. If a flag occurs, reset n_{same} to zero.
3. If $s_k = s_{k+1}$, increase n_{same} by one.

For the following protocol to satisfy Def. 5, one can assume there are at most 2 faults. If the Flag 2-FTEC condition is satisfied, the protocol is implemented as follows:

Flag 2-FTEC protocol – corrections:
Set $n_{\text{diff}} = 0$ and $n_{\text{same}} = 0$.
Repeat the syndrome measurement using flag circuits until one of the following is satisfied:

1. The same syndrome s is repeated $3 - n_{\text{diff}}$ times in a row and there were no flags, apply the correction $E_{\min}(s)$.
2. There were no flags and $n_{\text{diff}} = 2$. Repeat the syndrome measurement using non-flag circuits yielding syndrome s. Apply the correction $E_{\min}(s)$.
3. Some set of two circuits $C(g_i)$ and $C(g_j)$ have flagged. Repeat the syndrome measurement using non-flag circuits yielding syndrome s. Apply any correction from the set $\tilde{E}_2^2(g_i, g_j, s)$.
4. Any circuit $C(g_i)$ has flagged and $n_{\text{diff}} = 1$. Repeat the syndrome measurement using non-flag circuits yielding syndrome s. Apply any correction from the set $\tilde{E}_2^1(g_i, s)$.
5. Any circuit $C(g_i)$ has flagged and $n_{\text{diff}} = 0$ and $n_{\text{same}} = 1$. Use the measured syndrome s from the last round. Apply any correction from the set $\tilde{E}_2^1(g_i, s) \cup \tilde{E}_2^2(g_i, s)$.

Note that when computing the update rules, if a flag occurs during the j'th round of syndrome measurements, the syndrome is not recorded for that round since all stabilizers must be measured. Thus when computing n_{diff} and n_{same} using consecutive syndromes s_k and s_{k+1}, we are assuming that no flags occurred during rounds k and $k+1$.

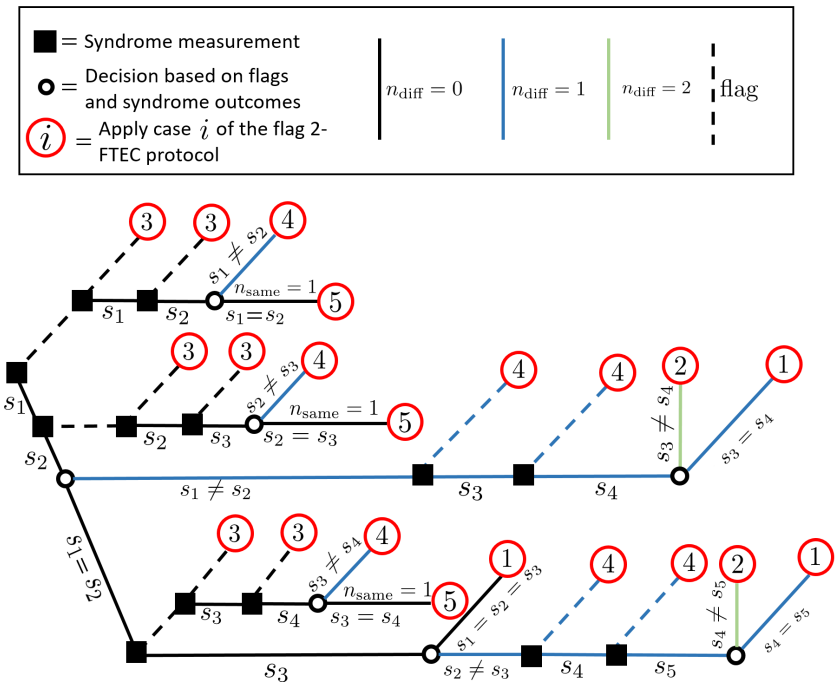


Figure 3.4: Tree diagram for the Flag 2-FTEC protocol. Numbers encircled in red at the end of the edges indicate which step to implement in the Flag 2-FTEC Protocol. A dashed line is followed when any of the 2-flag circuits $C(g_i)$ flags. Solid squares indicate a syndrome measurement using 2-flag circuits whereas rings indicate a decision based on syndrome outcomes. Edges with different colors indicate the current value of n_{diff} in the protocol. Note that the protocol is repeated at most 6 times.

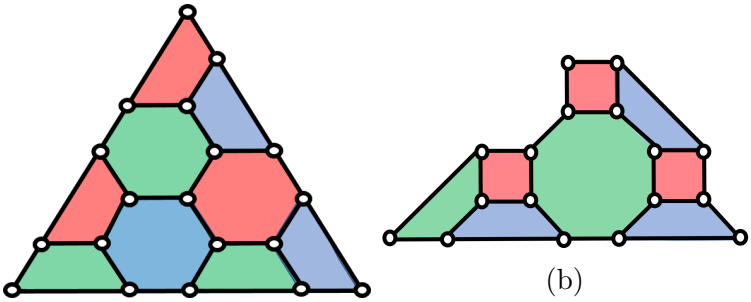


Figure 3.5: Graphical representation of (a) the 19-qubit 2D color code and (b) the 17-qubit 2D color code. The X and Z stabilizers of the code are symmetric, given by the vertices of each plaquette. Both codes have distance-5.

In each case of the protocol, the correction sets correspond to those data errors which could arise from up to two faults which are consistent with the conditions of the case. As the elements are logically equivalent (by Eq. 3.4 and Def. 13), which element is applied is unimportant.

The general protocol for codes of arbitrary distance is given in Sec. 3.4.

3.3 Examples of flag 2-FTEC applied to $d = 5$ codes

In this section we give examples of the flag 2-FTEC protocol applied to the 2-dimensional $[\![19, 1, 5]\!]$ and $[\![17, 1, 5]\!]$ color codes, (see Fig. 3.5a and Fig. 3.5b). We first find 2-flag circuits for all generators (weight-4 and -6 for the 19-qubit code and weight-4 and -8 for the 17-qubit code). We also show that the flag 2-FTEC condition is satisfied for both codes.

For a 2-flag circuit, two faults leading to an error of weight greater or equal to 3 (up to multiplication by the stabilizer) must always cause at least one of the flag qubits to flag. As shown in Sec. 3.5, a 2-flag circuit satisfying these properties can always be constructed using at most four flag qubits. We show 2-flag circuits for measuring weight six and eight generators in Fig. 3.6.

In Sec. 3.4.2, it will be shown that the family of color codes with a hexagonal lattice satisfy a sufficient condition which guarantees that the flag 2-FTEC condition is satisfied. However, there are codes that do not satisfy the sufficient condition but which nonetheless satisfy the 2-Flag FTEC condition. For the 19-qubit and 17-qubit color codes, we verified

Weight-4 measurement		Weight-6 measurement	
1-fault	2-faults	1-fault	2-faults
I,Z_1	I,Z_1,Z_2	I,Z_1,Z_6	I,Z_1,Z_2
Z_4	Z_3,Z_4	Z_1Z_2	Z_3, Z_4,Z_5,Z_6
Z_3Z_4	Z_1Z_2	Z_5Z_6	Z_1Z_2,Z_1Z_3
	Z_1Z_4	$Z_4Z_5Z_6$	Z_1Z_4,Z_1Z_5
	Z_2Z_4		Z_1Z_6,Z_2Z_3
			Z_2Z_6,Z_3Z_4
			Z_3Z_6,Z_4Z_5
			Z_4Z_6,Z_5Z_6
			$Z_1Z_2Z_3$,$Z_1Z_5Z_6$
			$Z_2Z_5Z_6$,$Z_3Z_4Z_5$
			$Z_3Z_4Z_6$,$Z_3Z_5Z_6$
			$Z_4Z_5Z_6$

Table 3.1: Z part of the flag error set of Def. 12 for flag circuits used to measure the stabilizers $g_1 = Z_1Z_2Z_3Z_4$ and $g_3 = Z_1Z_2Z_3Z_4Z_5Z_6$ (we removed errors equivalent up to the stabilizer being measured).

that the flag 2-FTEC condition was satisfied by enumerating all errors one would have for a generic code. In particular, in the case where the 2-flag circuits $C(g_i)$ and $C(g_j)$ flag, the resulting errors belonging to the set $\mathcal{E}_2(g_i, g_j)$ must be logically equivalent or have distinct syndromes (which we verified to be true). If a single circuit $C(g_i)$ flags, there could either have been two faults in the circuit or a single fault along with another error that did not cause a flag. If the same syndrome is measured twice in a row after a flag, then errors in the set $\mathcal{E}_2(g_i) \cup (\mathcal{E}_1(g_i) \times \mathcal{E}_1)$ must be logically equivalent or have distinct syndromes (which we verified). If there is a flag but two different syndromes are measured in a row, errors belonging to the set $\mathcal{E}_1(g_i) \times \mathcal{E}_1$ must be logically equivalent or have distinct syndromes (as was already checked). The flag error sets (see Def. 12) for the 19-qubit code can be obtained using the Pauli's shown in Tab. 3.1.

Given that the flag 2-FTEC condition is satisfied, the flag 2-FTEC protocol can be implemented following the steps of Sec. 3.2.2 and the tree diagram illustrated in Sec. 3.4.

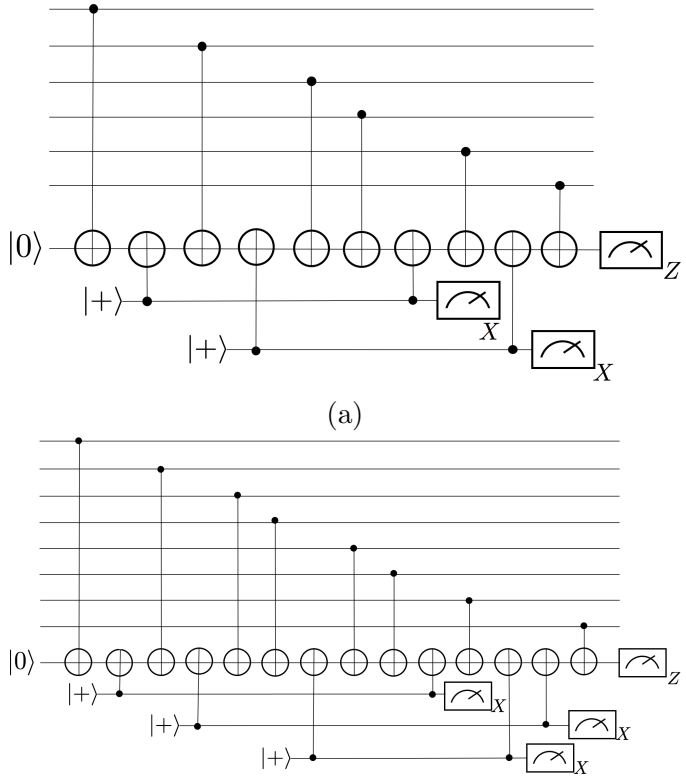


Figure 3.6: Illustration of 2-flag circuits for measuring (a) $Z^{\otimes 6}$ requiring only two flag qubits and (b) $Z^{\otimes 8}$ requiring only three flag qubits. Flag qubits are prepared in the $|+\rangle$ state, and measurement qubits in the $|0\rangle$ state.

3.4 Flag error correction protocol for arbitrary distance codes

In this section we first provide the general flag t-FTEC protocol. We then give a sufficient condition for stabilizer codes that allow us to easily prove that flag FTEC can be applied to a number of infinite code families. We show that the families of surface codes, hexagonal lattice color codes and quantum Reed-Muller codes satisfy the sufficient condition. General t-flag circuit constructions which are applicable to the code families described in this section will be given in Sec. 3.5.

3.4.1 Conditions and protocol

In what follows we generalize the fault-tolerant error correction protocol presented in Sec. 3.2.2 to stabilizer codes of arbitrary distance.

Definition 14. ***Flag t-FTEC condition:***

Consider a stabilizer code $\mathcal{S} = \langle g_1, g_2, \cdots, g_r \rangle$ and t-flag circuits $\{C(g_1), C(g_2), \cdots, C(g_r)\}$. For any set of m stabilizer generators $\{g_{i_1}, \cdots, g_{i_m}\}$ such that $1 \leq m \leq t$, every pair of elements $E, E' \in \bigcup_{j=0}^{t-m} \mathcal{E}_{t-j}(g_{i_1}, \cdots, g_{i_m}) \times \mathcal{E}_j$ either satisfy $s(E) \neq s(E')$ or $E \sim E'$.

The above conditions ensure that if there are at most $t = \lfloor (d-1)/2 \rfloor$ faults, the protocol described below will satisfy the fault-tolerance conditions of Def. 5.

In order to state the protocol, we define an update rule given a sequence of syndrome measurements using t-flag circuits for the counters n_{diff} and n_{same} as follows (see also Sec. 3.2.2 and the associated footnote):

Flag t-FTEC protocol – update rules:
Given a sequence of consecutive syndrome measurement outcomes s_k and s_{k+1}:

1. If n_{diff} didn't increase in the previous round, and $s_k \neq s_{k+1}$, increase n_{diff} by one.
2. If a flag occurs, reset n_{same} to zero.
3. If $s_k = s_{k+1}$, increase n_{same} by one.

Flag t-FTEC protocol – corrections:
Set $n_{\text{diff}} = 0$ and $n_{\text{same}} = 0$.
Repeat the syndrome measurement using flag circuits until one of the following is satisfied:

1. The same syndrome s is repeated $t - n_{\text{diff}} + 1$ times in a row and there are no flags, apply the correction $E_{\min}(s)$.

2. There were no flags and $n_{\text{diff}} = t$. Repeat the syndrome measurement using non-flag circuits yielding the syndrome s. Apply the correction $E_{\min}(s)$.

3. Some set of t circuits $\{C(g_{i_1}), \cdots, C(g_{i_t})\}$ have flagged. Repeat the syndrome measurement using non-flag circuits yielding syndrome s. Apply any correction from the set $\tilde{E}_t^t(g_{i_1}, \cdots, g_{i_t}, s)$.

4. Some set of m circuits $\{C(g_{i_1}), \cdots, C(g_{i_m})\}$ have flagged with $1 \leq m < t$ and $n_{\text{diff}} = t - m$. Repeat the syndrome measurement using non-flag circuits yielding syndrome s. Apply any correction from the set $\tilde{E}_t^m(g_{i_1}, \cdots, g_{i_m}, s)$.

5. Some set of m circuits $\{C(g_{i_1}), \cdots, C(g_{i_m})\}$ have flagged with $1 \leq m < t$; $n_{\text{diff}} < t - m$ and $n_{\text{same}} = t - m - n_{\text{diff}} + 1$. Use the syndrome s obtained during the last round and apply any correction from the set $\bigcup_{j=0}^{t-m-n_{\text{diff}}} \tilde{E}_t^{t-j-n_{\text{diff}}}(g_{i_1}, \cdots, g_{i_m}, s)$.

In each case of the protocol, the correction sets correspond to those data errors which could arise from up to t faults which are consistent with the conditions of the case. As the elements are logically equivalent (by Eq. 3.4 and Def. 14), which element is applied is unimportant.

For the protocol to satisfy the fault-tolerance criteria, the syndrome measurement needs to be repeated a minimum of $t + 1$ times. In the scenario where the most syndrome measurement rounds are performed, t identical syndromes are obtained before a fault causes the $t+1$'th syndrome to change (in which case n_{diff} would increase by one). Afterwords, one measures the same syndrome $t - 1$ times in a row until another fault causes the syndrome to change. This continues until all of the t possible faults have been exhausted. At this stage, $n_{\text{diff}} = t$ so an extra syndrome measurement round will be performed using non-flag circuits. Thus the maximum number of syndrome measurement rounds $n_{\max}$ is given by

$$n_{\max} = \sum_{j=0}^{t-1}(t - j) + t + 1 = \frac{1}{2}(t^2 + 3t + 2). \tag{3.5}$$

Recall that in Sec. 1.2.1 when discussing Shor-EC, we required $(t+1)^2$ syndrome measure-

ment repetitions. However, our scheme presented in this section requires fewer syndrome measurement repetitions and does not require the preparation and verification of a w-qubit cat state when measuring a stabilizer of weight-w. [3]

For codes that satisfy the flag t-FTEC condition, we also show in Appendix. B.2 how to fault-tolerantly prepare and measure logical states using the flag t-FTEC protocol.

3.4.2 Sufficient condition and satisfying code families

The general flag t-FTEC condition can be difficult to verify for a given code since it depends on precisely which t-flag circuits are used. A sufficient (but not necessary) condition that implies the flag t-FTEC condition is as follows:

Sufficient flag t-FTEC condition:

Given a stabilizer code with distance $d > 1$, and $\mathcal{S} = \langle g_1, g_2, \cdots, g_r \rangle$, we require that for all $v = 0, 1, \ldots t$, all choices Q_{t-v} of $2(t-v)$ qubits, and all subsets of v stabilizer generators $\{g_{i_1}, \ldots, g_{i_v}\} \subset \{g_1, \cdots, g_r\}$, there is no logical operator $l \in N(\mathcal{S}) \setminus \mathcal{S}$ such that

$$\text{supp}(l) \subset \text{supp}(g_{i_1}) \cup \cdots \cup \text{supp}(g_{i_v}) \cup Q_{t-v}, \tag{3.6}$$

where $N(\mathcal{S})$ is the normalizer of the stabilizer group.

If this condition holds, then the flag t-FTEC condition is implied for any choice of t-flag circuits $\{C(g_1), C(g_2), \cdots, C(g_r)\}$.

To prove this, we must show that it implies that none of the sets appearing in the t-FTEC condition contain elements that differ by a logical operator. Consider the set $\bigcup_{j=0}^{t-m} \mathcal{E}_{t-j}(g_{i_1}, \cdots, g_{i_m}) \times \mathcal{E}_j$ for some set of m stabilizer generators $\{g_{i_1}, \cdots, g_{i_m}\}$ with $1 \leq m \leq t$. An error E from this set will have support in the union of the support of the m stabilizer generators $\{g_{i_1}, \cdots, g_{i_m}\}$, along with up to $t-m$ other single qubits. Another error E' from this set will have support in the union of support of the same m stabilizer generators $\{g_{i_1}, \cdots, g_{i_m}\}$, along with up to $t-m$ other *potentially different* single qubits. If the sufficient condition holds, then $\text{supp}(EE')$ cannot contain a logical operator.

The sufficient flag t-FTEC condition is straightforward to verify for a number of code families with a lot of structure in their stabilizer generators and logical operators. We briefly provide a few examples.

[3] One could also define update rules analogous to those for n_{diff} and n_{same} when implementing Shor-EC which would only require at most $\frac{1}{2}(t^2 + 3t + 2)$ syndrome measurement repetitions as in the flag t-FTEC protocol.

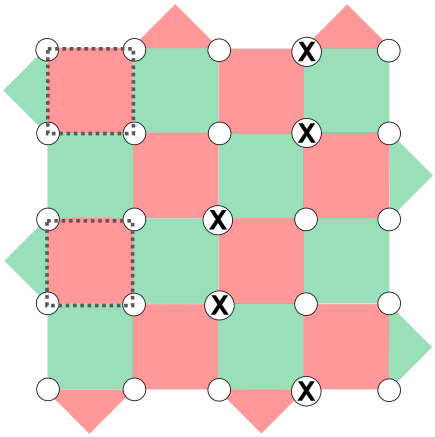


Figure 3.7: The $d = 5$ rotated surface code. Qubits are represented by white circles, and X and Z stabilizer generators are represented by red and green faces. As in the example, any logical X operator has X operators acting on at least five qubits, with at least one in each row of the lattice, involving an even number in any green face. In this case, no two stabilizer generators can have qubits in five rows, and therefore cannot contain an X type logical operator. The argument is analogous for logical Z operators.

Surface codes flag t-FTEC:

The rotated surface code [4, 45, 61, 62] family $[\![d^2, 1, d]\!]$ for all odd $d = 2t + 1$ (see Fig. 3.7) satisfies the flag t-FTEC condition using any 4-flag circuits.

Firstly, by performing an exhaustive search, we verified that the circuit of Fig. 3.1b is a 4-flag circuit.

As a CSS code, we can restrict our attention to purely X-type and Z-type logical operators. An X type logical operator must have at least one qubit in each of the $2t + 1$ rows of the lattice shown. However, each stabilizer only contains qubits in two different rows. Therefore, with v stabilizer generators, at most $2v$ of the rows could have support. With an additional $2(t - v)$ qubits, at most $2t$ rows can be covered, which is fewer than the number of rows, and therefore no logical X operator is supported on the union of the support of v stabilizers and $2(t-v)$ qubits. An analogous argument holds for Z-type logical operators, therefore the sufficient t-FTEC condition is satisfied.

Color codes flag t-FTEC:

Here we show that any distance $d = (2t + 1)$ self-dual CSS code with at most weight-6 stabilizer generators satisfies the flag t-FTEC condition using any 6-flag circuits (see

Fig. 3.9a for an example). Examples include the hexagonal color code [5] family $[\![(3d^2+1)/4, 1, d]\!]$ (see Fig. 3.5a).

As a self-dual CSS code, X and Z type stabilizer generators are identically supported and we can consider a pure X-type logical operator without loss of generality.

Consider an X type logical operator l such that

$$\text{supp}(l) \subset \text{supp}(g_{i_1}) \cup \cdots \cup \text{supp}(g_{i_v}) \cup Q_{t-v}, \tag{3.7}$$

for some set of v stabilizer generators $\{g_{i_1}, \ldots, g_{i_v}\} \subset \{g_1, \cdots, g_r\}$ along with $2(t-v)$ other qubits Q_{t-v}. Restricted to the support of any of the v stabilizers g_i, $l|_{g_i}$ must have weight 0, 2, 4, or 6 (otherwise it would anti-commute with the corresponding Z type stabilizer). If the restricted weight is 4 or 6, we can produce an equivalent lower weight logical operator $l' = g_i l$, which still satisfies Eq. 3.7. Repeating this procedure until the weight of the logical operator can no longer be reduced yields a logical operator l_{min} which has weight either 0 or 2 when restricted to the support of any of the v stabilizer generators. The total weight of l_{min} is then at most $2v + 2(t-v) = 2t$, which is less than the distance of the code, giving a contradiction which therefore implies that l could not have been a logical operator. An analogous argument holds for Z-type logical operators, therefore the sufficient t-FTEC condition is satisfied.

This proof can be easily extended to show that any distance $d = (2t+1)$ self-dual CSS code with at most weight-$2v$ stabilizer generators for some integer v satisfies the flag t'-FTEC condition using any $(v-1)$-flag circuits, where $t' = t/\lfloor v/2 \rfloor$.

Quantum Reed-Muller codes flag 1-FTEC:

The $[\![n = 2^m - 1, k = 1, d = 3]\!]$ quantum Reed-Muller code family for every integer $m \geq 3$ satisfies the flag 1-FTEC condition using any 1-flag circuits for the standard choice of generators.

We use the following facts about the Quantum Reed-Muller code family (see Appendix. B.4 and Ref. [63] for proofs of these facts): (1) The code is CSS, allowing us to restrict to pure X type and pure Z type logical operators, (2) all pure X or Z type logical operators have odd support, (3) every X-type stabilizer generator has the same support as some Z-type stabilizer generator, and (4) every Z-type stabilizer generator is contained within the support of an X type generator.

We only need to prove the sufficient condition for $v = 0, 1$ in this case. For $v = 0$, no two qubits can support a logical operator, as any logical operator has weight at least three. For $v = 1$, assume the support of an X-type stabilizer generator contains a logical operator l. That logical operator l cannot be Z type or it would anti-commute with the X-stabilizer

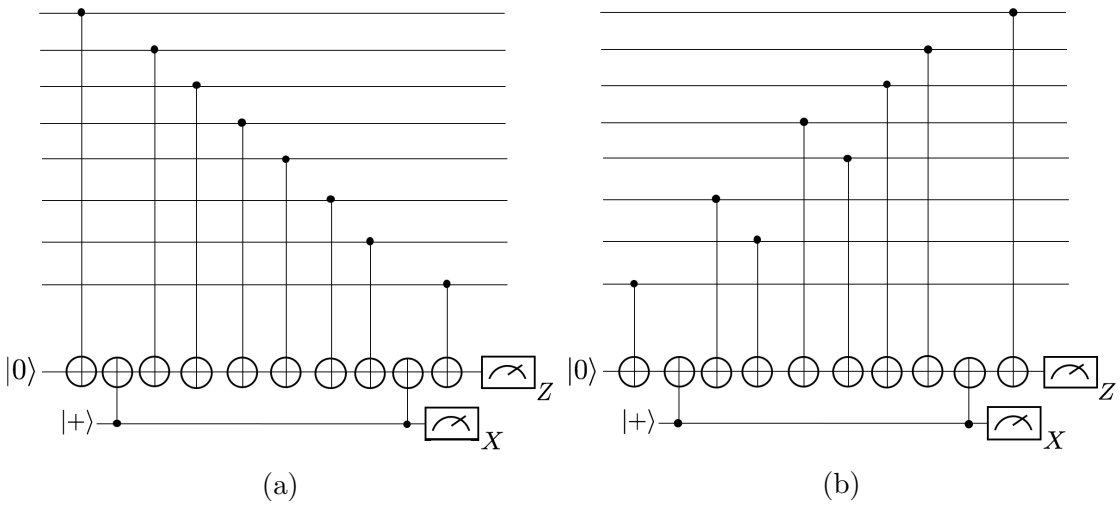


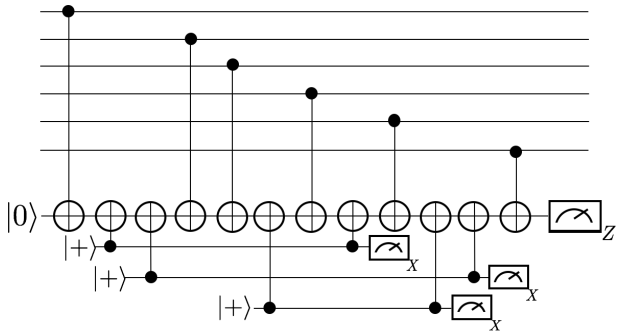
Figure 3.8: (a) A 1-flag circuit for measuring the stabilizer $Z_8Z_9Z_{10}Z_{11}Z_{12}Z_{13}Z_{14}Z_{15}$ of the $[\![15, 7, 3]\!]$ Hamming code. However a single fault on the fourth or fifth CNOT can lead to the error $Z_{12}Z_{13}Z_{14}Z_{15}$ on the data which is a logical fault. With the CNOT gates permuted as shown in (b), the $[\![15, 7, 3]\!]$ satisfies the general flag 1-FTEC condition.

due to its odd support. However, by fact (3), there is a Z type stabilizer with the same support as the X type stabilizer, therefore implying l cannot be X type either. Therefore, by contradiction we conclude that no logical operator can be contained in the support of an X stabilizer generator. Since every other stabilizer generator is contained within the support of an X-type stabilizer generator, a logical operator cannot be contained in the support of any stabilizer generator.

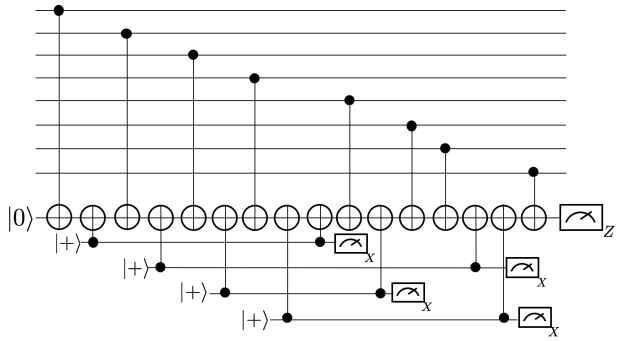
Note that the Hamming code family has a stabilizer group which is a proper subgroup of that of the quantum Reed-Muller codes described here. The X-type generators of each Hamming code are the same as for a quantum Reed-Muller code, and the Hamming codes are self-dual CSS codes. It is clear that the sufficient condition cannot be applied to the Hamming code since it has even-weight Z-type logical operators (which are stabilizers for the quantum Reed-Muller code) supported within the support of some stabilizer generators.

Codes which satisfy the flag t-FTEC condition but not the sufficient flag t-FTEC condition:

Note that there are codes which satisfy the general flag t-FTEC condition but not the sufficient condition presented in this section. An example of such a code is the $[\![5, 1, 3]\!]$ code (see Tab. 2.1 for the codes stabilizer generators and logical operators). Another example includes the Hamming codes as was explained in the discussion on quantum Reed-Muller



(a)



(b)

Figure 3.9: (a) Illustration of a w-flag circuit for measuring the operator $Z^{\otimes w}$ where $w = 6$ using the smallest number of flag qubits. (b) Illustration of a 3-flag circuit for measuring $Z^{\otimes 8}$ using the smallest number of flag qubits.

codes. For instance, consider the $[\![15, 7, 3]\!]$ Hamming code. Using the 1-flag circuit shown in Fig. 3.8a, the $[\![15, 7, 3]\!]$ will not satisfy the general flag 1-FTEC condition since a single fault can lead to a logical error on the data. As was shown in Ref. [57], by permuting the CNOT gates resulting in the circuit illustrated in Fig. 3.8b, the flag 1-FTEC condition is satisfied. A larger family of codes which satisfy the general flag t-FTEC condition but not the sufficient condition was shown in Ref. [64].

3.5 Circuits

In Sec. 3.4.2 we showed that the family of surface codes, color codes with a hexagonal lattice and quantum Reed-Muller codes satisfied a sufficient condition allowing them to

be used in the flag t-FTEC protocol. Along with the general 1-flag circuit construction of Fig. 3.10a, the 6-flag circuit for measuring $Z^{\otimes 6}$ of Fig. 3.9a can be used as t-flag circuits for all of the codes in Sec. 3.4.2. Note that the circuit in Fig. 3.1b (which is a special case of Fig. 3.10a when $w = 4$) is a 4-flag circuit which is used for measuring $Z^{\otimes 4}$.

Before describing general 1- and 2-flag circuit constructions, we give the following two definitions which we will frequently use: Any CNOT that couples a data qubit to the measurement qubit will be referred to as CNOT_{dm} and any CNOT coupling a measurement qubit to a flag qubit will be referred to as CNOT_{fm}. In both cases the target qubit will always be the measurement qubit.

1- and 2-flag circuits for weight w stabilizer measurements:

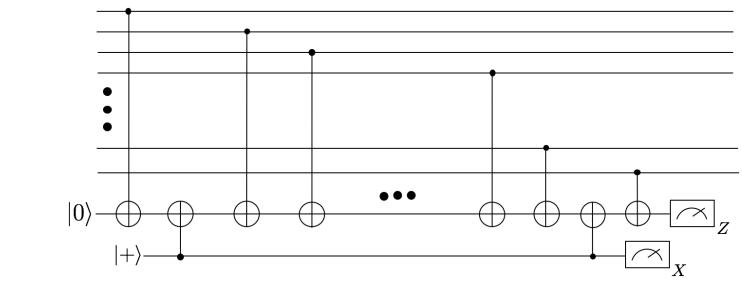
We provide 1- and 2-flag circuit constructions for measuring a weight-w stabilizer. The 1-flag circuit requires a single flag qubit, and the 2-flag circuit requires at most four flag qubits.

Without loss of generality, in proving that the circuit constructions described below are 1- and 2-flag circuits, we can assume that all faults occurred on CNOT gates. This is because any set of v faults (including those at idle, preparation or measurement locations) will have the same output Pauli operator and flag measurement results as some set of at most v faults on CNOT gates (since every qubit is involved in at least one CNOT).

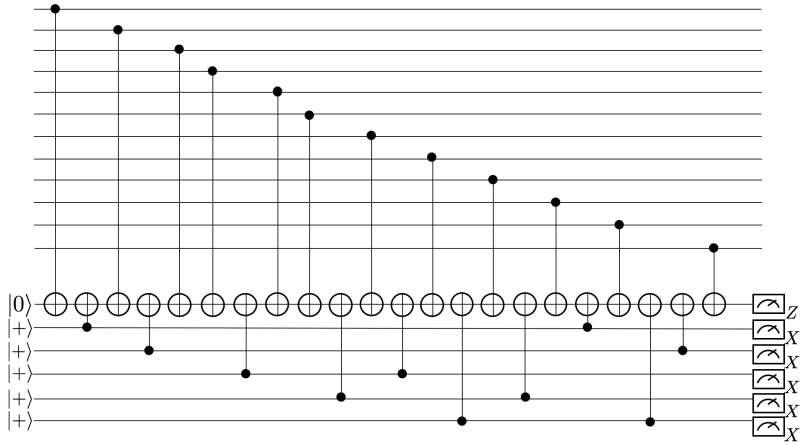
As was shown in Ref. [57], Fig. 3.10a illustrates a general 1-flag circuit construction for measuring the stabilizer $Z^{\otimes w}$ which requires only two CNOT_{fm} gates. To see that the first construction is a 1-flag circuit, note that an IZ error occurring on any CNOT will give rise to a flag unless it occurs on the first or last CNOT_{dm} gates or the last CNOT_{fm} gate. However, such a fault on any of these three gates can give rise to an error of weight at most one (after multiplying by the stabilizer $Z^{\otimes w}$). One can also verify that if there are no faults, the circuit in Fig. 3.10a implements a projective measurement of $Z^{\otimes w}$ without flagging. Following the approach in Ref. [65], one simply needs to check that the circuit preserves the stabilizer group generated by $Z^{\otimes w}$ and X on each ancilla prepared in the $|+\rangle$ state and Z on each ancilla prepared in the $|0\rangle$ state. By using pairs of CNOT_{fm} gates, this construction satisfies the requirement.

We now give a general 2-flag circuit construction for measuring $Z^{\otimes w}$ for arbitrary w (see Fig. 3.10b for an example). The circuit consists of pairs of CNOT_{fm} gates each connected to a different flag qubit prepared in the $|+\rangle$ state and measured in the X basis. The general 2-flag circuit construction involves the following placement of $w/2 - 1$ pairs of CNOT_{fm} gates:

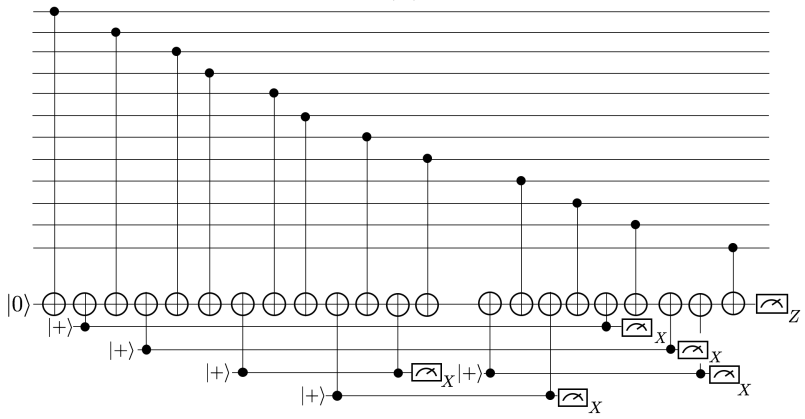
1. Place a CNOT_{fm} pair between the first and second last CNOT_{dm} gates.



(a)



(b)



(c)

Figure 3.10: (a) General 1-flag circuit for measuring the stabilizer $Z^{\otimes w}$. (b) Example of a 2-flag circuit for measuring $Z^{\otimes 12}$ using our general 2-flag circuit construction. (c) An equivalent circuit using fewer flag qubits by reusing a measured flag qubit and reinitializing it in the $|+\rangle$ state for use in another pair of $\mathrm{CNOT}_{\mathrm{fm}}$ gates.

2. Place a CNOT_{fm} pair between the second and last CNOT_{dm} gates.

3. After the second CNOT_{fm} gate, place the first CNOT_{fm} gate of the remaining pairs after every two CNOT_{dm} gates. The second CNOT_{fm} gate of a pair is placed after every three CNOT_{dm} gates.

As shown in Fig. 3.10c, it is possible to reuse some flag qubits to measure multiple pairs of CNOT_{fm} gates at the cost of introducing extra time steps into the circuit. For this reason, at most four flag qubits will be needed, however, if $w \leq 8$, then $w/2 - 1$ flag qubits are sufficient.

We now show that the above construction satisfies the requirements of a 2-flag circuit. If one CNOT gate fails, by an argument analogous to that used for the 1-flag circuit, there will be a flag or an error of at most weight-one on the data. If the first pair of CNOT_{fm} gates fail causing no flag qubits to flag, after multiplying the data qubits by $Z^{\otimes w}$, the resulting error E_r will have $\text{wt}(E_r) \leq 2$. For any other pair of CNOT_{fm} gates that fail causing an error of weight greater than two on the data, by construction there will always be another CNOT_{fm} gate between the two that fail which will propagate a Z error to a flag qubit causing it to flag. Similarly, if pairs of CNOT_{dm} gates fail resulting in the data error E_r with $\text{wt}(E_r) \geq 2$, by construction there will always be an odd number of Z errors propagating to a flag qubit due to the CNOT_{fm} gates in between the CNOT_{dm} gates that failed causing a flag qubit to flag. The same argument applies if a failure occurs between a CNOT_{dm} and CNOT_{fm} gate.

Lastly, a proposed general w-flag circuit construction for arbitrary w is provided in Appendix B.3.

Use of flag information:

As seen in Fig. 3.9a, Fig. 3.9b, Fig. 3.10b and Fig. 3.10c, in general t-flag circuits require more than one flag qubit. Apart from their use in satisfying the t-flag circuit properties, the extra flag qubits could be used to reduce the size of the flag error sets (defined in Def. 12) when verifying the Flag t-FTEC condition of Sec. 3.4. To do so, we first define f, where f is a bit string of length u (here u is the number of flag qubits) with $f_i = 1$ if the i'th flag qubit flagged and 0 otherwise. In this case, the correction set of Eq. 3.4 can be modified to include flag information as follows:

$$\tilde{E}_t^m(g_{i_1}, \cdots, g_{i_k}, s, f_{i_1}, \cdots, f_{i_k}) =$$
$$\begin{cases} \{E \in \mathcal{E}_m(g_{i_1}, \cdots, g_{i_k}, f_{i_1}, \cdots, f_{i_k}) \times \mathcal{E}_{t-m} \\ \text{ such that } s(E) = s\} \\ \{E_{\min}(s)\} \text{ if above set empty.}, \end{cases} \tag{3.8}$$

where $\mathcal{E}_m(g_{i_1}, \cdots, g_{i_k}, f_{i_1}, \cdots, f_{i_k})$ is the new flag error set containing only errors caused by precisely m faults spread amongst the circuits $C(g_{i_1}), C(g_{i_2}), \cdots, C(g_{i_k})$ which each gave rise to the flag outcomes $f_{i_1}, \cdots, f_{i_k}$.

Hence only errors which result from the measured flag outcome would be stored in the correction set. With enough flag qubits, this could potentially broaden the family of codes which satisfy the Flag t-FTEC condition.

3.6 Circuit level noise analysis

The purpose of this section is to demonstrate explicitly the flag 2-FTEC protocol, and to identify parameter regimes in which flag FTEC presented both here and in other works offers advantages over other existing FTEC schemes. We first analyze the logical failure rates of the $[\![19, 1, 5]\!]$ color code and compute its pseudo-threshold for the three choices of $\tilde{p}$. We then compare logical failure rates of several fault-tolerant error correction schemes applied to distance-three and distance-five stabilizer codes. The stabilizers for all of the studied codes are given in Tab. 2.1 and Tab. B.3. Logical failure rates are computed using the full circuit level noise model and simulation methods described in Sec. 3.1.1.

3.6.1 Numerical analysis of the $[\![19, 1, 5]\!]$ color code

The full circuit-level noise analysis of the flag 2-FTEC protocol applied to the $[\![19, 1, 5]\!]$ color code was performed using the stabilizer measurement circuits of Fig. 3.1b and Fig. 3.6a.

In the weight-six stabilizer measurement circuit of Fig. 3.6a, there are 10 CNOT gates, three measurement and state-preparation locations, and 230 resting qubit locations. When measuring all stabilizer generators using non-flag circuits, there are 42 CNOT and 42 XNOT gates, 18 measurement and state-preparation locations, and 2196 resting qubit locations. Consequently, we expect the error suppression capabilities of the flag EC scheme to depend strongly on the number of idle qubit locations.

three-qubit flag EC	pseudo-threshold
$[\![19,1,5]\!]$ and $\tilde{p}=p$	$p_{\text{pseudo}}=(1.14\pm0.02)\times10^{-5}$
$[\![19,1,5]\!]$ and $\tilde{p}=\frac{p}{10}$	$p_{\text{pseudo}}=(6.70\pm0.07)\times10^{-5}$
$[\![19,1,5]\!]$ and $\tilde{p}=\frac{p}{100}$	$p_{\text{pseudo}}=(7.74\pm0.16)\times10^{-5}$

Table 3.2: Table containing pseudo-threshold values for the flag 2-FTEC protocol applied to the $[\![19,1,5]\!]$ color code for $\tilde{p}=p$, $\tilde{p}=p/10$ and $\tilde{p}=p/100$.

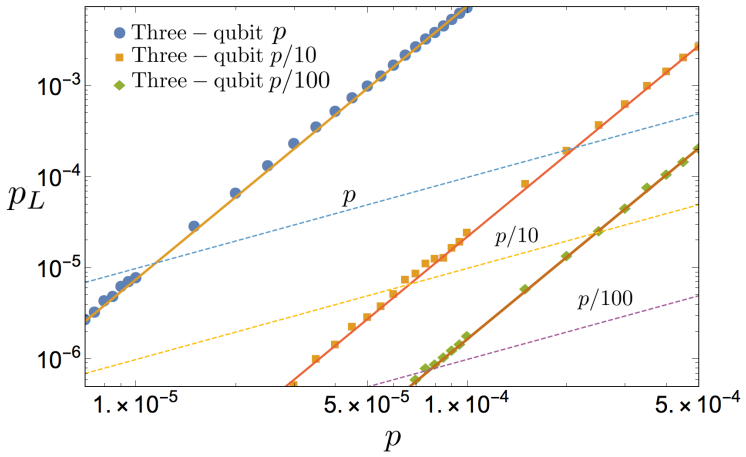


Figure 3.11: Logical failure rates of the $[\![19,1,5]\!]$ color code after implementing the flag 2-FTEC protocol presented in Sec. 3.2.2 for the three noise models described in Sec. 3.1.1. The dashed curves represent the lines $\tilde{p}=p$, $\tilde{p}=p/10$ and $\tilde{p}=p/100$. The crossing point between $\tilde{p}$ and the curve corresponding to $p_L^{([\![19,1,5]\!])}(\tilde{p})$ in Eq. 3.3 gives the pseudo-threshold.

Pseudo-thresholds of the $[\![19,1,5]\!]$ code were obtained using the methods of Sec. 3.1.1. Recall that for extending the lifetime of a qubit (when idle qubit locations fail with probability $\tilde{p}$), the probability of failure after implementing an FTEC protocol should be smaller than $\tilde{p}$. We calculated the pseudo-threshold using Eq. 3.3 for the three cases were idle qubits failed with probability $\tilde{p}=p$, $\tilde{p}=p/10$ and $\tilde{p}=p/100$. The results are shown in Tab. 3.2.

The logical failure rates for the three noise models are shown in Fig. 3.11. It can be seen that when the probability of error on a resting qubit decreases from p to $p/10$, the pseudo-threshold improves by nearly a factor of six showing the strong dependence of the scheme on the probability of failure of idle qubits.

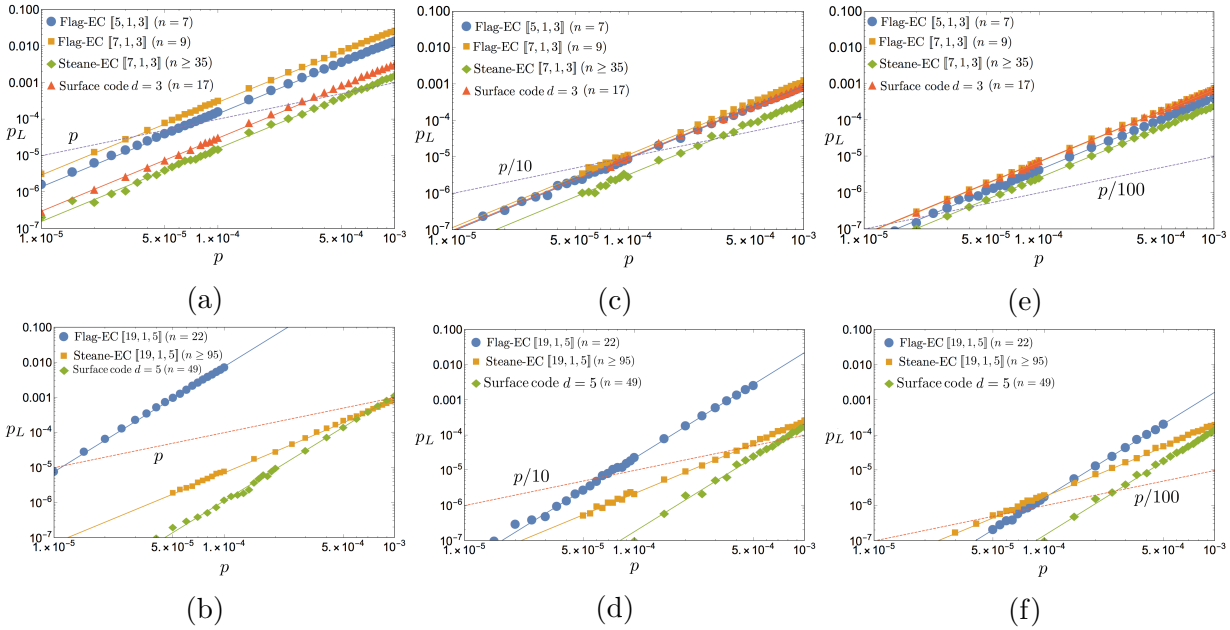


Figure 3.12: Logical failure rates for various fault-tolerant error correction methods applied to the $[\![5, 1, 3]\!]$ code, $[\![7, 1, 3]\!]$ Steane code and the $[\![19, 1, 5]\!]$ color code. The dashed curves correspond to the lines $\tilde{p} = p$, $\tilde{p} = p/10$ and $\tilde{p} = p/100$. In (a), (c) and (e), the flag 1-FTEC protocol is applied to the $[\![5, 1, 3]\!]$ and Steane code and the results are compared with the $d = 3$ surface code and Steane error correction applied to the Steane code. In (b), (d) and (f), the flag 2-FTEC protocol is applied to the $[\![19, 1, 5]\!]$ color code, and the results are compared with the $d = 5$ surface code and Steane error correction applied to the $[\![19, 1, 5]\!]$ color code. These numerical results suggest the following fault-tolerant experiments of the schemes we consider for extending the fidelity of a qubit. (1) If $7 \leq n \leq 16$, only the 5 and 7 qubit codes with flag 1-FTEC are accessible. However, the performance is much worse than higher qubit alternatives unless $\tilde{p}/p$ is small. (2) For $17 \leq n \leq 34$, the $d = 3$ surface code seems most promising, unless $\tilde{p}/p$ is small, in which case flag 2-FTEC with the 19-qubit code should be better. (3) For $35 \leq n \leq 48$, Steane EC applied to distance-three codes is better than all other approaches studied, except for very low p where flag 2-FTEC should be better due to ability to correct two rather than just one fault. (4) For $n \geq 49$, the d=5 surface code is expected to perform better than the other alternatives below pseudo-threshold.

3.6.2 Comparison of flag 1- and 2-FTEC with other FTEC schemes

The most promising schemes for testing fault-tolerance in near term quantum devices are those which achieve high pseudo-thresholds while maintaining a low qubit overhead. The flag FTEC protocol presented in this section uses fewer qubits compared to other well known fault-tolerance schemes but typically has increased circuit depth. In this section we apply the flag FTEC protocol of Sec. 3.2.1 and Sec. 3.2.2 to the $[\![5,1,3]\!]$, $[\![7,1,3]\!]$ and $[\![19,1,5]\!]$ codes. We compare logical failure rates for three values of $\tilde{p}$ with Steane error correction applied to the $[\![7,1,3]\!]$ and $[\![19,1,5]\!]$ codes and with the $d=3$ and $d=5$ rotated surface code. More details on Steane error correction and surface codes relevant for this section are provided in Appendix. B.5 and Appendix. B.6. Steane error correction is also discussed in detail in Sec. 4.2.2. Note that recent work by Goto has provided optimizations to prepare Steane ancillas [66]. However, our numerical results for Steane-EC were produced using the methods presented in Appendix. B.5.

Results of the logical failure rates for $\tilde{p}=p$, $\tilde{p}=p/10$ and $\tilde{p}=p/100$ are shown in Fig. 3.12. Various pseudo-thresholds and required time-steps for the considered fault-tolerant error correction methods are given in Tab. 3.3 and Tab. 3.4.

The circuits for measuring the stabilizers of the 5-qubit code were similar to the ones used in Fig. 3.1b (for an X Pauli replace the CNOT by an XNOT). For flag-FTEC methods, it can be seen that the $[\![5,1,3]\!]$ code always achieves lower logical failure rates compared to the $[\![7,1,3]\!]$ code. However, when $\tilde{p}=p$, both the $d=3$ surface code as well as Steane-EC achieves lower logical failure rates (with Steane-EC achieving the best performance). For $\tilde{p}=p/10$, flag-EC applied to the $[\![5,1,3]\!]$ code achieves nearly identical logical failure rates compared to the $d=3$ surface code. For $\tilde{p}=p/100$, flag 1-FTEC applied to the $[\![5,1,3]\!]$ code achieves lower logical failure rates than the $d=3$ surface code but still has higher logical failure rates compared to Steane-EC.

We also note that the pseudo-threshold increases when $\tilde{p}$ goes from p to $p/10$ for both the $[\![5,1,3]\!]$ and $[\![7,1,3]\!]$ codes when implemented using the flag 1-FTEC protocol. This is primarily due to the large circuit depth in flag-EC protocols since idle qubits locations significantly outnumber other gate locations. For the surface code, the opposite behaviour is observed. As was shown in [7], CNOT gate failures have the largest impact on the pseudo-threshold of the surface code. Thus, when idle qubits have lower failure probability, lower physical error rates will be required in order to achieve better logical failure rates. For instance, if idle qubits never failed, then performing error correction would be guaranteed to *increase* the probability of failure due to the non-zero failure probability of other types of locations (CNOT, measurements and state-preparation). Lastly, the pseudo-threshold for Steane-EC also decreases with lower idle qubit failure rates, but the change in pseudo-

FTEC scheme	Noise model	Number of qubits	Time steps (T_{time})	Pseudo-threshold
Flag-EC $[\![5,1,3]\!]$	$\tilde{p} = p$	7	$64 \leq T_{\text{time}} \leq 88$	$p_{\text{pseudo}} = 7.09 \times 10^{-5}$
Flag-EC $[\![7,1,3]\!]$		9	$72 \leq T_{\text{time}} \leq 108$	$p_{\text{pseudo}} = 3.39 \times 10^{-5}$
$d = 3$ Surface code		17	≥ 18	$p_{\text{pseudo}} = 3.29 \times 10^{-4}$
Steane-EC $[\![7,1,3]\!]$		≥ 35	15	$p_{\text{pseudo}} = 6.29 \times 10^{-4}$
Flag-EC $[\![5,1,3]\!]$	$\tilde{p} = p/10$	7	$64 \leq T_{\text{time}} \leq 88$	$p_{\text{pseudo}} = 1.11 \times 10^{-4}$
Flag-EC $[\![7,1,3]\!]$		9	$72 \leq T_{\text{time}} \leq 108$	$p_{\text{pseudo}} = 8.68 \times 10^{-5}$
$d = 3$ Surface code		17	≥ 18	$p_{\text{pseudo}} = 1.04 \times 10^{-4}$
Steane-EC $[\![7,1,3]\!]$		≥ 35	15	$p_{\text{pseudo}} = 3.08 \times 10^{-4}$
Flag-EC $[\![5,1,3]\!]$	$\tilde{p} = p/100$	7	$64 \leq T_{\text{time}} \leq 88$	$p_{\text{pseudo}} = 2.32 \times 10^{-5}$
Flag-EC $[\![7,1,3]\!]$		9	$72 \leq T_{\text{time}} \leq 108$	$p_{\text{pseudo}} = 1.41 \times 10^{-5}$
$d = 3$ Surface code		17	≥ 18	$p_{\text{pseudo}} = 1.37 \times 10^{-5}$
Steane-EC $[\![7,1,3]\!]$		≥ 35	15	$p_{\text{pseudo}} = 3.84 \times 10^{-5}$

Table 3.3: Distance-three pseudo-threshold results for various FTEC protocols and noise models applied to the $[\![5,1,3]\!]$, $[\![7,1,3]\!]$ and $d = 3$ rotated surface code. We also include the number of time steps required to implement the protocols.

threshold is not as large as the surface code. This is primarily due to the fact that all CNOT gates are applied transversally in Steane-EC, so that the pseudo-threshold is not as sensitive to CNOT errors compared to the surface code. Furthermore, most high-weight errors arising during the state-preparation of the logical ancilla's will be detected (see Appendix. B.5). Hence, idle qubit errors play a larger role than in the surface code, but Steane-EC has fewer idle qubit locations compared to flag-EC (see Tab. 3.3 for the circuit depths of all schemes).

Although Steane-EC achieves the lowest logical failure rates compared to the other fault-tolerant error correction schemes, it requires a minimum of 35 qubits (more details are provided in Appendix. B.5). In contrast, the $d = 3$ surface code requires 17 qubits, and flag 1-FTEC applied to the $[\![5,1,3]\!]$ code requires only 7 qubits. Therefore, if the probability of idle qubit errors is much lower than gate, state preparation and measurement errors, flag-FTEC methods could be good candidates for early fault-tolerant experiments.

It is important to keep in mind that for the flag 1-FTEC protocol applied to the distance-three codes considered in this section, the same ancilla qubits are used to measure all stabilizers. A more parallelized version of flag-FTEC applied to the $[\![7,1,3]\!]$ code using four ancilla qubits instead of two is considered in Appendix. B.7.

In computing the number of time steps required by the flag t-FTEC protocols, a lower bound is given in the case where there are no flags and the same syndrome is repeated

FTEC scheme	Noise model	Number of qubits	Time steps (T_{time})	Pseudo-threshold
Flag-EC $[\![19,1,5]\!]$	$\tilde{p}=p$	22	$504 \leq T_{\text{time}} \leq 960$	$p_{\text{pseudo}} = 1.14 \times 10^{-5}$
$d=5$ Surface code		49	≥ 18	$p_{\text{pseudo}} = 9.41 \times 10^{-4}$
Steane-EC $[\![19,1,5]\!]$		≥ 95	15	$p_{\text{pseudo}} = 1.18 \times 10^{-3}$
Flag-EC $[\![19,1,5]\!]$	$\tilde{p}=p/10$	22	$504 \leq T_{\text{time}} \leq 960$	$p_{\text{pseudo}} = 6.70 \times 10^{-5}$
$d=5$ Surface code		49	≥ 18	$p_{\text{pseudo}} = 7.38 \times 10^{-4}$
Steane-EC $[\![19,1,5]\!]$		≥ 95	15	$p_{\text{pseudo}} = 4.42 \times 10^{-4}$
Flag-EC $[\![19,1,5]\!]$	$\tilde{p}=p/100$	22	$504 \leq T_{\text{time}} \leq 960$	$p_{\text{pseudo}} = 7.74 \times 10^{-5}$
$d=5$ Surface code		49	≥ 18	$p_{\text{pseudo}} = 2.63 \times 10^{-4}$
Steane-EC $[\![19,1,5]\!]$		≥ 95	15	$p_{\text{pseudo}} = 5.60 \times 10^{-5}$

Table 3.4: Distance-five pseudo-threshold results for various FTEC protocols and noise models applied to the $[\![19,1,5]\!]$ color code and $d=5$ rotated surface code. We also include the number of time steps required to implement the protocols.

$t+1$ times. In Sec. 3.4 it was shown that the full syndrome measurement for flag-FTEC is repeated at most $\frac{1}{2}(t^2+3t+2)$ times where $t=\lfloor (d-1)/2 \rfloor$. An upper bound on the total number of required time steps is thus obtained from a worst case scenario where syndrome measurements are repeated $\frac{1}{2}(t^2+3t+2)$ times.

For distance-five codes, the first thing to notice from Fig. 3.12 is that the slopes of the logical failure rate curves of flag-EC applied to the $[\![19,1,5]\!]$ code and $d=5$ surface code are different from the slopes of Steane-EC applied to the $[\![19,1,5]\!]$ code. In particular, $p_{\text{L}} = cp^3 + \mathcal{O}(p^4)$ for flag-EC and the surface code whereas $p_{\text{L}} = c_1 p^2 + c_2 p^3 + \mathcal{O}(p^4)$ for Steane-EC (c, c_1 and c_2 are constants that depend on the code and FTEC method). The reason that Steane-EC has non-zero $\mathcal{O}(p^2)$ contributions to the logical failure rates is that there are instances where errors occurring at two different locations can lead to a logical fault. Consequently, the Steane-EC method that was used is not strictly fault-tolerant according to Def. 5. In Appendix. B.5, more details on the fault tolerant properties of Steane-EC are provided and a fully fault-tolerant implementation of Steane-EC is analyzed (at the cost of using more qubits).

For $d=5$, the surface code achieves significantly lower logical failure rates compared to all other distance 5 schemes but uses 49 qubits instead of 22 for the $[\![19,1,5]\!]$ code. Furthermore, due the differences in the slopes of the flag-2 FTEC protocol compared with Steane-EC applied to the $[\![19,1,5]\!]$ code, there is a regime where flag-2 FTEC achieves lower logical failure rates compared to Steane-EC. For $\tilde{p}=p/100$, it can be seen in Fig. 3.12 that this regime occurs when $p \lesssim 10^{-4}$. We also note that the pseudo-threshold of flag-EC applied to the $[\![19,1,5]\!]$ color code increases for all noise models whereas the pseudo-

threshold decreases for the other FTEC schemes. Again, this is due to the fact that flag-EC has a larger circuit depth compared to the other FTEC methods and is thus more sensitive to idle qubit errors.

Comparing the flag 2-FTEC protocol (applied to the $[\![19, 1, 5]\!]$ color code) to all the $d = 3$ schemes that were considered in this section, due to the higher distance of the 19-qubit code, there will always be a parameter regime where the 19-qubit color code acheives lower logical failure rates than both the $d = 3$ surface code and Steane-EC applied to the $[\![7, 1, 3]\!]$ code. In the case where $\tilde{p} = p/100$ and with $p \lesssim 1.5 \times 10^{-4}$, using flag error correction with only 22 qubits outperforms Steane error correction (which uses a minimum of 35 qubits) and the $d = 3$ rotated surface code (which uses 17 qubits).

Note the considerable number of time steps involved in a round of flag-EC, particularly in the $d = 5$ case (see Tab. 3.4). For many applications, this is a major drawback, for example for quantum computation when the time of an error correction round dictates the time of a logical gate. However there are some cases in which having a larger number of time-steps in an EC round while holding the logical error rate fixed is advantageous as it corresponds to a longer physical lifetime of the encoded information. Such schemes could be useful for example in demonstrating that encoded logical quantum information can be stored for longer time scales in the lab using repeated rounds of FTEC.

3.7 Summary and outlook

Building on definitions and a new flag FTEC protocol applied to distance-three and -five codes presented in Sec. 3.2, in Sec. 3.4.1 we presented a general flag FTEC protocol, which we called flag t-FTEC, and which is applicable to stabilizer codes of distance $d = 2t + 1$ that satisfy the flag t-FTEC condition. The protocol makes use of flag ancilla qubits which signal when v faults lead to errors of weight greater than v on the data when performing stabilizer measurements. In Sec. 3.3 and Sec. 3.5 we gave explicit circuit constructions, including those needed for distance 3 and 5 codes measuring stabilizers of weight 4, 6 and 8. In Sec. 3.4.2 we gave a sufficient condition for codes to satisfy the requirements for flag t-FTEC. Quantum Reed-Muller codes, Surface codes and hexagonal lattice color codes were shown to be families of codes that satisfy the sufficient condition.

The flag t-FTEC protocol could be useful for fault-tolerant experiments performed in near term quantum devices since it tends to use fewer qubits than other FTEC schemes such as Steane, Knill and Shor EC. In Sec. 3.6.2 we provided numerical evidence that with only 22 qubits, the flag 2-FTEC protocol applied to the $[\![19, 1, 5]\!]$ color code can achieve

lower logical failure rates than other codes using similar numbers of qubits such as the rotated distance-3 surface code and Steane-EC applied to the Steane code.

A clear direction of future work would be to find optimal general constructions of t-flag circuits for stabilizers of arbitrary weight that improve upon the general construction given in Appendix. B.3. Of particular interest would be circuits using few flag qubits and CNOT gates while minimizing the probability of false-positives (i.e. when the circuit flags without a high-weight error occurring). Finding other families of stabilizer codes which satisfy the sufficient or more general condition for flag t-FTEC would also be of great interest. One could also envisage hybrid schemes combining flag EC with other FTEC approaches.

Another direction of future research would be to find general circuit constructions for simultaneously measuring multiple stabilizers while minimizing the number of required ancilla qubits. Further, we believe performing a rigorous numerical analysis to understand the impact of more compact circuit constructions on the codes threshold is of great interest.

Lastly, the decoding complexity (i.e. generating the flag error set lookup tables) is limited by the decoding complexity of the code. In some cases, for example concatenated codes, it may be possible to exploit some structure to generate the flag error sets more efficiently. In the case of concatenated codes, the decoding complexity would be reduced to the decoding complexity of the codes used at every level. Finding other scalable constructions for efficient decoding schemes using flag error correction remains an open problem.

Chapter 4

Deep neural decoders for near term fault-tolerant experiments

The material of this section is based on the journal article of Ref.[67], copyrighted by 2018 IOP Publishing. Both authors contributed equally to the work. I developed the main ideas for the project and performed all the simulations to generate the training set data. Pooya Ronagh performed the machine learning experiments and discussed the classical resources required to implement the techniques. Both authors contributed equally to the writing and editing of the manuscript.

4.1 Introduction and Motivation

Recently, significant progress has been made in building small quantum devices with enough qubits allowing them to be potential candidates for several quantum information experiments [68, 69, 70, 71]. Fault-tolerant quantum computing is one such avenue that has so far had a very limited experimental analysis [72].

In recent years, several fault-tolerant protocols for both error correction and universal quantum computation have been proposed, each with their own trade-offs [7, 12, 13, 55, 73, 57, 58, 42, 74, 19, 75, 63, 76, 77, 59]. In Chapter 2 we developed new hard decoding algorithms which that can adapt to known noise models in order to achieve better error correcting capabilities. In [31, 78], tensor network algorithms were used for simulating the surface code and obtaining efficient decoders for general noise features. However, the above schemes are not adapted to fault-tolerant protocols where gate and measurement errors

plays a significant role. Furthermore, some knowledge of the noise is required in order for the decoding protocols to achieve good performance. This can be a significant drawback since it is often very difficult to fully characterize the noise in realistic quantum devices.

The above challenges motivate alternative methods for finding efficient decoders which can offer improvements over more standard methods such as minimum weight perfect matching for topological codes [79, 80] and message passing for concatenated codes [29]. One interesting idea is using deep neural networks for constructing decoders which are both efficient and can tolerate large noise rates. The hope is that even if the underlying noise model is completely unknown, with enough experimental data, deep neural networks could learn the probability density functions of the different possible errors corresponding to the sequences of measured syndromes.

The first work in which machine learning was used for decoding was in a paper by Torlai and Melko [81]. In this chapter, a Boltzmann machine was trained to correct phase-flip errors of a 2-dimensional toric code. Krastanov and Jiang obtained a neural network decoder applicable to general stabilizer codes and applied it to the 2-D toric code obtaining a higher code-capacity threshold than previous results [82]. Varsamopoulos, Criger and Bertels used a feed-forward neural network to decode the surface code [83]. They also applied their decoding scheme to the distance three surface code under a full circuit level noise model. Baireuther, O'Brien, Tarasinski and Beenakker used a recurrent neural network that could be trained with experimental data [84]. They applied their decoding scheme to compare the lifetime of qubits encoded in a distance-three surface code. The analysis was based on a full circuit level noise model, albeit with a modified CNOT gate error model. Breuckmann and Ni [85] gave a scalable neural decoder applicable to higher dimensional codes by taking advantage of the fact that these codes have local decoders. To our knowledge, these methods could not be applied to codes of dimensions less than four. After our work was released, Maskara, Kubica and Jochym-O'Connor used neural-network decoders to study the code capacity thresholds of color codes [86].

Despite the numerous works in using neural networks for decoding, there are still several open questions that remain:

1. What are the fastest possible decoders that can be achieved using neural networks and how does the decoding time compare to gate times in realistic quantum devices?

2. Can neural networks still offer good performance beyond distance three codes in a full circuit level noise model regime? If so, what are the limitations?

3. How well do neural networks perform near and below typical thresholds of fault-tolerant schemes under full circuit level noise models?

In this chapter we aim to address the above questions. We apply a plethora of neural network methods to analyze several fault-tolerant error correction schemes such as the surface code as well as the CNOT-exRec gate using Steane error correction (EC) and Knill-EC, and consider both distance-three *and* distance-five codes. We chose the CNOT-exRec circuit since (in most cases) it limits the threshold of the underlying code when used with Steane and Knill-EC units [18]. Our analysis is done using the full circuit level noise mode of Sec. 3.1.1. Furthermore our methods are designed to work with experimental data; i.e. no knowledge of the underlying noise model is required.

Lastly, we provide a rigorous analysis of the decoding times of the neural network decoders and compare our results with expected gate delays in future superconducting quantum devices. We suspect that even though inference from a trained neural network is a simple procedure comprising only of matrix multiplications and arithmetic operations, state-of-the-art parallel processing and high performance computing techniques would need to be employed in order for the inference to provide a reliable decoder given the anticipated gate times in future quantum devices.

The deep neural decoders (DND) we design in this chapter assist a *baseline* decoder. For the baseline decoders, we will use both lookup table and *naive* decoding schemes which will be described in Sec. 4.2. The goal of the deep neural decoder is to determine whether to add logical corrections to the corrections provided by the baseline decoders. Although the lookup table decoder is limited to small codes, the naive decoder can efficiently be implemented for arbitrary distance codes.

We stress that to offer a proper analysis of the performance of neural network decoders, the neural network should be trained for *all* considered physical error rates. We believe that from an experimental point of view, it is not realistic to apply a network trained for large physical error rates to lower rate noise regimes. The reason is simply that the network will be trained based on the hardware that is provided by the experimentalist. If the experimentalist tunes the device to make it noisier so that fewer non-trivial training samples are provided to the neural network, the decoder could be fine tuned to a different noise model than what was present in the original device. As will be shown, training neural networks at low error rates is a difficult task for machine learning and definitely an interesting challenge.

Our goal has been to write this chapter in such a way that makes it accessible to both quantum information scientists and machine learning experts. The chapter is structured as follows.

In Sec. 4.2.1, we review the rotated surface code and provide a new decoding algorithm that is particularly well adapted for deep neural decoders. In Sec. 4.2.2 and Sec. 4.2.3, we

review the Steane and Knill fault-tolerant error correction methods. In Sec. 4.2.4 we give a description of the naive decoder and in Sec. 4.2.5 we discuss the decoding complexity of both the lookup table and naive decoders.

Sec. 4.3 focuses on the deep neural decoders constructed, trained and analyzed in this chapter. In Sec. 4.3.1 we give an overview of deep learning by using the application of error decoding as a working example. We introduce three widely used architectures for deep neural networks: (1) simple feedforward networks with fully connected hidden layers, (2) recurrent neural networks, and (3) convolutional neural networks. We introduce hyperparameter tuning as a commonly used technique in machine learning and an important research tool for machine learning experts. In Sec. 4.3.2 and Sec. 4.3.3 we introduce the deep neural network architectures we designed for decoding the CNOT-exRec circuits in the case of Steane- and Knill-EC, and for multiple rounds of EC in the case of the rotated surface code.

In Sec. 4.4 we provide our numerical results by simulating the above circuits under a full circuit level depolarizing noise channel, and feeding the results as training and test datasets for various deep neural decoders.

Finally, in Sec. 4.5 we address the question of practical applicability of deep neural decoders in their inference mode for fault-tolerant quantum error correction. We will address several hardware and software considerations and recommend a new development in machine learning known as network quantization as a suitable technology for decoding quantum error correcting codes.

4.2 Fault-tolerant protocols

In this section we will describe the fault-tolerant protocols considered in this chapter. The surface code will be described in Sec. 4.2.1 while Steane and Knill error correction will be described in Sec. 4.2.2 and Sec. 4.2.3. For each protocol, we will also describe the baseline decoder used prior to implementing a deep neural decoder (DND). Since we are focusing on near term fault-tolerant experiments, we will first describe decoding schemes using lookup tables which can be implemented extremely quickly for small distance codes. In Sec. 4.4 we will show that the lookup table decoding schemes provide very competitive pseudo-thresholds. With existing computing resources and the code families considered in this chapter, the proposed decoders can be used for distances $d \leq 7$. For example, the distance-nine color code would require 8.8 exabytes of memory to store the lookup table. Lastly, in Sec. 4.2.4 we will describe a naive decoder which is scalable and can be

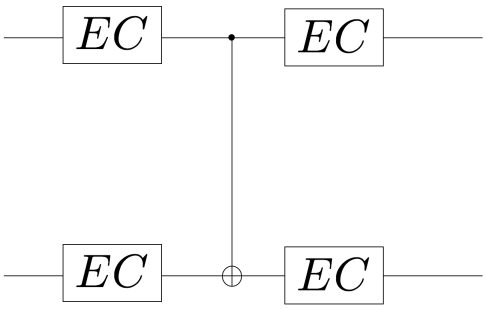


Figure 4.1: Illustration of an extended rectangle (exRec) for a logical CNOT gate. The EC box consists of performing a round of fault-tolerant error correction. The error correction rounds prior to applying the logical CNOT gate are referred to as leading-EC's (LEC) and the error correction rounds after the CNOT are referred to as trailing-EC's (TEC).

implemented efficiently while achieving competitive logical failure rates when paired with a deep neural decoder.

In this chapter, we will be considering the noise model of Sec. 3.1.1 and all our fault-tolerant protocols will be chosen to satisfy Def. 5.

Since we are focusing on small distance codes which could potentially be implemented in near term fault-tolerant experiments, when comparing the performance of fault-tolerant error correction protocols, we need to consider a full *extended rectangle* (exRec) which consists of leading and trailing error correction rounds in between logical gates. Note that this also applies to topological codes. An example of an exRec is given in Fig. 4.1. We refer the reader to [18, 34] for further details on exRec's.

4.2.1 Rotated surface code

In this section we focus on the rotated surface code [45, 46, 7, 61, 4, 62] first introduced in Sec. 3.4.2. We remind the reader that rotated surface code is a $[\![d^2, 1, d]\!]$ stabilizer code with qubits arranged on a 2-dimensional lattice as shown in Fig. 4.2. Any logical X operator has X operators acting on at least d qubits with one X operator in each row of the lattice involving an even number of green faces. Similarly, any logical Z operator has Z operators acting on at least d qubits with one Z operator in every column of the lattice involving an even number of red faces.

It is possible to measure all the stabilizer generators by performing only nearest neighbor interactions between the data qubits and neighboring ancilla qubits. The circuits used to

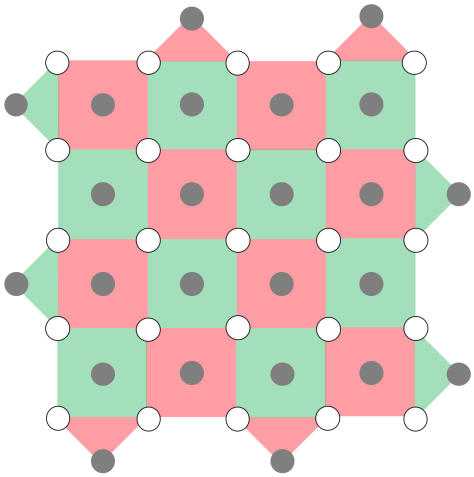
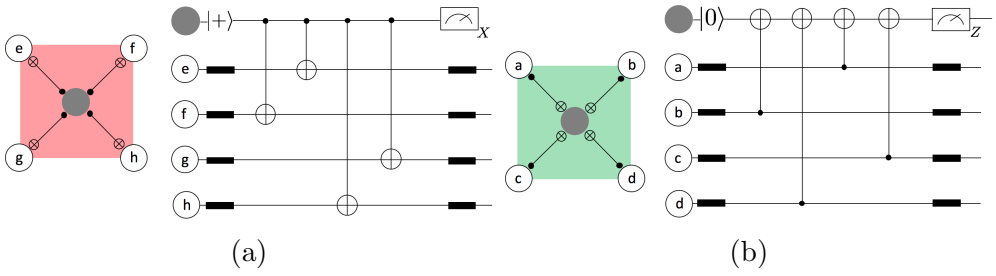


Figure 4.2: Illustration of the $d = 5$ rotated surface code. Data qubits are located at the white circles and the ancilla qubits used to measure the stabilizers are located on the black circles of the lattice. Green squares measure the Z stabilizers and red squares measure X stabilizers.



(a) (b)

Figure 4.3: Fig. 4.3a illustrates the circuit used to measure the stabilizer $X^{\otimes 4}$ and Fig. 4.3b illustrates the circuit used to measure the stabilizer $Z^{\otimes 4}$. As can be seen, a full surface code measurement cycle is implemented in six time steps.

measure both X and Z stabilizers are shown in Fig. 4.3. Note that all stabilizer generators are of weight two or four regardless of the size of the lattice.

Several decoding protocols have been devised for topological codes. Ideally, we would like decoders which have extremely fast decoding times to prevent errors from accumulating in hardware during the classical processing time while also having very high thresholds. The most common algorithm for decoding topological codes is Edmond's perfect matching algorithm (PMA) [79]. Although the best known thresholds for topological codes under circuit level noise have been achieved using a slightly modified version of PMA [80], the decoding algorithm has a worst case complexity of $\mathcal{O}(n^3)$. Recent progress has shown that minimum weight perfect matching can be performed in $\mathcal{O}(1)$ time on average given constant computing resources per unit area on a 2D quantum computer [87]. With a single processing element and given n detection events, the runtime can be made $\mathcal{O}(n)$ [88]. Renormalization group (RG) decoders have been devised that can achieve $\mathcal{O}(\log n)$ decoding times under parallelization [89, 90, 91]. However such decoders typically have lower thresholds than PMA. Wootton and Loss [92] use a Markov chain Monte Carlo method to obtain near optimal code capacity noise thresholds of the surface code at the cost of slower decoding times compared to other schemes. Recently, Delfosse and Nickerson [93] have devised a near linear time decoder for topological codes that achieves thresholds slightly lower than PMA for the 2-dimensional toric code.

Here we construct a decoder for the surface code which has extremely fast decoding times and achieves high pseudo-thresholds which will serve as a core for our deep neural decoder construction of Sec. 4.3. Our decoder will be based on a lookup table construction which could be used for distances $d \leq 7$. Before describing the construction of the lookup table, we point out that a single fault on the second or third CNOT gates in Fig. 4.3a and Fig. 4.3b can propagate to a data qubit error of weight-two (see also Appendix. B.6 for more details). Thus for a surface code that can correct $t = 2d + 1$ errors, a correction E' for an error E resulting from t faults, with $E' \sim E$, must be used when the syndrome $s(E)$ is measured. In other words, the minimum weight correction must not always be used for errors that result from faults occurring at the CNOT gates mentioned above.

With the above in mind, the lookup table is constructed a follows. For every $1 \leq m \leq 2^{d^2-1}$, use the lowest weight error $E' \sim E$ such that converting the bit string $s(E)$ to decimal results in m. If E is an error that results from $v \leq t = 2d + 1$ faults with $\text{wt}(E) > t$, then use $E' \sim E$ instead of the lowest weight error corresponding to the syndrome $s(E)$. Note that for this method to work, all errors E with $\text{wt}(E) \leq t$ must have distinct syndromes from errors E' that arise from $v \leq t$ faults with $\text{wt}(E') > t$. However this will always be the case for surface codes with the CNOT ordering chosen in Fig. 4.3.

Note that with the above construction, after measuring the syndrome s, decoding simply consists of converting s to decimal (say m) and correcting by choosing the error on the m'th row of the lookup table. However, this method is not scalable since the number of syndromes scales exponentially with the code distance.

Lastly, the decoding scheme as currently stated is not fault-tolerant. The reason is that if syndromes are measured only once, in some cases it would be impossible to distinguish data qubit errors from measurement errors. For instance, a measurement error occurring when measuring the green triangle of the upper left corner of Fig. 4.2 would result in the same syndrome as an X error on the first data qubit. However, with a simple modification, the surface code decoder can be made fault-tolerant. For distance 3 codes, the syndrome is measured three times and we decode using the majority syndrome. If there are no majority syndromes, the syndrome from the last round is used to decode. For instance, suppose that the syndromes s_1, s_2, s_2 were obtained, then the syndrome s_2 would be used to decode with the lookup table. If all three syndromes s_1, s_2, s_3 were different, then s_3 would be used to decode with the lookup table. This decoder was shown to be fault-tolerant in Ref. [94].

For higher distance codes, we use the following scheme which is derived from the protocol presented in Sec. 3.4. First, we define the counter n_{diff} (used for keeping track of the minimum number of faults which causes changes in consecutive syndrome measurements) as follows

Decoding protocol – update rules:

Given a sequence of consecutive syndrome measurement outcomes s_k and s_{k+1}:

1. If n_{diff} did not increase in the previous round, and $s_k \neq s_{k+1}$, increase n_{diff} by one.

We also define $E(s)$ to be the correction obtained from either the lookup table decoder or naive decoder (described in section Sec. 4.2.4) using the syndrome s. With the above definition of n_{diff}, the decoding protocol for a code that can correct any error E with $\text{wt}(E) \leq t = \lfloor \frac{(d-1)}{2} \rfloor$ is implemented as

Decoding protocol – corrections:
Set $n_{\text{diff}} = 0$.
Repeat the syndrome measurement.
Update n_{diff} according to the update rule above.

1. If at anytime $n_{\text{diff}} = t$, repeat the syndrome measurement yielding the syndrome s. Apply the correction $E(s)$.

2. If the same syndrome s is repeated $t - n_{\text{diff}} + 1$ times in a row, apply the correction $E(s)$.

Note that in the above protocol, the number of times the syndrome is repeated is non-deterministic. The minimum number of syndrome measurement repetitions is $t + 1$ while in Sec. 3.4 we showed that the maximum number of syndrome measurement repetitions is $\frac{1}{2}(t^2 + 3t + 2)$. Further, a proof that the above protocol satisfies both fault-tolerance criteria in Def. 5 is given in Appendix. B.1.

4.2.2 Steane error correction

Steane error correction [43] takes advantage of properties of CSS codes (reviewed in Sec. 1.1.1) to measure the X and Z stabilizers using transversal CNOT gates. To see this, consider the circuit in Fig. 4.4a. The transversal CNOT gate between the encoded data block $|\overline{\psi}\rangle$ and ancilla $|\overline{+}\rangle$ acts trivially (i.e. $\overline{\text{CNOT}}|\overline{\psi}\rangle|\overline{+}\rangle = |\overline{\psi}\rangle|\overline{+}\rangle$). However, any X errors afflicting the data block would then be copied to the ancilla state. Furthermore, CSS codes have the property that by performing a transversal measurement of the codeword $|\overline{+}\rangle$ in the absence of errors, the result will be a codeword of C_1 chosen uniformly at random. If X errors are present on the codeword $|\overline{+}\rangle$, then the transversal measurement will yield the classical codeword $e + f + g$. Here, $(e|0)$ (written in binary symplectic form) are the X errors on the data qubits, $(f|0)$ are the X errors that arise during the preparation of the $|\overline{+}\rangle$ state and $(g|0)$ are bit-flip errors that arise during the transversal measurement. Applying the correction $X_e X_f X_g$ on the data would result in an X error of weight $f + g$. An analogous argument can be made for Z errors using the circuit of Fig. 4.4b (note that in this case we measure in the X-basis which maps $C_1 \to C_2$ and $Z \to X$).

The circuits used to prepared the encoded $|\overline{+}\rangle$ and $|\overline{0}\rangle$ states are in general not fault-tolerant. In the case of $|\overline{+}\rangle$, low weight errors can spread to high-weight X errors (which can change the outcome of the measurement) and Z errors (which can propagate to the data block due to the transversal CNOT gates). However, by preparing extra "verifier" states

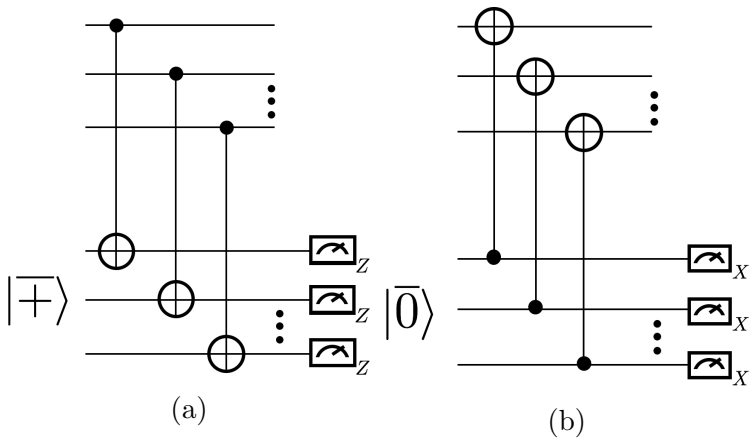


Figure 4.4: Circuits for measuring X and Z stabilizers in Steane-EC. The circuit in Fig. 4.4a measures bit-flip errors whereas the circuit in Fig. 4.4b measures phase-flip errors. Note that the first block consists of the data qubits encoded in a CSS code. The states $|\overline{0}\rangle$ and $|\overline{+}\rangle$ represent logical $|0\rangle$ and $|+\rangle$ states encoded in the same CSS code used to protect the data.

encoded in $|\overline{+}\rangle$ and coupling these states to the original $|\overline{+}\rangle$ ancilla as shown in Fig. 4.5, high weight X and Z errors arising from the ancilla can be detected. Furthermore, after a classical error correction step, the eigenvalue of $\overline{X}$ and $\overline{Z}$ can be measured. Therefore if a non-trivial syndrome is measured in the verifier states or the -1 eigenvalue of a logical operator is measured, the ancilla qubits are rejected and new ancilla qubits are brought in to start the process anew.

We would like to point out that instead of verifying the ancilla qubits for errors and rejecting them when a non-trivial syndrome is measured, it is also possible to replace the verification circuit with a decoding circuit. By performing appropriate measurements on the ancilla qubits and making use of Pauli frames [55, 95, 96], any errors arising from t-faults in the ancilla circuits can be identified and corrected [20] (note that DiVincenzo and Aliferis provided circuits for Steane's $[\![7, 1, 3]\!]$ code so that $t = 1$). However in this chapter we will focus on ancilla verification methods.

It can be shown that the Steane-EC circuit of Fig. 4.5 satisfies both fault-tolerant conditions of Def. 5 for distance-three codes [11]. It is possible to use the same ancilla verification circuits in some circumstances for higher distance codes by carefully choosing different circuits for preparing the logical $|\overline{0}\rangle$ and $|\overline{+}\rangle$ states (see [6] for some examples). For the analysis performed in this chapter, we chose appropriate $|\overline{0}\rangle$ and $|\overline{+}\rangle$ states such that the decoding schemes are fault-tolerant using the ancilla verification circuits in Fig. 4.5.

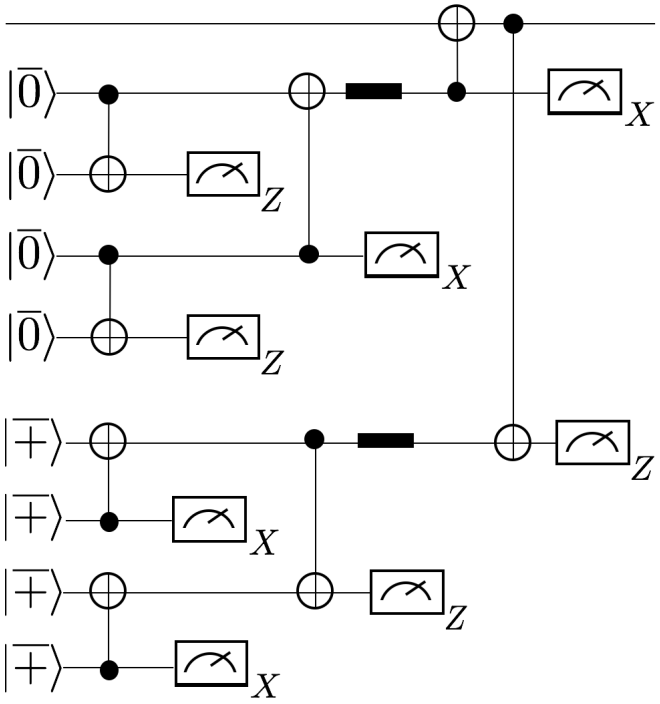


Figure 4.5: Full Steane error correction circuit. Each line represents encoded data qubits and all CNOT gates and measurements are performed transversally. The circuits used to prepare the encoded $|\overline{+}\rangle$ and $|\overline{0}\rangle$ are in general not fault-tolerant. Consequently, extra "verifier" ancilla states are used to detect errors arising during the preparation of $|\overline{+}\rangle$ and $|\overline{0}\rangle$. If the verifier states measure a non-trivial syndrome or the -1 eigenvalue of a logical Pauli is measured, the ancilla states are rejected and new ancilla states are brought in until they pass the verification step.

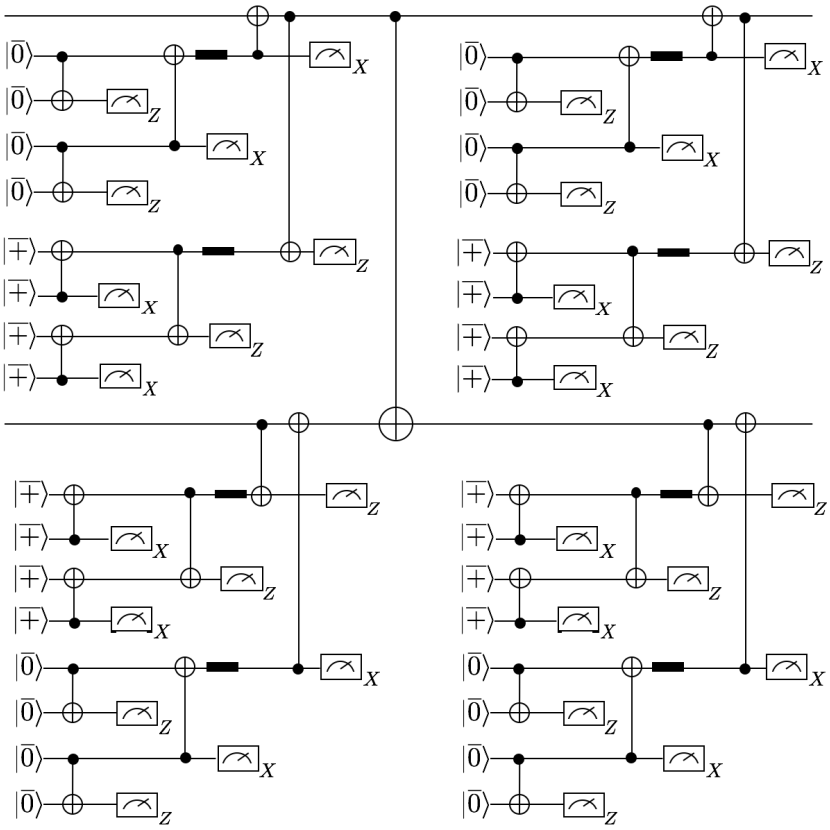


Figure 4.6: CNOT-exRec for Steane-EC which contains four EC blocks. The CNOT-exRec limits the pseudo-threshold of the $[\![7,1,3]\!]$ and $[\![19,1,5]\!]$ color code due to the large number of locations and thus makes an ideal circuit to optimize our decoding algorithm using the neural decoders described in Sec. 4.3.

We would like to add that although the order in which transversal measurements to correct bit-flip and phase-flip errors does not affect the fault-tolerant properties of Steane-EC, it does create an asymmetry in the X and Z logical failure rates [6, 35, 36]. For instance, an X error arising on the target qubit of the logical CNOT used to detect phase errors would be copied to the $|\overline{+}\rangle$ ancilla. However a Z error arising on the target of this CNOT or control of the CNOT used to correct bit-flip errors would not be copied to any of the ancilla qubits.

When analyzing both Steane and Knill-EC schemes using neural decoders, the underlying quantum error correcting codes will be the $[\![7,1,3]\!]$ Steane code and the $[\![19,1,5]\!]$ color code (see Fig. 3.3 and Fig. 3.5a for a description of these codes). To obtain a

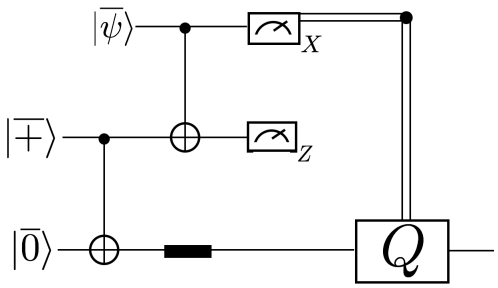


Figure 4.7: Knill error correction circuit. As with Steane-EC, all CNOT gates and measurements are performed transversally. The logical $|\overline{0}\rangle$ and $|\overline{+}\rangle$ states are also encoded using the same code that protects the data. A transversal CNOT gate is applied between them to form a logical Bell state. The operator Q is used to complete the teleportation protocol of the logical state as well as to correct errors which were on the original data block.

pseudo-threshold for both of these codes, we will consider the CNOT-exRec since it is the logical gate with the largest number of locations and thus will limit the performance of both codes [18] (here we are considering the universal gate set generated by $\langle \text{CNOT}, T, H \rangle$ where $T = \text{diag}(1, e^{i\pi/4})$ and H is the Hadamard gate [97]). The full CNOT-exRec circuit for Steane-EC is shown in Fig. 4.6. Note that the large number of CNOT gates will result in a lot of correlated errors which adds a further motivation to consider several neural network techniques to optimize the decoding performance.

4.2.3 Knill error correction

Steane error correction described in Sec. 4.2.2 only applies to stabilizer codes which are CSS codes. Further, the protocol requires two transversal CNOT gates between the data and ancilla qubits. In this section we will give an overview of Knill error correction [73, 55] which is applicable to any stabilizer code. As will be shown, Knill-EC only requires a single transversal CNOT gate between the data qubits and ancilla qubits.

Consider a Pauli operator P acting on the data block of the circuit in Fig. 4.7. Consider the same Pauli P (but with a possibly different sign) acting on the first ancilla block of the logical Bell pair. P can be any Pauli but in the argument that follows we will be interested in cases where $P \in N(\mathcal{S})$. Taking into account the sign of P and writing it as a product of X and Z, we have that

$$(-1)^{b_i} P = i^{c(P_X, P_Z)} (-1)^{b_i} P_X P_Z. \tag{4.1}$$

The function $c(P_X, P_Z) = 0$ if P_X and P_Z commute and one otherwise. The phase $i^{c(P_X,P_Z)}$ comes from the Y operators in P and $(-1)^{b_i}$ indicates the sign of the Pauli where $i = 0$ for the data block and $i = 1$ for the ancilla block.

Applying the transversal CNOT's between the ancilla and data block performs the following transformations

$$(-1)^{b_0} P \otimes I \to i^{c(P_X,P_Z)} (-1)^{b_0} P_X P_Z \otimes P_X, \tag{4.2}$$
$$(-1)^{b_1} I \otimes P \to i^{c(P_X,P_Z)} (-1)^{b_1} P_Z \otimes P_X P_Z, \tag{4.3}$$

and therefore

$$(-1)^{b_0+b_1} P \otimes P \to (-1)^{b_0+b_1+c(P_X,P_Z)} P_X \otimes P_Z. \tag{4.4}$$

From Eq. 4.4, we can deduce that a subsequent measurement of X on each physical data qubit and measurement of Z on each physical qubit in the first ancilla block lets us deduce the eigenvalue of P (since $c(P_X, P_Z)$ is known, we learn $b_0 + b_1$).

Since the above arguments apply to any Pauli, if P is a stabilizer we learn $s_0 + s_1$ where s_0 is the syndrome of the data block and s_1 is the error syndrome of the first ancilla block. Furthermore, the measurements also allow us to deduce the eigenvalues of the logical Pauli's $\overline{X}_i \otimes \overline{X}_i$ and $\overline{Z}_i \otimes \overline{Z}_i$ for every logical qubit i. This means that in addition to error correction we can also perform the logical Bell measurement required to teleport the encoded data to the second ancilla block.

Note that pre-existing errors on the data or ancilla block can change the eigenvalue of the logical operator $\overline{P} \otimes \overline{P}$ without changing the codeword that would be deduced using an ideal decoder. For instance, if E_d is the error on the data block and E_a the error on the ancilla block with $\mathrm{wt}(E_d) + \mathrm{wt}(E_a) \leq t$, then if $(-1)^b$ is the eigenvalue of $\overline{P} \otimes \overline{P}$, we would instead measure $(-1)^{b'}$ where $b' = b + c(E_d, \overline{P}) + c(E_a, \overline{P})$. The same set of measurements also lets us deduce the syndrome $s(E_d) + s(E_a) = s(E_d E_a)$. But since $\mathrm{wt}(E_d E_a \leq t)$, from $s(E_d E_a)$ we deduce the error $E' = E_a E_d M$ where $M \in \mathcal{S}$. Hence once E' is deduced, we also get the correct eigenvalue of $\overline{P} \otimes \overline{P}$ thus obtaining the correct outcome for the logical Bell measurement.

There could also be faults in the CNOT's and measurements when performing Knill-EC. We can combine the errors from the CNOT's and measurements into the Pauli G on

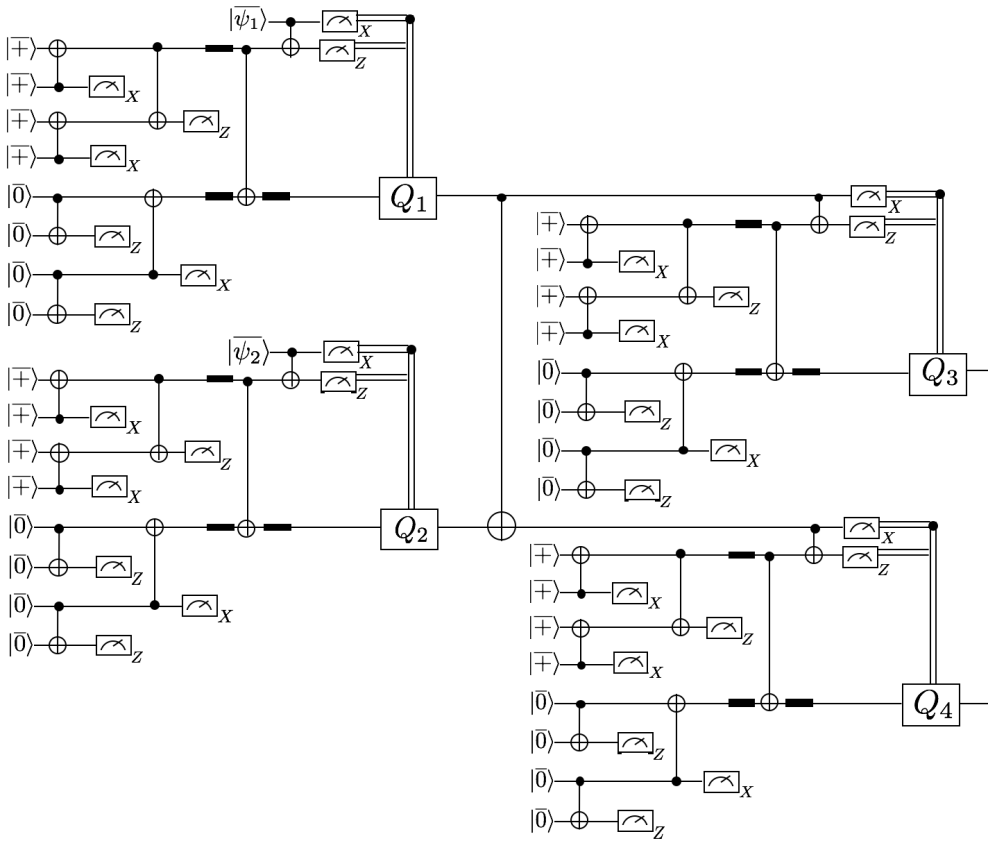


Figure 4.8: Full CNOT-exRec circuit using Knill error correction. Each Pauli operator Q_1, Q_2, Q_3 and Q_4 is used to correct errors in the initial data blocks as well as the complete teleportation protocol of the logical Bell measurement.

the data block and F on the ancilla block where the weight of GF is less than or equal to the number faults at the CNOT and measurement locations. Given the basis in which the measurements are performed, we can assume that G consists only of Z errors and F of X errors. Consequently, for a full circuit level noise model, the final measured syndrome is $s(E_d E_a GF)$.

As in Steane-EC, the circuits for preparing the logical $|\overline{0}\rangle$ and $|\overline{+}\rangle$ states are not fault-tolerant and can result in high weight errors on the data. However, if the error correcting code is a CSS code, then we can use the same ancilla verification method presented in Sec. 4.2.2 to make the full Knill-EC protocol fault-tolerant. In Fig. 4.8 we show the full CNOT-exRec circuit using Knill-EC. Note that for each EC unit, there is an extra idle

qubit location compared to Steane-EC.

Lastly, we point out that another motivation for using Knill-EC is its ability to handle leakage errors. A leakage error occurs when the state of a two-level system, which is part of a higher dimensional subspace, transitions outside of the subspace. In Ref. [98], it was shown how leakage faults can be reduced to a regular fault (which acts only on the qubit subspace) with the use of Leakage-Reduction Units (LRU's). One of the most natural ways to implement LRU's is through quantum teleportation [99]. Since Knill-EC teleports the data block to the ancilla block, unlike in Steane-EC, LRU's don't need to be inserted on the input data block. However, LRU's still need to be inserted after the preparation of every $|\overline{0}\rangle$ and $|\overline{+}\rangle$ states.

4.2.4 Naive decoder

Since the lookup table decoder scheme presented in previous sections is not scalable, it would be desirable to have a scalable and fast decoding scheme that can achieve competitive thresholds when paired with a deep neural decoder. In this section we provide a detailed description of a naive decoder which can replace the lookup table scheme in all of the above protocols.

We first note that the recovery operator R_s for a measured syndrome s can be written as [29, 89]

$$R_s = \mathcal{L}(s)\mathcal{T}(s)\mathcal{G}(s) \tag{4.5}$$

which we will refer to as the LTS decomposition of E. In Eq. 4.5, $\mathcal{L}(s)$ is a product of logical operators (operators in $N(\mathcal{S}) \setminus \mathcal{S}$), $\mathcal{G}(s)$ is a product of stabilizers (operators in $\mathcal{S}$) and $\mathcal{T}(s)$ is a product of pure errors. Pure errors form an abelian group with the property that T_i appears in $\mathcal{T}(s)$ if and only if the i'th syndrome bit is 1 (i.e. $[T_i, T_j] = 0$ and $[T_j, g_k] = \delta_{j,k}$ where g_k is the k'th stabilizer generator). Thus pure errors can be obtained from Gaussian elimination. Note that the choice of operators in $\mathcal{G}(s)$ will not affect the outcome of the recovered state. Consequently, given a measured syndrome s, decoding can be viewed as finding the most likely logical operator in $\mathcal{L}(s)$.

For a measured syndrome s, a naive decoding scheme is to always choose the recovery operator $R_l = \mathcal{T}(s)$ which is clearly suboptimal. However, for such a decoder, the decoding complexity results simply from performing the matrix multiplication $\mathbf{s}\ T$ where $\mathbf{s} = (s_1, s_2, \cdots, s_{n-k})$ is the syndrome written as a $1 \times (n-k)$ vector and T is a $(n-k) \times n$ matrix where the j'th row corresponds to T_j. The goal of all neural networks considered in Sec. 4.3 will then be to find the most likely operator $\mathcal{L}(s)$ from the input syndrome l.

The set of stabilizer generators, logical operators and pure errors for all the codes considered in this chapter are provided in Tab. 4.1. Lastly, we point out that a version of the above decoding scheme was implemented in [83] for the distance-three surface code.

4.2.5 Lookup table and naive decoder complexity

From a complexity theoretic point of view, read-out of an entry of an array or a hash table requires constant time. In hash tables, a hash function is calculated to find the address of the entry inquired. The hash function calculation takes the same processing steps for any entry, making this calculation $O(1)$. In the case of an array, the key point is that the array is a sequential block of the memory with a known initial pointer. Accessing any entry requires calculating its address in the memory by adding its index to the address of the beginning of the array. Therefore, calculating the address of an entry in an array also takes $O(1)$.

It remains to understand that accessing any location in the memory given its address is also $O(1)$ as far as the working of the memory hardware is concerned. This is the assumption behind *random access memory* (RAM) where accessing the memory comprises of a constant time operation performed by the multiplexing and demultiplexing circuitry of the RAM. This is in contrast with direct-access memories (e.g. hard disks, magnetic tapes, etc) in which the time required to read and write data depends on their physical locations on the device and the lag resulting from disk rotation and arm movement.

Given the explanation above, a decoder that relies solely on accessing recovery operators from an array operates in $O(1)$ time. This includes the lookup table and the inference mapping method of Sec. 4.5.2 below.

For the naive decoder of Sec. 4.2.4, we may also assume that the table of all pure errors (denoted as T in Sec. 4.2.4) is stored in a random access memory. However, the algorithm for generating a recovery from the naive decoder is more complicated than only accessing an element of T. With n qubits and $n-k$ syndromes, for every occurrence of 1 in the syndrome string, we access an element of T. The elements accessed in this procedure have to be added together. With parallelization, we may assume that a tree adder is used which, at every stage, adds two of the selected pure error strings to each other. Addition of every two pure error strings is performed modulo 2 which is simply the XOR of the two strings, which takes $O(1)$ time assuming parallel resources. The entire procedure therefore has a time complexity of $O((n-k)\log(n-k))$, again assuming parallel digital resources.

$[\![7,1,3]\!]$ Steane code	$[\![9,1,3]\!]$ (Surface-17) code	$[\![19,1,5]\!]$ color code	$[\![25,1,5]\!]$ (Surface-49) code
$g_1^{(x)} = X_4X_5X_6X_7$	$g_1^{(x)} = X_1X_2X_4X_5$	$g_1^{(x)} = X_1X_2X_3X_4$	$g_1^{(x)} = X_1X_2X_6X_7$
$g_2^{(x)} = X_2X_3X_6X_7$	$g_2^{(x)} = X_7X_8$	$g_2^{(x)} = X_1X_3X_5X_7$	$g_2^{(x)} = X_{11}X_{12}X_{16}X_{17}$
$g_3^{(x)} = X_1X_3X_5X_7$	$g_3^{(x)} = X_2X_3$	$g_3^{(x)} = X_5X_7X_8X_{11}X_{12}X_{13}$	$g_3^{(x)} = X_{21}X_{22}$
$g_1^{(z)} = Z_4Z_5Z_6Z_7$	$g_4^{(x)} = X_5X_6X_8X_9$	$g_4^{(x)} = X_1X_2X_5X_6X_8X_9$	$g_4^{(x)} = X_2X_3$
$g_2^{(z)} = Z_2Z_3Z_6Z_7$	$g_1^{(z)} = Z_1Z_4$	$g_5^{(x)} = X_6X_9X_{16}X_{19}$	$g_5^{(x)} = X_7X_8X_{12}X_{13}$
$g_3^{(z)} = Z_1Z_3Z_5Z_7$	$g_2^{(z)} = Z_2Z_3Z_5Z_6$	$g_6^{(x)} = X_{16}X_{17}X_{18}X_{19}$	$g_6^{(x)} = X_{17}X_{18}X_{22}X_{23}$
	$g_3^{(z)} = Z_4Z_5Z_7Z_8$	$g_7^{(x)} = X_8X_9X_{10}X_{11}X_{16}X_{17}$	$g_7^{(x)} = X_3X_4X_8X_9$
	$g_4^{(z)} = Z_6Z_9$	$g_8^{(x)} = X_{10}X_{11}X_{12}X_{15}$	$g_8^{(x)} = X_{13}X_{14}X_{18}X_{19}$
		$g_9^{(x)} = X_{12}X_{13}X_{14}X_{15}$	$g_9^{(x)} = X_{23}X_{24}$
		$g_1^{(z)} = Z_1Z_2Z_3Z_4$	$g_{10}^{(x)} = X_4X_5$
		$g_2^{(z)} = Z_1Z_3Z_5Z_7$	$g_{11}^{(x)} = X_9X_{10}X_{14}X_{15}$
		$g_3^{(z)} = Z_5Z_7Z_8Z_{11}Z_{12}Z_{13}$	$g_{12}^{(x)} = X_{19}X_{20}X_{24}X_{25}$
		$g_4^{(z)} = Z_1Z_2Z_5Z_6Z_8Z_9$	$g_1^{(z)} = Z_1Z_6$
		$g_5^{(z)} = Z_6Z_9Z_{16}Z_{19}$	$g_2^{(z)} = Z_2Z_3Z_7Z_8$
		$g_6^{(z)} = Z_{16}Z_{17}Z_{18}Z_{19}$	$g_3^{(z)} = Z_4Z_5Z_9Z_{10}$
		$g_7^{(z)} = Z_8Z_9Z_{10}Z_{11}Z_{16}Z_{17}$	$g_4^{(z)} = Z_6Z_7Z_{11}Z_{12}$
		$g_8^{(z)} = Z_{10}Z_{11}Z_{12}Z_{15}$	$g_5^{(z)} = Z_8Z_9Z_{13}Z_{14}$
		$g_9^{(z)} = Z_{12}Z_{13}Z_{14}Z_{15}$	$g_6^{(z)} = Z_{10}Z_{15}$
			$g_7^{(z)} = Z_{11}Z_{16}$
			$g_8^{(z)} = Z_{12}Z_{13}Z_{17}Z_{18}$
			$g_9^{(z)} = Z_{14}Z_{15}Z_{19}Z_{20}$
			$g_{10}^{(z)} = Z_{16}Z_{17}Z_{21}Z_{22}$
			$g_{11}^{(z)} = Z_{18}Z_{19}Z_{23}Z_{24}$
			$g_{12}^{(z)} = Z_{20}Z_{25}$
$T_1^{(x)} = X_4$	$T_1^{(x)} = X_1$	$T_1^{(x)} = X_4$	$T_1^{(x)} = X_1$
$T_2^{(x)} = X_2$	$T_2^{(x)} = X_3$	$T_2^{(x)} = X_3X_4$	$T_2^{(x)} = X_3$
$T_3^{(x)} = X_1$	$T_3^{(x)} = X_8$	$T_3^{(x)} = X_{13}X_{14}$	$T_3^{(x)} = X_5$
$T_1^{(z)} = Z_3Z_7$	$T_4^{(x)} = X_9$	$T_4^{(x)} = X_5X_7$	$T_4^{(x)} = X_3X_7$
$T_2^{(z)} = Z_5Z_7$	$T_1^{(z)} = Z_4$	$T_5^{(x)} = X_{18}X_{19}$	$T_5^{(x)} = X_5X_9$
$T_3^{(z)} = Z_6Z_7$	$T_2^{(z)} = Z_7$	$T_6^{(x)} = X_6X_9X_{17}$	$T_6^{(x)} = X_5X_{10}$
	$T_3^{(z)} = Z_2Z_4$	$T_7^{(x)} = X_6X_9$	$T_7^{(x)} = X_3X_7X_{11}$
	$T_4^{(z)} = Z_6$	$T_8^{(x)} = X_6X_9X_{10}$	$T_8^{(x)} = X_{18}X_{24}$
		$T_9^{(x)} = X_6X_9X_{10}X_{15}$	$T_9^{(x)} = X_5X_{10}X_{15}$
		$T_1^{(z)} = Z_2Z_5Z_7$	$T_{10}^{(x)} = X_{17}X_{18}X_{24}$
		$T_2^{(z)} = Z_1Z_2$	$T_{11}^{(x)} = X_{24}$
		$T_3^{(z)} = Z_{13}Z_{14}$	$T_{12}^{(x)} = X_5X_{10}X_{15}X_{20}$
		$T_4^{(z)} = Z_5Z_7$	$T_1^{(z)} = Z_6$
		$T_5^{(z)} = Z_{18}Z_{19}$	$T_2^{(z)} = Z_{16}$
		$T_6^{(z)} = Z_6Z_9Z_{16}Z_{18}Z_{19}$	$T_3^{(z)} = Z_{21}$
		$T_7^{(z)} = Z_6Z_9$	$T_4^{(z)} = Z_2Z_6$
		$T_8^{(z)} = Z_5Z_7Z_8Z_{11}$	$T_5^{(z)} = Z_{12}Z_{16}$
		$T_9^{(x)} = Z_6Z_9Z_{11}Z_{12}$	$T_6^{(z)} = Z_{21}Z_{22}$
			$T_7^{(z)} = Z_8Z_{12}Z_{16}$
			$T_8^{(z)} = Z_{19}Z_{25}$
			$T_9^{(z)} = Z_{21}Z_{22}Z_{23}$
			$T_{10}^{(z)} = Z_4Z_8Z_{12}Z_{16}$
			$T_{11}^{(z)} = Z_{14}Z_{19}Z_{25}$
			$T_{12}^{(z)} = Z_{25}$
$X_L = X^{\otimes 7}, Z_L = Z^{\otimes 7}$	$X_L = X_3X_5X_7, Z_L = Z_1Z_5Z_9$	$X_L = X^{\otimes 19}, Z_L = Z^{\otimes 19}$	$X_L = X_5X_9X_{13}X_{17}X_{21}, Z_L = Z_1Z_7Z_{13}Z_{19}Z_{25}$

Table 4.1: Table containing a list of the stabilizer generators (second row), pure errors (third row) and logical operators (fourth row) for all the codes considered in this article.

4.3 Deep neural decoders

In most quantum devices, fully characterizing the noise model afflicting the system can be a significant challenge. Furthermore, for *circuit level* noise models which cannot be described by Pauli channels, efficient simulations of a code's performance in a fault-tolerant implementation cannot be performed without making certain approximations (a few exceptions for repetition codes can be found in [100]). However, large codes are often required to achieve low failure rates such that long quantum computations can be performed reliably. These considerations motivate fast decoding schemes which can adapt to unknown noise models encountered in experimental settings.

Recall from Sec. 4.2.4 that decoding can be viewed as finding the most likely operator $L \in \mathcal{L}(\mathbf{s})$ given a measured syndrome $\mathbf{s}$. Since all codes considered in this chapter encode a single logical qubit, the recovery operator for a measured syndrome $\mathbf{s}$ can be written as

$$R_{\mathbf{s}} = X_L^{b_1(\mathbf{s})} Z_L^{b_2(\mathbf{s})} \mathcal{T}(\mathbf{s}) \mathcal{G}(\mathbf{s}) \tag{4.6}$$

where X_L and Z_L are the codes logical X and Z operators and $b_1(\mathbf{s}), b_2(\mathbf{s}) \in \mathbb{Z}_2$. In Chapter 2, our hard decoding algorithm applicable to general Markovian channels was presented for finding the coefficients $b_1(\mathbf{s})$ and $b_2(\mathbf{s})$ which optimized the performance of error correcting codes. However, the algorithm required knowledge of the noise channel and could not be directly applied to circuit level noise thus adding further motivation for a neural network decoding implementation.

In practice, the deep learning schemes described in this section can be trained as follows. First, to obtain the training set, the data qubits are fault-tolerantly prepared in a known logical $|\overline{0}\rangle$ or $|\overline{+}\rangle$ state followed by a round of fault-tolerant error correction (using either the lookup table or naive decoders). The encoded *data* is then measured in the logical Z or X basis yielding a -1 eigenvalue if a logical X or Z error occurred. The training set is constructed by repeating this sequence several times both for states prepared in $|\overline{0}\rangle$ or $|\overline{+}\rangle$. For each experiment, all syndromes are recorded as well as the outcome of the logical measurement. Given the most likely error E with syndrome $s(E) = \mathbf{s}$ (in general E will not be known), the neural network must then find the vector $\mathbf{b} = (b_1(\mathbf{s}), b_2(\mathbf{s}))$ such that $X_L^{b_1(\mathbf{s})} Z_L^{b_2(\mathbf{s})} R_{\mathbf{s}} E \in \mathcal{S}$ where $R_{\mathbf{s}}$ was the original recovery operator obtained from either the lookup table or naive decoders described in Sec. 4.2.

Once the neural network is trained, to use it in the inference mode (as explained in Section 4.5.2), a query to the network simply consists of taking as input all the measured syndromes and returning as output the vector $\mathbf{b}$. For Steane and Knill EC, the syndromes are simply the outcomes of the transversal X and Z measurements in the leading and

trailing EC blocks. For the surface code, the syndromes are the outcomes of the ancilla measurements obtained from each EC round until the protocols presented in Sec. 4.2.1 terminate.

Lastly, we note that a similar protocol was used in [84] which also used the outcome of the final measurement on the data qubits to decode. However by using our method, once the neural network is trained, it only takes as input the measured syndromes in an EC round to compute the most likely $\mathbf{b}$.

4.3.1 Deep learning

Here we explain the generic framework of our deep learning experiments. We refer the reader to [101] for an introduction to deep learning and to [102] for machine learning methods in classification tasks.

Let $D \subseteq \mathcal{D}$ be a data set. In our case, $\mathcal{D} = S \times B$ is the set of all pairs of syndromes and *error labels*. Every element in D and $\mathcal{D}$ is therefore a pair $(\mathbf{s}, \mathbf{e})$ of measured syndromes $\mathbf{s} \in S$ and error labels $\mathbf{e} \in B$. The error labels can be different depending on how we model the learning problem. For instance, every $\mathbf{e} \in B$ can be a bit string carrying a prescription of recovery operators:

$$B = \{I, X, Y, Z\}^{\#\text{physical qubits}}.$$

There is however a major drawback in modelling the errors in the above fashion. For deep learning purposes the elements $\mathbf{e} \in B$ are represented in their *1-hot encoding*, i.e. a bit string consisting of only a single 1, and zeros everywhere else. The 1-hot encoding therefore needs $|E|$ bits of memory allocated to itself which by the definitions above, grows exponentially in either the number of physical qubits.

Our solution for overcoming this exponentially growing model is to take advantage of the decomposition (Eq. 4.6) of the recovery operator and only predict vectors $\mathbf{b} = (b_1(\ell), b_2(\ell))$ as explained earlier. In other words, the elements of B contain information about the logical errors remaining from the application of another auxiliary encoding scheme:

$$B = \{I, X, Y, Z\}^{\#\text{logical qubits}}.$$

The objective function. As customary in machine learning, the occurrences $x = (\mathbf{s}, \mathbf{b}) \in D$ are viewed as statistics gathered from a conditional probability distribution function $p(x) = \mathbb{P}(\mathbf{b} \,|\, \mathbf{s})$ defined over $S \times E$. The goal is then to approximate p by another distribution p_w which is easy to compute from a set of real-valued parameters w. The *training*

phase in machine learning consists of optimizing the parameter vector w such that p_w is a good approximation of p. The optimization problem to solve is therefore

$$\min_w \quad \Delta(p, p_w). \tag{4.7}$$

Here Δ is some notion of distance in the space of probability distribution functions which, when applied to machine learning, is also called *the loss function*. In our case, the distance is the *softmax* cross entropy as explained here. The softmax function with respect to p is given via

$$\rho(x) = \frac{e^{p(x)}}{\sum_{x \in \mathcal{D}} e^{p(x)}}. \tag{4.8}$$

From this definition, it is obvious that no normalization of the dataset D is needed since softmax already results in a probability distribution function. The cross entropy function

$$H(\pi_1, \pi_2) = H(\pi_1) + D_{KL}(\pi_1 \| \pi_2) = -\sum_x \pi_1(x) \log \pi_2(x) \tag{4.9}$$

is then applied after softmax. This turns (4.7) into

$$\min_w \quad h(w) = H(\rho(p), \rho(p_w)). \tag{4.10}$$

Optimization of the softmax cross-entropy is a common practice in classification problems.

The neural network. A neural network is a directed graph equipped with a random variable assigned to each of its nodes. The elements of the parameter vector w are assigned either to an edge of the graph or a node of the graph (in the former case they are called *weights* and in the latter case they are called *biases*). The role of the neural network in solving (4.10) is to facilitate a gradient descent direction for the vector w in (4.10). This is achieved by imposing the random variables of each node to be a function of the random variables with incoming edges to the former one. The common choice for such a functional relationship is an affine transformation composed with a nonlinear function (called the *activation* function) with an easy to compute derivative. Given every node v of the neural network, we define:

$$X_v = a_v \left(\sum_{u \to v} w_{uv} X_u + w_v \right). \tag{4.11}$$

The simplest activation function is of course the identity. Historically, the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ was the most commonly used activation function and is motivated by its appearance in training restricted Boltzmann machines. By performing a change of variables,

one obtains the trigonometric activation function $\tanh(x)$. These activation functions can cause the learning rate to slow down due to vanishing gradients in the early layers of deep neural networks, and this is the motivation for other proposed activation functions such as the rectified linear unit $\mathrm{relu}(x)$. Design and analysis of activation functions is an important step in machine learning [103, 104, 105].

The first and last layers of the network are known as the *visible* layers and respectively correspond to the input and output data (in our case the tuples $(\mathbf{s}, \mathbf{b}) \in S \times B$ as explained above). Successive applications of Eq. 4.11 restricts the conditional distribution $p_w(\mathbf{b} \,|\, \mathbf{s})$ into a highly nonlinear function $f(w, \mathbf{s}, \mathbf{b})$, for which the derivatives with respect to the parameters w are easy to compute via the chain rule. We may therefore devise a gradient descent method for solving Eq. 4.10 by successive choices of descent directions starting from the deep layers and iterating towards the input nodes. In machine learning, this process is known as *back-propagation*.

Remark. *The softmax function (Eq. 4.8) is in other words the activation function between the last two layers of the neural network.*

Layouts. Although deep learning restricts the approximation of $p_w(\mathbf{b}|\mathbf{s})$ to functions of the form $f(w, \mathbf{s}, \mathbf{b})$ as explained above, the latter has tremendous representation power, specially given the freedom in choice of the layout of the neural network. Designing efficient layouts for various applications is an artful and challenging area of research in machine learning. In this chapter, we discuss three such layouts and justify their usage for the purposes of our deep neural decoding.

Feedforward neural network. By this we mean a multi-layer neural network consisting of consecutive layers, each layer fully connected to the next one. Therefore, the underlying undirected subgraph of the neural network consisting of the neurons of two consecutive layers is a complete bipartite graph. In the case that the neural network only consists of the input and output layers (with no hidden layers), the network is a generalization of logistic regression (known as the softmax regression method).

Recurrent neural network (RNN). RNNs have performed incredibly well in speech recognition and natural language processing tasks [106, 107, 108, 109]. The network is designed to resemble a temporal sequence of input data, with each input layer connecting to the rest of the network at a corresponding temporal epoch. The *hidden cell* of the network could be as simple as a single feedforward layer or more complicated. Much of the success of RNNs is based on peculiar designs of the hidden cell such as the Long-Short Term Memory (LSTM) unit as proposed in [110].

Convolutional neural network (CNN). CNNs have been successfully used in image processing tasks [111, 112]. The network is designed to take advantage of local properties of an image by probing a kernel across the input image and calculating the cross-correlation of the kernel vector with the image. By applying multiple kernels, a layer of *features* is constructed. The features can then be post-processed via downsizing (called *max-pooling*) or by yet other feedforward neural networks.

In sections 4.3.2 and 4.3.3, we present further details about applications of these neural networks to the error-decoding task.

Stochastic gradient descent. Since the cross-entropy in Eq. 4.9 is calculated by a weighted sum over all events $x \in \mathcal{D}$, it is impractical to exactly calculate it or its derivatives as needed for backpropagation. Instead, one may choose only a single sample $x = (\mathbf{s}, \mathbf{b})$ as a representative of the entire $\mathcal{D}$ in every iteration. Of course, this is a poor approximation of the true gradient but one hopes that the occurrences of the samples according to the true distribution would allow for the descent method to 'average out' over many iterations. This method is known as stochastic gradient descent (SGD) or *online learning*. We refer the reader to [113] and [114] and the references therein for proofs of convergences and convergence rates of online learning. In practice, a middle ground between passing through the entire dataset and sampling a single example is observed to perform better for machine learning tasks [103]: we fix a batch size and in every iteration average over a batch of the samples of this size. We call this approach *batch gradient descent* (also called mini-batch gradient descent for better contrast). The result is an update rule for the parameter vector of the form $w_{t+1} \leftarrow w_t + \Delta_t$ where Δ_t is calculated as

$$\Delta_t = -\eta_t \nabla_{t-1},$$

for some step size η_t, where $\nabla_{t-1} = \nabla_{w_{t-1}} \tilde{h}(w_{t-1})$ to simplify the notation. Here $\tilde{h}$ is an approximation of h in (4.10) by the partial sum over the training batch. Finding a good schedule for η_t can be a challenging engineering task that will be addressed in Sec. 4.3.1. Depending on the optimization landscape, SGD might require extremely large numbers of iterations for convergence. One way to improve the convergence rate of SGD is to add a *momentum* term [115]:

$$\Delta_t = p\Delta_{t-1} - \eta_t \nabla_{t-1}.$$

On the other hand, it is convenient to have the schedule of η_t be determined through the training by a heuristic algorithm that adapts to the frequency of every event. The method AdaGrad was developed to allow much larger updates for infrequent samples [116]:

$$\Delta_t = -\operatorname{diag}\left(\frac{\eta}{\sqrt{\Sigma_{ti} + \varepsilon}}\right) \nabla_{t-1}.$$

Here Σ_{ti} is the sum of the squares of all previous values of the i-th entry of the gradient. The quantity ε is a small (e.g. 10^{-8}) smoothening factor in order to avoid dividing by zero. The denominator in this formula is called the *root mean squared* (RMS). An important advantage of AdaGrad is the fact that the freedom in the choice of the step-size schedule is restricted to choosing one parameter η, which is called *the learning rate*.

Finally RMSProp is an improvement on AdaGrad in order to slow down the aggressive vanishing rate of the gradients in AdaGrad [117]. This is achieved by adding a momentum term to the root mean squared:

$$\operatorname{diag}(\Sigma_t) = p \operatorname{diag}(\Sigma_{t-1}) + (1-p)\nabla_{t-1}\nabla_{t-1}^T.$$

Hyperparameter tuning. From the above exposition, it is apparent that a machine learning framework involves many algorithms and design choices. The performance of the framework depends on optimal and consistent choices of the free parameters of each piece, the *hyperparameters*. For example, while a learning rate of 10^{-3} might be customary for a small dataset such as that of MNIST digit recognition, it might be a good choice for a small feedforward network and a bad choice for the RNN used in our problem scenario. In our case, the hyperparameters include the decay rate, the learning rate, the momentum in RMSProp, the number of hidden nodes in each layer of the network, the number of hidden layers and filters, and some categorical variables such as the activation function of each layer, the choice of having peepholes or not in the RNN.

It would be desirable if a metaheuristic can find appropriate choices of hyperparameters. The challenges are

1. Costly function evaluation: the only way to know if a set of hyperparameters is appropriate for the deep learning framework, is to run the deep learning algorithm with these parameters;

2. Lack of a gradient-based solution: the solution of the deep learning framework does not have a known functional dependence on the hyperparameters. Therefore, the metaheuristic has no knowledge of a steepest descent direction.

It is therefore required for the metaheuristic to be (1) sample efficient and (2) gradient-free. Having a good metaheuristic as such is extremely desirable, since:

1. The performance of the ML framework might be more sensitive to some parameters than to others. It is desirable for the metaheuristic to identify this.

2. Compatibility of the parameters: leaving the hypertuning job to a researcher can lead to search in very specific regimes of hyperparameters that are expected to be good choices individually but not in combination.

3. Objectivity of the result: a researcher might spend more time tuning the parameters of their proposal than on a competing algorithm. If the same metaheuristic is used to tune various networks, such as feedforward networks, RNNs and CNNs, the result would be a reliable comparison between all suggestions.

Bayesian optimization. Bayesian optimization [118] is a nonlinear optimization algorithm that associates a surrogate model to its objective function and modifies it at every function evaluation. It then uses this surrogate model to decide which point to explore next for a better objective value [119]. Bayesian optimization is a good candidate for hypertuning as it is sample efficient and can perform well for multi-modal functions without a closed formula. A disadvantage of Bayesian optimization to keep in mind is that it relies on design choices and parameters of its own that can affect its performance in a hyperparameter search.

4.3.2 Steane and Knill EC deep neural decoder for the CNOT-exRec

The simplest deep neural decoder for any dataset is a feedforward network with none or many hidden layers, each layer fully connected to the next one. The input layer receives the bit strings of X and Z syndromes. And the output layer corresponds to the X and Z recovery operators on the physical qubits of the code. Since multiple physical qubits might be used to encode a single logical operator, a better choice is for the output layer to encode whether an auxiliary (but efficient) decoding scheme is causing logical faults or not. The goal would be to predict such logical faults by the deep neural decoder and when the deep neural decoder predicts such a fault, we will impose a logical Pauli operator after the recovery suggested by the auxiliary decoder. The 1-hot encoding in two bits, 10 and 01, respectively stand for I and X for the X-errors, and it stands for I and Z for the Z errors.

From our early experiments it became apparent that it is beneficial to half separate X and Z neural networks that share a loss function, that is the sum of the soft-max cross entropies of the two networks. Fig. 4.9 shows the schematics of such a feedforward network.

The CNOT-exRec RNN. In the case of the CNOT-exRec, the leading EC rounds have temporal precedence to the trailing EC rounds. Therefore a plausible design choice for

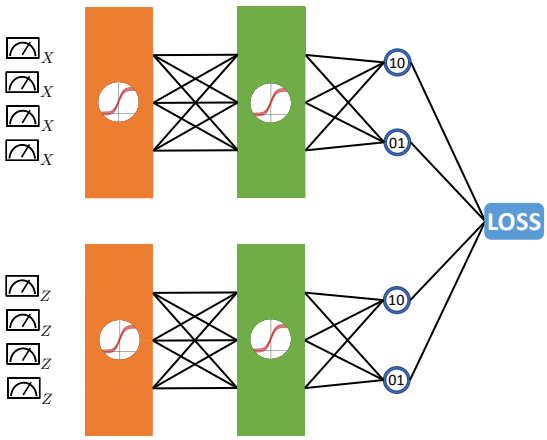


Figure 4.9: Schematics of a feedforward network consisting of disjoint X and Z networks. There may be none, one or multiple hidden layers with different activation functions. The output layers correspond to logical I- and X-errors for the X network and to logical I- and Z-errors for the Z network. The activation function of the last layer before the error layer is the identity since in the softmax cross entropy loss function, the activation (by softmax) is already included.

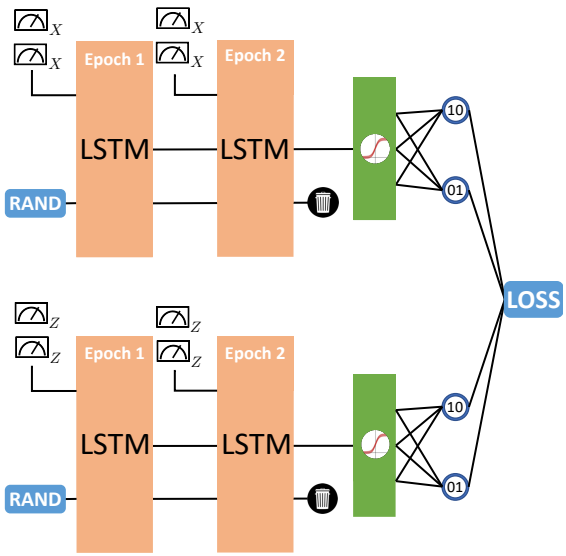


Figure 4.10: Schematics of a network consisting of two disjoint X and Z RNNs. Each RNN receives the syndromes of leading and trailing EC rounds as inputs for two epochs of its LSTM unit. The internal state of the first copy is initialized randomly and the internal state of the last copy is garbage-collected. The hidden state of the last copy of the LSTM unit is then fully connected to a hidden layer with user-defined activation function. This hidden unit is then fully connected to output nodes denoted by 01 and 10 which are respectively the one-hot encoding of the prediction as to whether an X-recovery or a Z-recovery operation is needed on the output qubits from the CNOT-exRec. The loss function is the sum of the loss functions of the two networks.

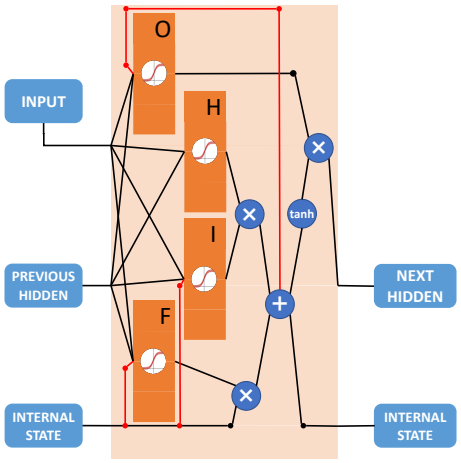


Figure 4.11: Schematics of a long-short term memory (LSTM) cell. Without the red circuits, this neural network is called a simple LSTM unit. The red circuit is called peepholes. An LSTM cell with peepholes can outperform a simple LSTM cell in some tasks. There are four hidden layers with user-defined activation functions in an LSTM unit known as the forget layer (F), input layer (I), hidden layer (H) and the output layer (O). There are four 2 to 1 logical gates in the unit that depending on the sign written on them applies an element-wise operation between the vectors fed into the logical gates. There is also a 1 to 1 logical gate that applies an element-wise *tanh* function on its input vector. The *internal state* of an LSTM unit serves as the backbone of a sequence of replications of the LSTM unit. The role of the internal state is to capture temporal features of the sequence of input data.

the deep neural decoder would be to employ an RNN with two iterations on the hidden cell. In the first iteration, the syndrome data from the leading EC rounds are provided and in the second iteration the syndrome data from the trailing EC rounds are provided. A demonstration of this network is given in Fig. 4.10.

The hidden cell of the RNN may be an LSTM, or an LSTM with peepholes as shown in Fig. 4.11. An LSTM cell consists of an *internal state* which is a vector in charge of carrying temporal information through the unrolling of the LSTM cell in time epochs. There are 4 hidden layers. The layer H is the 'actual' hidden layer including the input data of the current epoch with the previous hidden layer from the previous epoch. The activation of H is usually tanh. The 'input' layer I is responsible for learning to be a bottleneck on how important the new input is, and the 'forget' layer F is responsible for creating a bottleneck on how much to forget about the previous epochs. Finally the 'output' layer O

is responsible for creating a bottleneck on how much data is passed through from the new internal state to the new hidden layer. The peepholes in Fig. 4.11 allow the internal state to also contribute in the hidden layers F, I and O.

4.3.3 Surface code deep neural decoder

Other than the multi-layer feedforward network of Fig. 4.9, there are two other reasonable designs for a deep neural network when applied to the surface code.

The surface code RNN. In the fault-tolerant scheme of the rotated surface code, multiple rounds of error correction are done in a sequence as explained in Sec. 4.2.1. It is therefore encouraging to consider an RNN with inputs as syndromes of the consecutive EC rounds. The network looks similar to that of Fig. 4.10 except that the number of epochs is equal to the maximum number of EC rounds. In particular, the fault tolerant scheme for the distance-three rotated surface code consists of three EC rounds. In the case of the distance-five surface code, the maximum number of EC rounds through the algorithm of Sec. 4.2.1 is six. If the rounds of EC stop earlier, then the temporal input sequence of syndrome strings is padded by repeating the last syndrome string. As an example, if after three rounds the fault tolerant scheme terminates, then the input syndromes of epochs three to six of the RNN are all identical and equal to the third syndrome string.

The surface code CNN. The errors, syndromes and recovery operators of the surface code are locally affected by each other. It is therefore suggestive to treat the syndromes of the surface code as a 2-dimensional array, the same way pixels of an image are treated in image processing tasks. The multiple rounds of EC would account for a sequence of such images, an *animation*. Therefore a 3-dimensional CNN appears to be appropriate. This means that the kernels of the convolutions are also 3-dimensional, probing the animation along the two axes of each image and also along the third axis representing time.

Through our test-driven design, it became obvious that treating the X and Z syndromes as channels of the same 3-dimensional input animation is not a good choice. Instead, the X and Z syndromes should be treated as disjoint inputs of disjoint networks which in the end contribute to the same loss function. Notice that in the case of the distance-five rotated surface code, the X network receives a 3D input of dimensions $3 \times 4 \times 6$ and the Z network receives a 3D input of dimensions $4 \times 3 \times 6$. To create edge features, the inputs were padded outwards *symmetrically*, i.e. with the same binary values as their adjacent bits. This changes the input dimensions to $4 \times 5 \times 6$ and $5 \times 4 \times 6$ respectively for the X and Z animations. Via similar experiments, we realized that two convolutional layers do

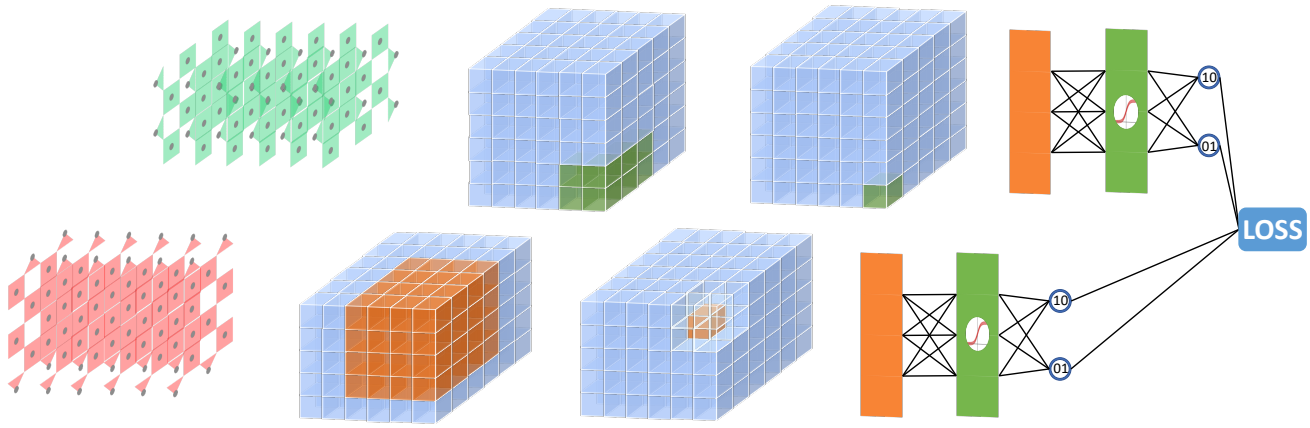


Figure 4.12: Schematics of a deep neural decoder for the distance-five rotated surface code. The network consists of two disjoint neural networks contributing to the same loss function via softmax cross entropy. Each neural network consists of two layers of 3D CNNs. The first layer consists of a number of filters, each filter performing a convolution of a $3 \times 3 \times 3$ kernel by the input syndromes. The second 3D CNN layer uses $4 \times 4 \times 4$ kernels. The colored boxes demonstrate how each layer is padded in order for the size of the 3D layers to be preserved. When the kernel dimension is even for instance, the padding from the top and left are of size 1, and the padding from the bottom and right are of size 2.

a better job in capturing patterns in the syndromes data. The first convolutional layer is probed by a $3 \times 3 \times 3$ kernel, and the second layer is probed by a $4 \times 4 \times 4$ kernel. After convolutional layers, a fully connected feedforward layer with dropouts and relu activations is applied to the extracted features and then the softmax cross-entropy is measured. The schematic of such a neural network is depicted in Fig. 4.12.

4.4 Numerical experiments

In the experimental results reported in this section, multiple data sets were generated by various choices of physical error rates ranging between $p = 1.0 \times 10^{-4}$ to $p = 2.0 \times 10^{-3}$. Every data set consisted of simulating the circuit-level depolarizing channel (see Sec. 3.1.1 for a detailed description of the noise model) for the corresponding circuit, and included the syndrome and resulting error bit strings in the data set. Note that the error strings are only used as part of the simulation to compute the vector $\mathbf{b}$ of logical faults. In an actual experiment, $\mathbf{b}$ would be given directly (see the discussion above Sec. 4.3.1). We excluded

the cases where both the syndrome and error strings were all zeros. The simulation was continued until a target number of *non-zero* training samples were gathered. The target size of the training data set was chosen as 2×10^6 for distance-three codes, and 2×10^7 for distance-five codes.

Hypertuning was performed with the help of BayesOpt [119]. In every hypertuning experiment, each query consisted of a full round of training the deep learning network on 90% of the entire dataset and cross-validating on the remaining 10%. It is important to add randomness to the selection of the training and cross-validating data sets so that the hyperparameters do not get tuned for a fixed choice of data entries. To this aim, we uniformly randomly choose an initial element in the entire data set, take the 90% of the dataset starting from that initial element (in a cyclic fashion) as the training set, and the following 10% as the test dataset.

The cross-entropy of the test set is returned as the final outcome of one query made by the hypertuning engine. For all hypertuning experiments, 10 initial queries were performed via Latin hypercube sampling. After the initial queries, 50 iterations of hypertuning were performed.

For each fault-tolerant error correction scheme, hypertuning was performed on only a single data set (i.e. only for one of the physical error rates). A more meticulous investigation may consist of hypertuning for each individual physical error rate separately but we avoided that, since we empirically observed that the results are independent of the choice of hypertuning data set. At any rate, the data chosen for distance-three codes was the one corresponding to $p = 4 \times 10^{-4}$. For the distance-five rotated surface code, $p = 6.0 \times 10^{-4}$ and for the 19-qubit color code using Steane and Knill-EC, $p = 1.0 \times 10^{-3}$ were chosen for hypertuning.

Hyperparameters chosen from this step were used identically for training all other data sets. For every data set (i.e. every choice of physical fault rate p), the deep learning experiment was run 10 times and in the diagrams reported below the average and standard deviations are reported as points and error bars. In every one of the 10 runs, the training was done on 90% of a data set, and cross validation was done on the remaining 10%. All the machine learning experiments were implemented in Python 2.7 using TensorFlow 1.4[120] on top of CUDA 9.0 running installed on TitanXp and TitanV GPUs produced by NVIDIA[121].

All experiments are reported in Fig. 4.13–Fig. 4.24. Before continuing with detailed information on each experiment, we refer the reader to Tab. 4.2 where we provide the largest ratios of the pseudo-thresholds obtained using a neural network decoder to pseudo-thresholds obtained from the bare lookup table decoders of each fault-tolerant protocol

considered in this chapter.

FTEC	Lookup	DND	Ratio
$d = 3$ Steane	$p_{\text{th}} = 2.10 \times 10^{-4}$	$p_{\text{th}} = 3.98 \times 10^{-4}$	1.90
$d = 5$ Steane	$p_{\text{th}} = 1.43 \times 10^{-3}$	$p_{\text{th}} = 2.17 \times 10^{-3}$	1.52
$d = 3$ Knill	$p_{\text{th}} = 1.76 \times 10^{-4}$	$p_{\text{th}} = 2.22 \times 10^{-4}$	1.26
$d = 5$ Knill	$p_{\text{th}} = 1.34 \times 10^{-3}$	$p_{\text{th}} = 1.54 \times 10^{-3}$	1.15
$d = 3$ Surface code	$p_{\text{th}} = 2.57 \times 10^{-4}$	$p_{\text{th}} = 3.18 \times 10^{-4}$	1.24
$d = 5$ Surface code	$p_{\text{th}} = 5.82 \times 10^{-4}$	$p_{\text{th}} = 7.11 \times 10^{-4}$	1.22

Table 4.2: Pseudo-thresholds for the 6 fault-tolerant error correction protocols considered in the experiments. The second column corresponds to the highest pseudo-thresholds obtained from a bare lookup table decoder whereas the third column gives the highest pseudo-thresholds using neural network decoders. The last column corresponds to the ratio between the pseudo-thresholds obtained from the best neural network decoders and the lookup table decoders.

parameter	lower bound	upper bound
decay rate	0.0	$1.0 - 10^{-6.0}$
momentum	0.0	$1.0 - 10^{-6.0}$
learning rate	$10^{-5.0}$	$10^{-1.0}$
initial std	$10^{-3.0}$	$10^{-1.0}$
num hiddens	100	1000

Table 4.3: Bayesian optimization parameters for the CNOT-exRec of the $[\![7, 1, 3]\!]$ code using Steane and Knill-EC and the distance-three rotated surface code. Here the decay rate, momentum and learning rate pertain to the parameters of RMSProp. The row 'initial std' refers to the standard deviation of the initial weights in the neural networks, the mean of the weights was set to zero. The initial biases of the neural networks were set to zero. The row 'num hiddens' refers to the number of hidden nodes in the layers of neural network. This parameter is optimized for each layer of the neural network independently (e.g. for a feedforward network consisting of 3 hidden layers, there are 3 numbers of hidden nodes to be tuned). For an RNN this number indicates the number of hidden nodes in every one of the 4 hidden layers of the LSTM unit (all of the same size).

Steane-EC CNOT-exRec for the $[\![7, 1, 3]\!]$ code. The considered continuous and integer hyperparameters are given in Tab. 4.3.

We also tuned over the categorical parameters of Tab. 4.4. The categorical parameters

parameter	values
activation functions	relu, tanh, sigmoid, identity
numbers of hidden layers	0, 1, 2, ...

Table 4.4: Categorical hyperparameters. Optimizations over activation functions was only performed for the distance-three Steane code. Since rectified linear units showed better results, we committed to this choice for all other error correction schemes. However, for the second categorical hyperparameter (the numbers of hidden layers), the search was performed for all error correction schemes separately and was stopped at the numbers of hidden layers where the improvements in the results discontinued.

are tuned via grid-search. We observed that for all choices of neural networks (feedforward networks with various numbers of hidden layers and recurrent neural networks with or without peepholes), the rectified linear unit in the hidden layers and identity for the last layer resulted in the best performance. We accepted this choice of activation functions in all other experiments without repeating a grid-search.

Fig. 4.13 and Fig. 4.14 compare the performance of the feedforward and RNN decoders that respectively use the lookup table and naive-decoder as their underlying decoders, respectively referred to as LU-based deep neural decoders (LU-DND) and PE-based deep neural decoders (PE-DND). We use PE since naive-decoders correct by applying pure errors. We observe that softmax regression (i.e. zero hidden layers) is enough to get results on par with the lookup table method in the LU-based training method, this was not the case in the PE-based method. The RNNs perform well but they are outperformed by two-hidden-layer feedforward networks. Additional hidden layers improve the results in deep learning. However, since this is an expense for a cross-entropy optimization in higher dimensions, the training of deeper networks is significantly more challenging. This trade-off is the reason the feedforward networks improve up to two hidden layers, but passing to three and higher numbers of hidden layers gave worse results (not reported in these diagrams).

We finally observe that PE-DND with even a single hidden layer feedforward network is almost on par with the LU-DND with two hidden layers. This is an impressive result given the fact that a table of pure errors grows linearly in the number of syndromes, but a lookup table grows exponentially. We believe this is a result of the fact that logical faults are much more likely to occur when using recovery operators which only consist of products of pure-errors, the training sets are less sparse and therefore deep learning is able to capture more patterns for the classification task at hand.

Knill-EC CNOT-exRec for the $[\![7,1,3]\!]$ code. The hypertuning of continuous variables was done using the same bounds as in Tab. 4.3. Fig. 4.15 and Fig. 4.16 respectively show the results of LU-DND and PE-DND methods. The best results were obtained by feedforward networks with respectively 3 and 2 hidden layers, in both cases slightly outperforming RNNs.

Distance-three rotated surface code. Similar to the previous distance-three codes, we compared using RNNs with feedforward networks with multiple hidden layers. We observed that the feedfoward network with a single hidden layer achieves the best performance and RNNs do not improve the results. Also consistent with the distance-three CNOT- exRec results, the PE-based DND can perform as well as the LU-based one (and slightly improves upon it). Results of these experiments are reported in Fig. 4.17 and Fig. 4.18.

Steane-EC CNOT-exRec for the $[\![19,1,5]\!]$ code. As the size of the input and output layers of DNNs grow, the ranges of the optimal hyperparameters change. For the distance-five Steane exRec circuit applied to the $[\![19,1,5]\!]$ color code, the considered hyperparameter ranges (allowing smaller orders of magnitudes for the initial weight standard deviations and much smaller learning rates) are given in Tab. 4.5.

parameter	**lower bound**	**upper bound**
decay rate	0.0	$1.0 - 10^{-6.0}$
momentum	0.0	$1.0 - 10^{-6.0}$
learning rate	$10^{-7.0}$	$10^{-3.0}$
initial std	$10^{-5.0}$	$10^{-3.0}$
num hiddens	100	1000

Table 4.5: Bayesian optimization parameters for $d = 5$ Steane and Knill CNOT-exRecs. Given the larger size of the training sets and longer input strings, for these datasets, smaller orders of magnitudes for the initial weight standard deviations and much smaller learning rates were explored.

Fig. 4.19 and Fig. 4.20 show that the PE-DNDs has a slightly harder time with pattern recognition compared to the LU-DNDs. Nevertheless, both methods significantly improve the pseudo-thresholds of the distance-five Steane-EC scheme, with no advantage obtained from using an RNN over a 2-hidden layer feedforward network. In both experiments, the 3-hidden layer feedforward networks also did not result any improvements.

Knill-EC CNOT-exRec for the $[\![19,1,5]\!]$ code. The hyperparameter ranges used for hypertuning were similar to those obtained for the Steane-EC CNOT-exRec applied to

the $[\![19, 1, 5]\!]$ code. Given the effectiveness of the 2-hidden layer feedforward network, this feedforward neural network was chosen for the Knill exRec $d = 5$ experiment. We see a similar improvement on the pseudo-threshold of the error correction scheme using either of LU-DND and PE-DND.

Distance-five rotated surface code. For rotated surface codes, we only considered numerical simulations using one EC rather than the full exRec. This choice was made to be consistent with previous analyses of the surface codes performance.

The hyperparameter ranges used for hypertuning the feedforward neural networks were chosen according to Tab. 4.6.

parameter	**lower bound**	**upper bound**
decay rate	0.0	$1.0 - 10^{-6.0}$
momentum	0.0	$1.0 - 10^{-6.0}$
learning rate	$10^{-6.0}$	$10^{-2.0}$
initial std	$10^{-6.0}$	$10^{-2.0}$
num hiddens	100	1000

Table 4.6: Bayesian optimization parameters for the distance-five rotated surface code. The parameter search is in a slightly tighter domain than in the case of the distance-five Knill and Steane CNOT-exRecs in view of the empirical initial tests performed.

As explained in the previous section, a CNN engineered appropriately could be a viable layout design for large surface codes. Besides previous hyperparameters, we now also need to tune the number of filters, and drop-out rate. A summary of the settings for Bayesian optimization are given in Tab. 4.7.

We compare the PE-based and LU-based feedforward networks with the CNN proposed in Sec. 4.3.3. Fig. 4.23 and Fig. 4.24 show that feedforward networks with 2 hidden layers result in significant improvements both using the PE-based and LU-based DNDs. The 3D-CNN is slightly improving the results of the feedforward network in PE-DND but is only slightly better than the lookup table based method in the LU-DND case. The best overall performance is obtained by using a feedfoward network with 2 hidden layers for the LU-DND. A slightly less performant result can also be obtained if the PE-DND method is used in conjunction with either of the 2-hidden layer feedforward network or the 3D convolutional neural network.

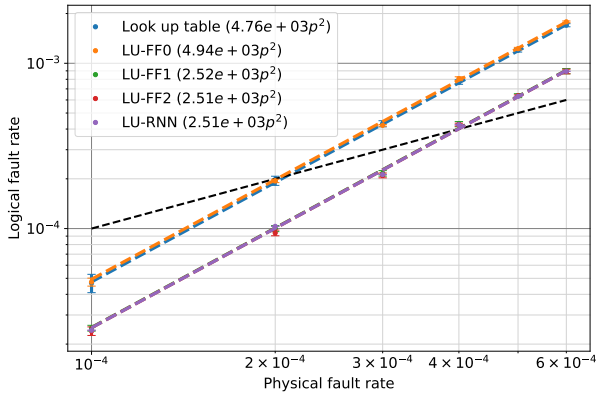


Figure 4.13: LU-DND for the distance-three Steane CNOT-exRec.

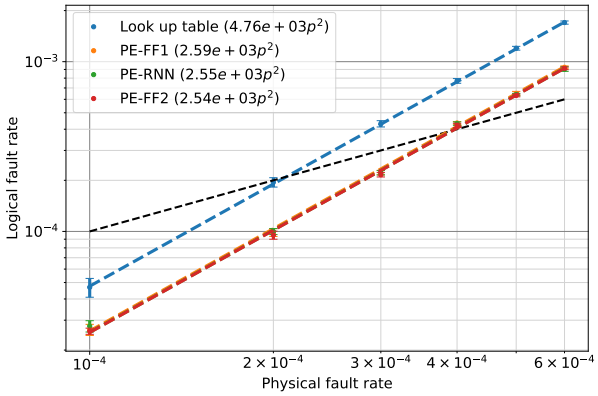


Figure 4.14: PE-DND for the distance-three Steane CNOT-exRec.

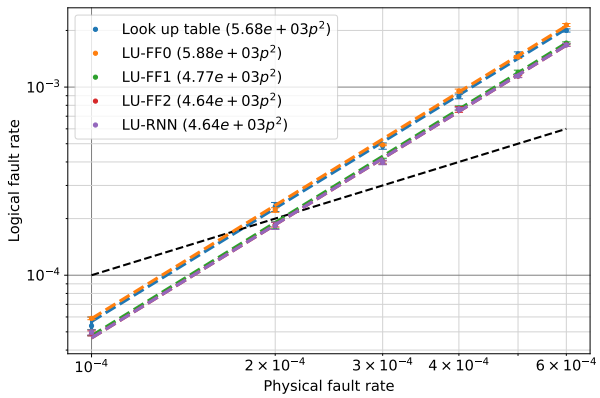


Figure 4.15: LU-DND for the distance-three Knill CNOT-exRec.

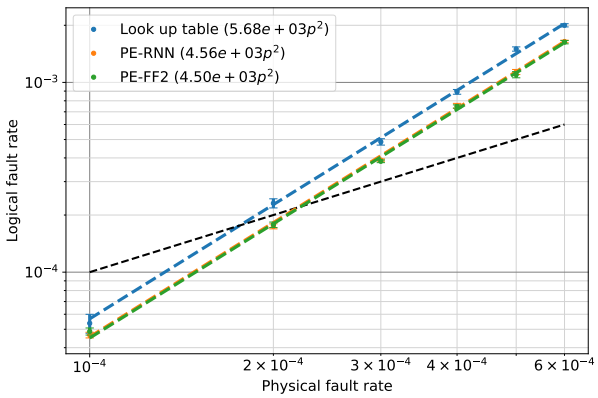


Figure 4.16: PE-DND for the distance-three Knill CNOT-exRec.

In Fig. 4.13–Fig. 4.16 each data point has the height on the vertical axis being the average of 10 logical fault rates collected for each physical fault rate p specified on the horizontal axis. Error bars represent the standard deviation from these average values. For each DND-based decoder, the curve-fitting method used is a non-linear least square fitting between the average logical fault rates as a function of the physical fault rates, and a quadratic monomial.

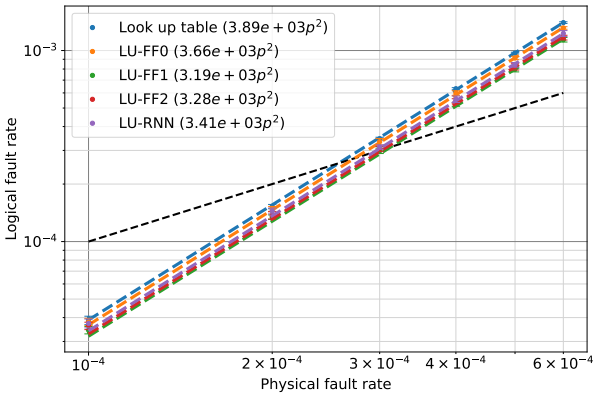


Figure 4.17: LU-DND for the distance-three surface code.

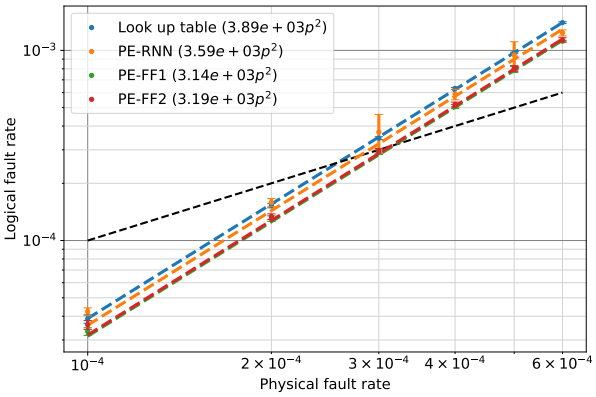


Figure 4.18: PE-DND for the distance-five surface code.

In Fig. 4.17–Fig. 4.18 each data point has the height on the vertical axis being the average of 10 logical fault rates collected for each physical fault rate p specified on the horizontal axis. Error bars represent the standard deviation from these average values. For each DND-based decoder, the curve-fitting method used is a non-linear least square fitting between the average logical fault rates as a function of the physical fault rates, and a quadratic monomial.

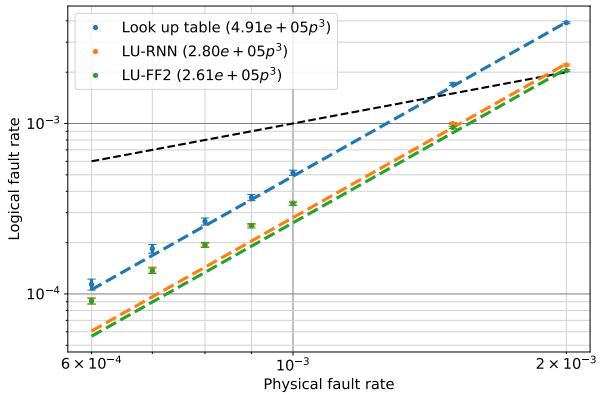


Figure 4.19: LU-DND for the distance-five Steane CNOT-exRec.

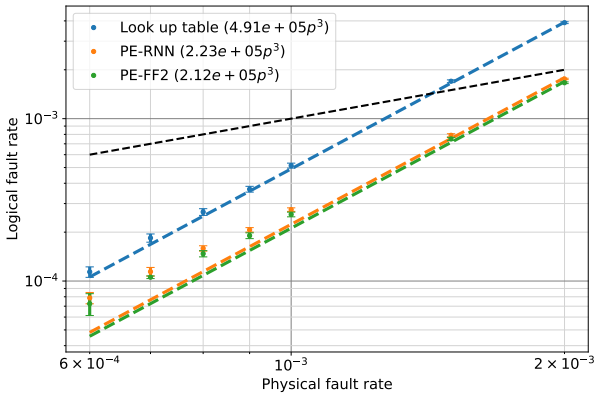


Figure 4.20: PE-DND for the distance-five Steane CNOT-exRec.

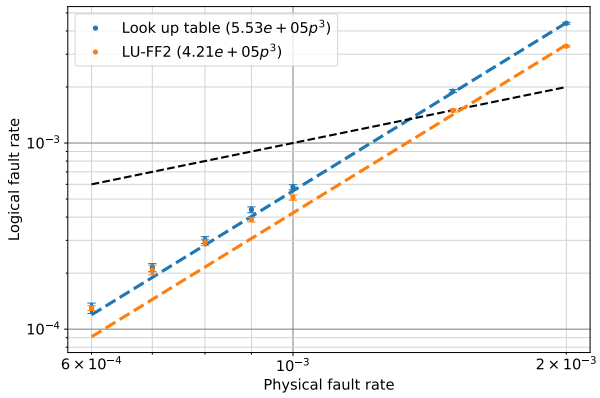


Figure 4.21: LU-DND for the distance-five Knill CNOT-exRec.

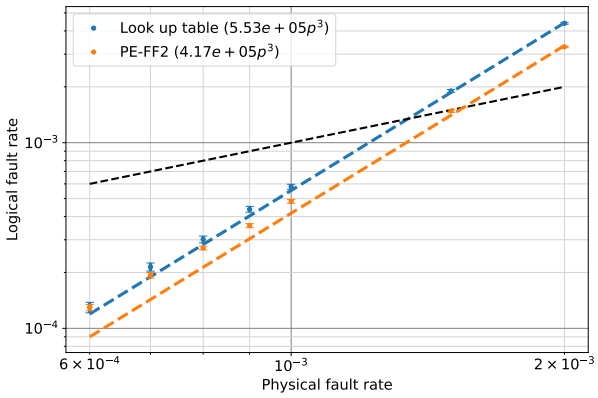


Figure 4.22: PE-DND for the distance-five Knill CNOT-exRec.

In Fig. 4.19–Fig. 4.22 data points, averages and error bars are obtaines in a similar fashion to Fig. 4.13–Fig. 4.18. The curve-fitting method is also a non-linear least square method, this time fitting a cubic monomial through the data points.

parameter	lower bound	upper bound
decay rate	0.0	$1.0 - 10^{-6.0}$
momentum	0.0	$1.0 - 10^{-6.0}$
learning rate	$10^{-6.0}$	$10^{-2.0}$
initial std	$10^{-6.0}$	$10^{-2.0}$
num hiddens	100	1000
keep rate	0.0	1.0
num filters	5	10

Table 4.7: Bayesian optimization parameters for a 3-dimensional CNN. The filters were fixed to be $3\times3\times3$ and $4\times4\times4$ but their quantities were tuned. Since CNNs are larger and deeper than other networks considered in this chapter, they are more prone to vanishing gradients. Therefore it is beneficial to consider drop-outs in the hidden layer after feature extraction. The hyperparameter corresponding to drop-outs is 'keep rate' allowing more drop-outs when it is smaller.

4.5 Performance analysis

In this section we consider the efficiency of the deep neural decoders in comparison to the lookup table decoders described in Sec. 4.2.1 and Sec. 4.2.2. The size of a lookup table grows exponentially in the number of syndromes therefore making lookup table based decoding intractable as the codes grow. However, it is important to note that as long as the size of the lookup table allows for storage of the entire table in memory, as described in Sec. 4.2.5, the lookup from an array or a hash table happens effectively in $O(1)$ time. Therefore a lookup table based decoding scheme would be the most efficient decoder by far. A similar approach to a lookup table decoder is possible by making an inference mapping from all the possible input strings of a trained neural decoder. This method is discussed in Sec. 4.5.1. For larger codes, neither a lookup table decoder, nor an inference mapping decoder is an option due to exponentially growing memory usage.

More complicated decoders such as minimum weight perfect matching can be extremely inefficient solutions for decoding despite polynomial asymptotic complexity. With gates operating at 100Mhz (that is 10ns gate times) [122], which is much faster than the state of the art[1], the simplest quantum algorithms foreseen to run on near term devices would

[1]In fact, existing prototypes of quantum computers have much longer gate delays. Typical gate times in a superconducting system are 130ns for single-qubit and $250 - 450$ns for 2-qubit gates. For a trapped-ion system, gate times are even longer, reaching 20μs for single-qubit gates and 250μs for 2-qubit gates [123, 124].

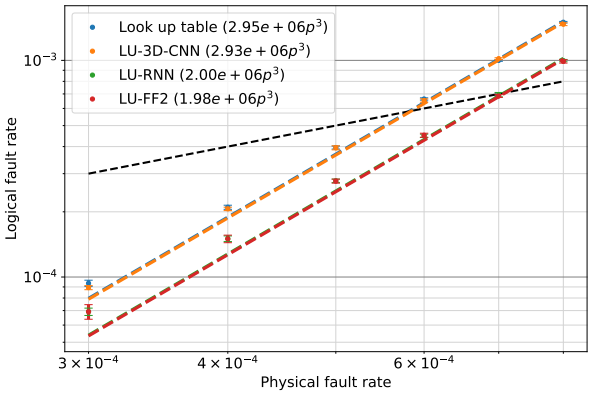


Figure 4.23: LU-DND for the distance-five surface code.

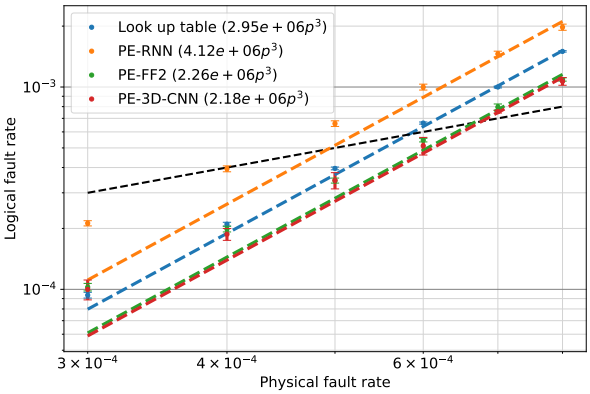


Figure 4.24: PE-DND for the distance-five surface code.

In Fig. 4.23–Fig. 4.24 data points, averages and error bars are obtaines in a similar fashion to Fig. 4.13–Fig. 4.18. The curve-fitting method is also a non-linear least square method, this time fitting a cubic monomial through the data points.

require days of runtime on the system [125]. With the above gate times, the CNOT-exRec using Steane and Knill EC units as well as the multiple rounds of EC for surface codes would take as small as a hundred nanoseconds. In order to perform active error correction, we require classical decoding times to be implemented on (at worst) a comparable time scale as the EC units, and therefore merely a complexity theoretic analysis of a decoding algorithm is not enough for making it a viable solution. Alternatively, given a trained DND, inference of new recovery operations from it is a simple algorithm requiring a sequence of highly parallelizable matrix multiplications. We will discuss this approach in Sec. 4.5.2 and Sec. 4.5.3.

4.5.1 Inference mapping from a neural decoder

For codes of arbitrary size, the most time-performant way to use a deep neural decoder is to create an array of all inputs and outputs of the DNN in the test mode (i.e. an inference map which stores all possible syndromes obtained from an EC unit and assigns each combination to a recovery operator[2]). This is possible for distance-three fault-tolerant

[2]For the CNOT-exRec, the inference map would map syndromes from all four EC units to a recovery operator. For the surface code, the inference map would map syndromes measured in each round to a

EC schemes such as Steane, Knill and surface codes (as well as other topological schemes such as those used for color codes). For all these codes, the memory required to store the inference map is 2.10 megabytes. This method is not feasible for larger distance codes. For the Knill and Steane-EC schemes applied to the $[\![19, 1, 5]\!]$ color code, the memory required is 590 exabytes and for the distance-five rotated surface code it is 2.79×10^{24} exabytes.

4.5.2 Fast inference from a trained neural network

An advantage of a deep neural decoder is that the complications of decoding are to be dealt with in the training mode of the neural network. The trained network is then used to suggest recovery operations. The usage of the neural network in this passive step, i.e. without further training, is called the *inference mode*. Once the neural network is trained, usage of it in the inference mode requires only a sequence of few simple arithmetic operations between the assigned value of its input nodes and the trained weights. This makes inference an extremely simple algorithm and therefore a great candidate for usage as a decoder while the quantum algorithm is proceeding.

However, even for an algorithm as simple as inference, further hardware and software optimization is required. For example, [83] predicts that on an FPGA (field-programmable gate array) every inference from a single layer feedforward network would take as long as 800ns. This is with the optimistic assumption that float-point arithmetic (in 32 and 64-bit precision) takes 2.5 to 5 nanoseconds and only considering a single layer feedforward network.

In this section, we consider alternative optimization techniques for fast inference. We will consider a feedforward network with two hidden layers given their promising performance in our experiments.

Network quantization. Fortunately, quantum error correction is not the only place where fast inference is critical. Search engines, voice and speech recognition, image recognition, image tagging, and many more applications of machine learning are nowadays critical functions of smart phones and many other digital devices. As the usage grows, the need for efficient inference from the trained models of these applications grow. It is also convenient to move such inference procedures to the usage platforms (e.g. the users smart phones and other digital devices) than merely a cloud based inference via a data centre. Recent efforts in high performance computing has focused on fabricating ASICs (Application Specific Integrated Circuits) specifically for inference from neural networks. Google's TPU (Tensor

recovery operator.

Processing Unit) [126] is being used for inference in Google Search, Google Photos and in DeepMind's AlphaGo against one of the the world's top Go player, Lee Sedol.

It is claimed that the reduction in precision of a trained neural network from 32-bit float point precision in weights, biases, and arithmetic operations, to only 8-bit fixed point preserves the quality of inference from trained models [127]. This procedure is called *network quantization*. There is no mathematical reason to believe that the inference quality should hold up under network quantization. However, the intuitive explanation has been that although the training mode is very sensitive to small variations of parameters and hyperparameters, and fluctuations of the high precision weights of the network in individual iterations of training is very small, the resulting trained network is in principle robust to noise in data and weights.

The challenge in our case is that in quantum error correction, the input data is already at the lowest possible precision (each neuron attains 0 or 1, therefore only using a single bit). Furthermore, an error in the input neurons results in moving from one input syndrome to a completely different one (for instance, as opposed to moving from a high resolution picture to a low resolution, or poorly communicated one in an image processing task). We therefore see the need to experimentally verify whether network quantization is a viable approach to high-performance inference from a DND.

Fig. 4.25 demonstrates an experiment to validate network quantization on a trained DND. Using 32-bit float-point precision, the results of Fig. 4.13 show that the trained DND improves the logical failure rate from 1.95×10^{-4} obtained by lookup table methods to 9.45×10^{-5} obtained by the LU-DND with 2 hidden layers. We observe that this improvement is preserved by the quantized networks with 8 bits and even 7 bits of precision using fix-point arithmetic.

We now explain how the quantized network for this experiment was constructed. Let us assume the available precision is up to k bits. First, the weights and biases of the network are rescaled and rounded to nearest integers such that the resulting parameters are all integers between $-2^{k-1}+1$ and 2^{k-1} stored as signed k-bit integers. Each individual input neuron only requires a single bit since they store zeros and ones. But we also require that the result of feedforward obtained by multiplications and additions and stored in the hidden layers is also a k-bit signed integer. Unlike float-point arithmetic, fixed point arithmetic operations can and often overflow. The result of multiplication of two k-bit fixed-point integers can span $2k$ bits in the worst case. Therefore the results of each hidden layer has to be shifted to a number of significant digits and the rightmost insignificant digits have to be forgotten. For instance, in the case of the CNOT-exRec with Steane EC units, each input layer has 12 bits, which get multiplied by 12 signed integers each with k-bit fixed point

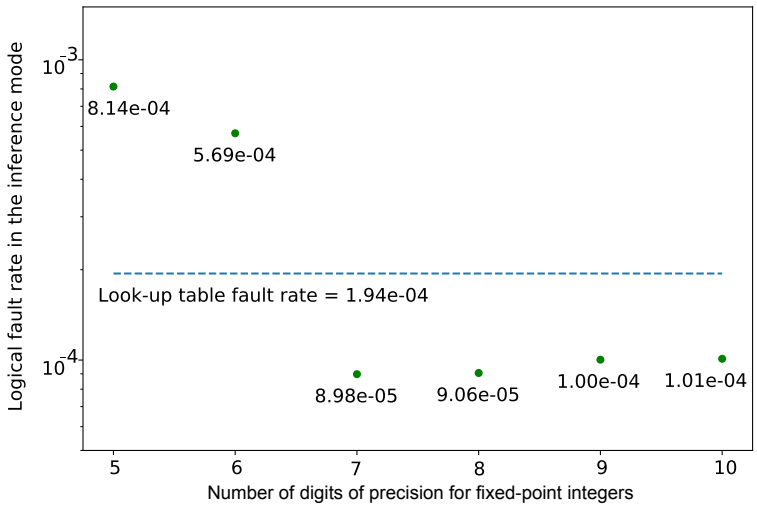


Figure 4.25: Quantization of the feedforward neural network with 2 hidden layers, trained on the Steane EC dataset at a physical error rate of $p = 2 \times 10^{-4}$. Each point is calculated as the average logical error rate obtained from 10 rounds of training and cross-validating similar to the experiments in Sec. 4.4.

precision. A bias with k-bit fixed point precision is then added to the result. We therefore need at most $k + \lceil \log_2(13) \rceil$-bits to store the result. Therefore the rightmost $\lceil \log_2(13) \rceil$ bits have to be forgotten. If the weights of the trained neural network are symmetric around zero, it is likely that only a shift to the right by 2 bits is needed in this case. Similarly, if each hidden layer has L nodes, the largest shift needed would be $\lceil \log_2(L+1) \rceil$ but most likely $\lceil \log_2(L+1) \rceil - 1$ shifts suffices. In the experiment of Fig. 4.25, each hidden layer had 1000 nodes and the feedforward results were truncated in their rightmost 9 digits.

4.5.3 Classical arithmetic performance

In the previous section we showed that 8-bit fixed point arithmetic is all that is needed for high quality inference from the trained deep neural decoder. We now consider a customized digital circuit for the inference task and estimate how fast the arithmetic processing units of this circuit have to be in order for the inference to be of practical use for active quantum error correction.

The runtime of a digital circuit is estimated by considering the time that is required for

the electric signal to travel through the *critical path* of the logical circuit [128], the path with the longest sequence of serial digital operations.

Fig. 4.26 shows the critical path of a circuit customized to carry inference in a feed-forward network with 2 hidden layers. Since the input neurons represent syndrome bits, multiplying them with the first set of weights can be done with parallel AND between the syndrome bit and the weight bits. The rectified linear unit is efficient since it only requires a NAND between the sign of the 8-bit signed integer with the other 7 bits of it. The most expensive units in this circuit are the 8×8 multipliers and adders. Every 8×8 multiplier gives a 16-bit fixed point integer which is then shifted 8-bits to the right by ignoring the first 8-bits. The total time delay t_{TOT} of this path in the circuit is

$$t_{\text{TOT}} = t_{\text{AND}} + \lceil \log(S+1) \rceil \, t_{\text{ADD}} + t_{\text{MAX}} + \sum_{i=1}^{H} (t_{\text{NOT}} + t_{\text{AND}} + t_{\text{MULT}} + \lceil \log(L_i+1) \rceil t_{\text{ADD}}), \tag{4.12}$$

where H is the number of hidden layers and L_i is the number of neurons in the i-th hidden layer. From a complexity theoretic point of view this is promising since it shows that the cost of inference is logarithmic in the number of syndromes and the size of hidden layers, and linear in the number of hidden layers. For a feedforward network with two hidden layers and at most 1000 neurons in each hidden layer,

$$t_{\text{TOT}} \quad = \quad 3t_{\text{AND}} \; + \; 2t_{\text{NOT}} \; + \; 2t_{\text{MULT}} \; + \; t_{\text{MAX}} \; + \; (\lceil \log(S+1) \rceil + 20) \, t_{\text{ADD}}. \tag{4.13}$$

Since the adders contribute the most in the above time delay, let us give an upper bound on how fast the adder units need to be in order for the total time delay to be comparable to the runtime of the fault-tolerant quantum error correction protocols considered in this chapter.

In Tab. 4.8 we compute upper bounds on the adder units for the fault-tolerant error correction protocols considered in this chapter. We emphasize that this estimation is by the optimistic assumption that all independent arithmetic operations are done in parallel. In reality, this is not possible due to limitations in area and power consumption of the ASIC. Also, considering that multiple rounds of inference have to happen, a pipeline architecture should be considered for independent batches of inference on the ASIC. Lastly, the time for a multiplier unit and the comparator are ignored since (if all independent jobs are done in parallel) there are only two serial multipliers in the critical path. With all of these considerations, the last row of this table should be interpreted as an optimistic allowed time for the adder units and that the actual adder delays should be *well below* these numbers.

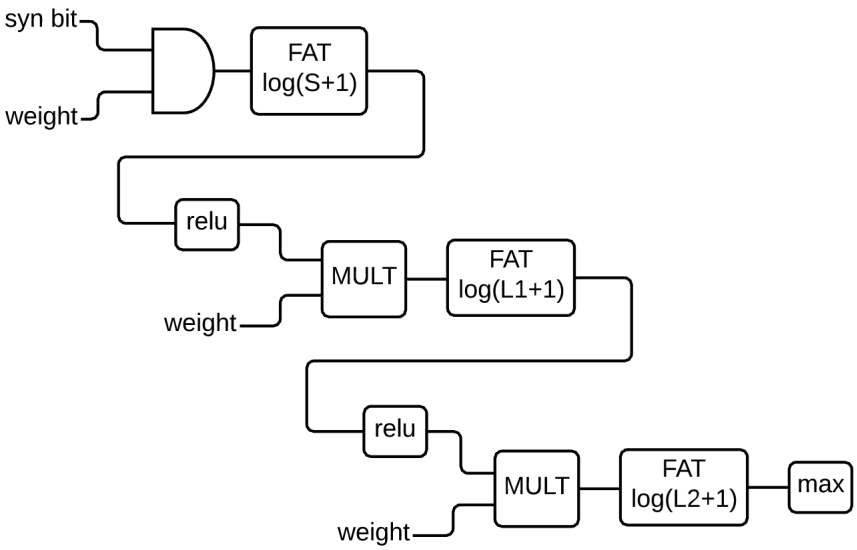


Figure 4.26: The critical path of a custom inference circuit. Every syndrome bit represents an input node of the neural network and is multiplied by 8-bit integer weights. A set of such products are added together and together with an 8-bit bias integer to find the activation on a node of the first hidden layer. Given S input syndromes, this amounts to the addition of $S+1$ integers which can be done with a tree of 8-bit integer full-adders (Full-Adder Tree or FAT for short) of depth $\log(S+1)$. After the quantized rectified linear unit, a similar procedure is iterated for the first hidden layer with the full-adder tree of depth $\log(L_1+1)$ where L_1 is the number of neurons in the first hidden layer. This pattern continues for other hidden layers. The MAX unit compares two 8-bit integers and outputs 0 if the first one is bigger and 1 if the second one is bigger.

FTEC Circuit	FTEC Depth	Syn Size	Num Adders	Adder Leniency
$d = 3$ Steane	6	12	24	2.5ns
$d = 5$ Steane	6	36	26	2.3ns
$d = 3$ Knill	8	12	24	3.3ns
$d = 5$ Knill	8	36	26	3.1ns
$d = 3$ Surface code	18	12	24	7.5ns
$d = 5$ Surface code	36	72	27	13.3ns

Table 4.8: FTEC depth is the depth of the FTEC circuit. For Steane and Knill EC, this is the depth of the CNOT-exRec circuit (excluding the ancilla verification steps) and in the surface code, it is the depth of the circuit for multiple rounds of syndrome measurement (note that for the distance 5 surface code we considered the worst case of 6 syndrome measurement rounds). The syndrome size is only that of one of X and Z since the inference for X and Z logical errors can happen in parallel and independently. The adder time leniency is calculated based on 10ns quantum gate delays. Therefore, it is the depth of the FTEC multiplied by 10ns and divided by the number of adders.

In particular we conclude that in order to perform active error correction with the methods summarized in Tab. 4.8 on a quantum computer with 10ns gate delays, the classical control unit of the quantum computer has to comprise of arithmetic units that are fast enough to perform arithmetic operations well below the time limits reported in the last column of this table. In hardware engineering, there are many approaches to implementation of arithmetic and logical units [129]. Without going into the details of the circuit designs we mention that the adder leniencies in Tab. 4.8 are in reach of high performance VLSI [130, 131], but could be challenging to achieve using FPGAs [132, 133, 134].

4.5.4 Limitations of deep neural decoders

We interpret the results of this section to suggest that, once implemented on a high performance computing platform, inference can be computed efficiently from a trained deep neural decoder. Further, the results of Sec. 4.4 show that with a large enough training set, neural network decoders achieve lower logical failure rates compared to the lookup table schemes presented in this chapter. However, this does not imply that deep neural decoders are scalable. As the size of the codes grow, training the neural decoders becomes much more daunting. This is due to the fact that deep learning classifiers are not viable solutions for *sparse* classification problems. As the codes become better and/or physical

error rates become smaller, the training samples become more and more sparse, providing less and less effective training samples for the neural network. Without nontrivial training samples, the neural networks learn "zeros" rather than capturing significant patterns in the data set.

As evidence for the effect of sparsity of the dataset on successful training of the deep neural decoding we refer the reader to an experiment reported in Fig. 4.27. In this experiment, the DND is trained on the dataset corresponding to the highest physical fault rate $p = 2 \times 10^{-3}$. The same trained DND is used to cross-validate on test datasets for all other physical fault rates. We observe that this DND is more successful in recovery inference for smaller physical error rates, even though it is trained on a "wrong" dataset. It is important to note that this experiment does not provide an improved method for training a neural network for error correction on a physical realization of a quantum processor. Firstly, in any manufactured quantum device the error model will not be entirely known (and is not necessarily close to a theoretic noise model such as the depolarizing channel). And secondly, the error of the device cannot be intensified intentionally for the purpose of training a deep neural decoder, to be later used on a less noisy device.

4.6 Summary and outlook

To summarize, in this chapter we considered multiple fault-tolerant schemes and using several neural network architectures to train decoders in a full circuit-level noise framework. Although our analysis was done for Pauli channels, we expect that for non-Pauli noise models, the improvements could be even more significant than what was observed in our work. Evidence of this can be found from the results obtained in Chapter 2.

From a machine learning point of view, we applied state-of-the-art techniques used in training neural networks. While considering many network designs, we used the same hyperparameter tuning methodology to achieve unbiased and reliable results. Consequently, we successfully observed a clear advantage in using deep networks in comparison with single hidden layer networks and regression methods. On the other hand, we provided clear evidence of the realistic limitations of deep learning in low noise rate regimes. In particular, scaling the neural network to large distance codes appears to be a significant challenge. For large scale quantum computations, decoders that work less well than neural decoders trained on small distance codes but which are scalable would clearly be the better option. Lastly, we gave a rigorous account of the digital hardware resources needed for inference and runtime analysis of the critical path of the customized digital circuitry for high performance inference.

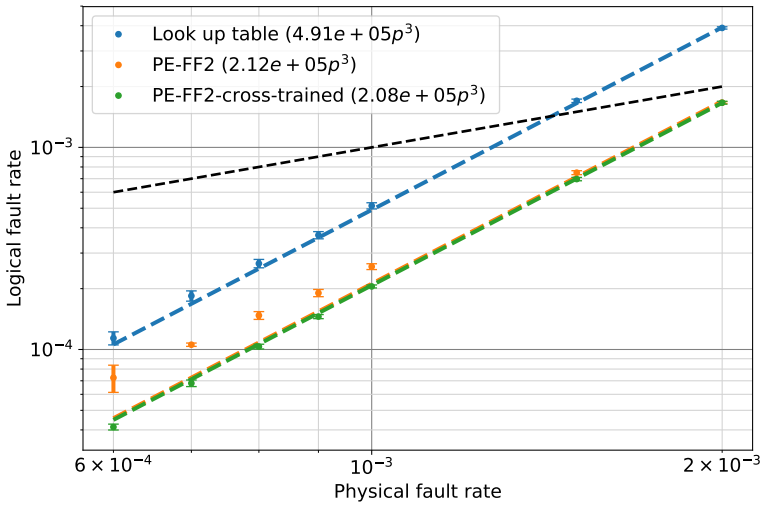


Figure 4.27: A comparison between two training procedures for the CNOT-exRec of the $[\![19, 1, 5]\!]$ color code using Steane-EC units. The orange dots are the results of training a feedforward network with 2 hidden layers as reported also in Fig. 4.19. In this case, the DND is trained on a given physical error rate p and tested on a test dataset for the same physical error rate. We observe that the logical error rate does not exactly follow a cubic growth since the training is less successful when the physical error rate is small. The green line demonstrates the performance of the same DND if trained only for the largest physical error rate $p = 2 \times 10^{-3}$ and later on tested on test datasets from every other physical error rate. As previously explained, such a training scenario is not possible for real-world experiments, or on physical realizations of quantum computers.

There remain many interesting future directions for designing improved and efficient decoders which work well in fault-tolerant regimes. One such avenue would be to tailor machine learning algorithms specifically designed for decoding tasks. In particular, finding machine learning algorithms which work well with sparse data would be of critical importance. It would also be interesting to apply the methods introduced in this work to actual quantum devices that are currently being developed. It most certainly will be the case that fault-tolerant designs will be tailored to a particular quantum architecture. This would lead to further areas in which machine learning could be extremely useful for finding improved decoders.

Chapter 5

Fault-tolerant quantum computing in the Pauli or Clifford frame with slow error diagnostics

The material of this section is based on the journal article of Ref.[96], copyrighted by the Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften. I was the primary investigator of the research which was initiated by David Poulin. I wrote the majority of the manuscript and Pavithran Iyer wrote the appendix. All three authors contributed in the editing of the manuscript.

5.1 Introduction and Motivation

In this chapter, we are concerned with the impact of slow error diagnostics on fault-tolerance schemes. There are several origins for this concern. First, in certain solid-state and ion-trap qubit architectures, measurement times can be between 10 to 1000 times slower than gate times [135, 136, 137, 138, 139, 140, 141]. Thus, on the natural operating time-scale of the quantum computer, there is a long delay between an error event and its detection. Second, processing the measurement data to diagnose an error—i.e., decoding—can be computationally demanding. For instance in Chapter 4, we saw that even with optimistic assumptions, performing active error correction could be a significant challenge. Thus, there can be an additional delay between the data acquisition and the error identification. At first glance, these delays might cause the error probability to build

up between logical gates, thus effectively decreasing the fault-tolerance threshold. But as we will see, this is not necessarily the case.

One of the key tricks to cope with slow error diagnostics is the use of error frames [55]. While the basic idea of error correction is to diagnose and correct errors, it can often be more efficient to keep track of the correction in classical software instead of performing active error correction. In particular, this saves us from performing additional gates on the system, thus removing potential sources of errors. In most quantum error-correction schemes, the correction consists of a Pauli operator, i.e., tensor products of the identity I and the three Pauli matrices X, Y, and Z. Thus, at any given time, the computation is operated in a random but known *Pauli frame*: its state at time t is $P|\psi(t)\rangle$ for some Pauli operator P and where $|\psi(t)\rangle$ denotes the ideal state at time t. When error-diagnostics are slow, the system will unavoidably evolve in an unknown error frame for some time.

The problem of slow measurements in solid-state systems was addressed by DiVincenzo and Aliferis [20] in the context of concatenated codes. In addition to error diagnostics, concatenated schemes require measurements to prepare certain ancilla states used to fault-tolerantly extract the error syndrome and to inject magic states to complete a universal gate set. DiVincenzo and Aliferis' scheme hinges on the fact that the logical gate rate decreases exponentially with the number of concatenation levels; thus, at a sufficiently high level, measurement and gate times become commensurate. To concretely realize this simple observation, they combine a number of known and new techniques including ancilla correction, high-level state injection, and Pauli frames.

One limitation of this solution is that it only applies to concatenated codes realizing a universal gate set through magic state injection. This leaves out for instance surface codes [7] or concatenated codes with alternate universal gate constructions [19, 63, 74, 75, 76, 77]. In particular, they inject noisy magic-states directly at high concatenation levels, thus losing the benefit of low-level correction. In addition, when decoding times are very slow, this solution could wastefully use additional layers of concatenation with the sole purpose of slowing down the logical gates. Another drawback of their scheme is that it requires active error correction to ensure that high-level corrections are always trivial.

In this article, we introduce an alternative scheme to cope with slow error diagnostics which applies broadly to all codes and universal gate constructions. Like the DiVincenzo and Aliferis scheme, the key idea will be to slow-down the logical gate rate to learn the error frame before it propagates to the rest of the computation. This is simply achieved by spacing out gates in the logical circuit and thus circumvents the unnecessary additional qubit overhead associated to extra concatenation layers. Our scheme does not require active error correction, it is entirely realized in an error frame. In addition, our scheme is

compatible with a more general form of error correction which uses *Clifford frames*, where at any given time t, the state of the computer is $C|\psi(t)\rangle$ where C is a tensor-product of single-qubit Clifford group elements. Note that details of the Pauli frame scheme are given in Sec. 5.2 whereas details of the Clifford frame scheme are given in Sec. 5.3.

Regarding slow measurements, it is important to distinguish two time scales. First, the time t_l it takes for the outcome of a measurement to become accessible to the outside world. For instance, t_l could be caused by various amplification stages of the measurement, and we refer to it as a measurement *latency*. Second, the measurement *repetition* time t_r is the minimum time between consecutive measurements of a given qubit. In principle, t_r can be made as small as the two-qubit gate time, provided that a large pool of fresh qubits are accessible. Indeed, it is possible to measure a qubit by performing a CNOT to an ancillary qubit initially prepared in the state $|0\rangle$ and then measuring this ancillary qubit. While the ancillary qubit may be held back by measurement latencies for a time t_l before it can be reset and used again, other fresh qubits can be brought in to perform more measurements in the meantime. The scheme we present here and the one presented in [20] are designed to cope with measurement latencies t_l, but both require small t_r. Indeed, the accuracy threshold is a function of the error rate per gate time (including measurement gates). Thus, increasing the measurement repetition time t_r relative to the decoherence rate will unavoidably yield a lower threshold.

While the DiVincenzo and Aliferis scheme was motivated by slow measurements, most of it applies directly to the case of slow decoding. Our scheme too is oblivious to the origin of slow error diagnostics. We emphasize that slow decoding is a very serious concern. For instance, numerical simulations used to estimate the threshold or overhead of the surface code are computationally dominated by the decoding algorithm and require intense computational resources. Depending on the code distance and error rate, a single decoding cycle can take well above $1\mu s$ [142] on a standard processor, considerably slower than the natural GHz gate rate in the solid state.

As explained in Ref. [95], if the rate of the classical syndrome processing (decoding) is smaller than the rate at which the syndrome is generated, an exponential slow down would occur during the computation. However, just as measurements with long latencies can be handled with a supply of additional fresh measurement qubits, slow decoding can be handled with a supply of parallel classical computers so that the overall decoding rate matches the syndrome creation rate. Thus, in this article, slow error diagnostics is used to designate *latencies* in measurements and/or decoding.

Finally, to our knowledge, a theory of Clifford frames for fault-tolerant quantum computation has not been worked out before. By enabling them, our scheme offers a greater

flexibility for error correction, thus potentially correcting previously non-correctable error models, or increasing the threshold of other noise models as was seen in Chapter 2. More generally, the possibility of fault-tolerantly computing in a Clifford frame opens up the door to new fault-tolerant protocols, e.g., using new codes or new hardware that have a different set of native fault-tolerant gates. For instance, Clifford frames were used as an accessory in measurement-based quantum computation with Majorana fermions [143]. Lastly, the possibility of computing in a Clifford frame could have applications to randomized compiling [144] which introduces random frames to de-correlate physical errors.

In Sec. 5.2 we will focus on quantum computation in the Pauli frame and present our scheme. In Sec. 5.3 we will introduce quantum computing in the Clifford frame and discuss the primary challenges. The remaining sections (Sec. 5.3.1 and Sec. 5.3.2) will provide the details for how to overcome these challenges.

5.2 Pauli frame

Recall that in Sec. 1.1, we defined $\mathcal{P}_n^{(1)}$ to be the n-qubit Pauli group and the Clifford hierarchy was defined by $\mathcal{P}_n^{(k)} = \{U : UPU^\dagger \in \mathcal{P}_n^{(k-1)} \quad \forall P \in \mathcal{P}_n^{(1)}\}$.

In physical implementations where measurement times are much longer than gate times or when error decoding is slow, if one were to perform active error correction, a large number of errors could potentially build up in memory during the readout times of the measurement. However, for circuits containing only Clifford gates, all Pauli recovery operators can be tracked in classical software without ever exiting the Pauli group. Indeed, suppose that at some given time during the computation, the state of the computer is in some Pauli frame defined by P—i.e., the state of the computer is $P|\psi\rangle$ where $|\psi\rangle$ is the ideal state (here P can be any element of $\mathcal{P}_n^{(1)}$, not necessarily a logical Pauli operator). If we then apply a *Clifford gate* $U \in \mathcal{P}_n^{(2)}$ (again, not necessarily a logical gate), then the state will be $UP|\psi\rangle = P'U|\psi\rangle$ where P' is another Pauli operator. We thus see that the effect of U is to correctly transform the perfect state $|\psi\rangle$ and to change the Pauli frame P in some known way. Moreover, updating the Pauli frame $P' = UPU^\dagger$ can be done efficiently, with complexity $\mathcal{O}(n^2)$ [23]. This shows that as long as we only apply Clifford gates, there is no need to actively error-correct, we can instead efficiently keep track of the Pauli frame in classical software [55].

In concatenated codes for instance, error correction is performed in between the application of gates to ensure that errors don't accumulate in an uncontrolled fashion. A gate location in a quantum algorithm is thus replaced by an extended rectangle, where error

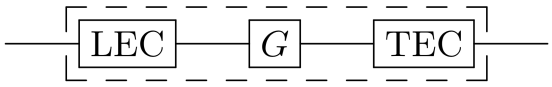


Figure 5.1: Example of extended rectangle (exRec) for implementing the logical gate G. The leading and trailing EC circuits (LEC and TEC) perform fault-tolerant error correction for input errors and errors occurring during the implementation of G.

correction is performed both before and after the application of the gate [18]. The leading error correction circuit in an extended rectangle is used to correct input errors that could have occurred prior to the application of the gate. The trailing error correction circuits correct errors that can occur during the application of the gate (see Fig. 5.1). Each error correction circuit will multiply the current Pauli frame by a Pauli operator (the correction).

Since Clifford gates can be efficiently simulated on a classical computer, non-Clifford gates are needed for universal quantum computation. Universal quantum computation can be achieved for instance using gates from the Clifford group combined with the $T = \mathrm{diag}(1, e^{i\pi/4}) \in \mathcal{P}_1^{(3)}$ gate [97]. In general, Pauli operators will not remain in the Pauli-group when conjugated by non-Clifford gates and so the Pauli frame cannot simply propagate through them.

Consider the application of a logical T gate in a quantum algorithm. Since measurement times are much longer than gate times, the Pauli frame right before the application of a T gate can only be known at a later time. Furthermore, by definition of the Clifford hierarchy, Pauli operators are mapped to Clifford operators under conjugation by $T \in \mathcal{P}_1^{(3)}$ gates. Therefore, the output correction after applying a logical T gate can be outside the Pauli frame.

In order to overcome these obstacles, we note that if error correction is performed immediately after the application of the T gate, the output correction can be written as a *logical* Clifford gate C times a Pauli matrix (a proof is presented in Appendix. C).

If we were to keep track of the logical Clifford correction C in classical software, it could propagate through the next T gate, resulting in a correction involving even more T gates. It could also propagate through a logical two-qubit gate (such as a CNOT) resulting in a two-qubit correction (the exact transformation rules are derived in Sec. 5.3). The two-qubit corrections could then propagate through other gates in the quantum algorithm leading to a generic Clifford correction. To prevent these scenarios from occurring, a buffer can be inserted right before the next logical two-qubit gate or T gate part of the quantum algorithm. The role of the buffer is to learn what the Pauli frame was right before the

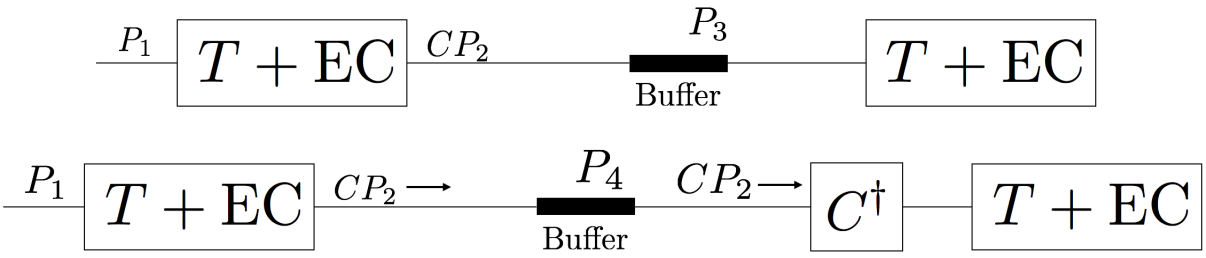


Figure 5.2: Illustration of the scheme for propagating Pauli corrections through a T gate when error diagnostics are much longer than gate times. (TOP) When propagating the input Pauli P_1 through a T gate and performing error correction, the output can be written as CP_2 where C is a logical Clifford gate and P_2 is a Pauli matrix. A buffer is introduced to learn the Pauli frame immediately before applying the T gate which enables the logical Clifford correction C to be known. During the buffer, repeated rounds of error correction are performed to prevent the accumulation of errors for qubits waiting in memory. We denote the final Pauli correction arising from the EC rounds as P_3. (BOTTOM) We propagate the correction CP_2 through the buffer and apply a logical Clifford gate $C^\dagger$ in order to remove C thus restoring the Pauli frame. Although the propagation can map the buffer correction $P_3 \to P_4$, P_4 remains Pauli and can be known at a later time.

application of the previous logical T gate. During the buffer, repeated rounds of error correction are performed to prevent the accumulation of a large number of errors. There could be leftover Pauli corrections arising from error correction rounds which would only be known at a later time. However, by propagating the logical Clifford correction through the buffer, the Pauli corrections would remain Pauli.

Once the logical Clifford correction is propagated through the buffer, we apply a logical $C^\dagger$ thus restoring the Pauli frame. The protocol guarantees that only Pauli corrections would be propagated through the next logical T or two-qubit gates. An illustration of the scheme is outlined in Fig. 5.2. As in [20], the proposed approach also effectively slows down gate times making them comparable to measurement times[1]. However, buffers are only introduced when necessary and without having to increase the size of the code.

We conclude this section by noting that in [20, 95], the Pauli frame scheme was described in the context of concatenated codes where T gates are implemented using state injection as shown in Fig. 5.3. In these schemes, a buffer is included to learn what the logical Pauli frame was before applying the SX correction in order to correctly interpret the outcome of

[1]Note that the buffer increases the overall time for implementing a non-Clifford gate. However, this impacts the overall running time of the quantum algorithm only by a constant factor.

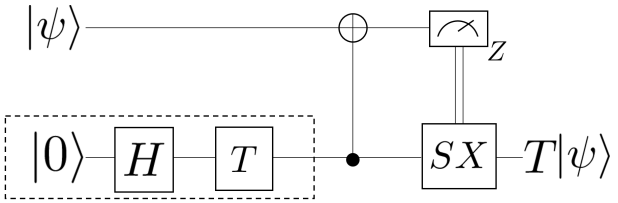


Figure 5.3: Circuit for implementing a T gate. The circuit uses the state $T|+\rangle$ which is prepared offline and applies a sequence of Clifford gates. The correction SX is only applied if the Z-basis measurement outcome is -1.

the Z-basis measurement. For instance, if a logical Pauli X occurred on the data prior to implementing the T gate, the Clifford correction SX would be applied if the measurement outcome was $+1$ instead of -1 (see the caption of Fig. 5.3). Further, as was described in [20], additional layers of concatenation are required to prevent the build up of errors in the presence of quantum measurements with a long latency.

While it builds on the same general ideas, the Pauli frame scheme presented in this section is not restricted to a particular implementation of the T gate (for instance, it also directly applies to codes which can implement a logical T gate transversely) nor to concatenated schemes. In fact, our scheme slightly differs from [20, 95] even in the case where the T gate is applied using state injection. Indeed, one does not wait to learn what the frame was before determining whether to apply the logical SX correction (the Z-basis measurement outcome is always interpreted in the same way). The entire state injection circuit is completed before the Pauli frame (prior to the application of the state injection circuit) is known. Once it becomes known, one would know if the wrong logical SX correction was applied and any remaining logical Clifford errors would be removed. So in particular, there is no need to introduce additional coding layers to slow-down the logical gate rate during this waiting time. While this is a fairly minor distinction, it does enable us to generalize to any other coding schemes and implementations of the T gate, a feature which has not been addressed prior to our work.

5.3 Clifford frame

As was shown in Sec. 2.6, including Clifford corrections as part of the recovery protocol for error correcting codes that can implement logical Clifford gates transversely can significantly improve the code's threshold for certain noise models. In particular, for coherent noise channels ($\mathcal{N}(\rho) = e^{i\boldsymbol{n}\cdot\boldsymbol{\sigma}}\rho e^{-i\boldsymbol{n}\cdot\boldsymbol{\sigma}}$ where $\boldsymbol{n} = (n_x, n_y, n_z)$ with $||\boldsymbol{n}|| = 1$), we showed

that in some regimes the 5-qubit code can tolerate an arbitrary amount of coherent noise when Clifford corrections are used, while Pauli corrections have a finite threshold.

In this section, we extend the protocol used to cope with slow error-diagnostics to the case where error correction uses Clifford gates. When performing error correction on a set of encoded qubits with a stabilizer code, if one measures a non-trivial syndrome value l, a recovery map R_l is applied to the data block. The recovery map can always be written in the form $R_l = \mathcal{L}(l)\mathcal{T}(l)\mathcal{G}(l)$ where $\mathcal{G}(l) \in \mathcal{P}_n^{(1)}$ is a product of stabilizer generators, $\mathcal{L}(l)$ is a product of logical operators and $\mathcal{T}(l) \in \mathcal{P}_n^{(1)}$ is a product of pure errors [29], see also Appendix. C. Pure errors form an abelian group of operators that commute with all of the code's logical operators and all but one of the codes stabilizer generators. The logical operators $\mathcal{L}(l)$ are chosen to maximize the probability of recovering the correct codeword – this is the decoding problem. The operators in the set $\mathcal{L}(l)$ are not necessarily restricted to logical Pauli operators as they can be extended to include all logical operators that can be applied fault-tolerantly.

As was done in Chapter 2, it is natural to restrict $\mathcal{L}(l)$ to gates that can be applied transversally on the code. In particular, we will consider logical corrections in $(\mathcal{P}_1^{(2)})^{\otimes n}$, the group generated by n-fold tensor products of single-qubit Clifford operators. This latter group is generated by $\mathcal{P}_1^{(2)} = \langle H, S\rangle$. The order of the group is $|\mathcal{P}_1^{(2)}| = 24$ (ignoring global phases). For the n-qubit case, $\mathcal{P}_n^{(2)} = \langle H_i, S_i, \mathrm{CNOT}_{ij}\rangle$ where the indices i, j indicate which qubits to apply the Clifford gates (see Eq. 1.4). We remind the reader that 2-D color codes [76] admit transversal realizations of all gates in $\mathcal{P}_1^{(2)}$ and the 5-qubit code admits transversal realizations of logical gates in the set generated by $\langle SH, X, Z\rangle$.

For concatenated codes, we restrict the discussion to the case where logical Clifford corrections are performed at the last concatenation level only. If Clifford corrections were performed at every level, one would need to wait for the measurement outcomes of every level and actively perform Clifford corrections. This is because a level-k logical Clifford correction does not generally commute with the level-$(k+1)$ syndrome measurements. The goal of defining a Clifford frame is to avoid actively correcting since the corrections themselves can introduce more errors into the computation.

We now derive the transformation rules for Clifford operators propagating through CNOT gates. Two-qubit controlled unitary gates $C\text{-}U_{12}|a\rangle|b\rangle = |a\rangle U^a|b\rangle$, where the first qubit is the control and the second qubit is the target, can be written as

$$C\text{-}U_{12} = \frac{1}{2}(I+Z)\otimes I + \frac{1}{2}(I-Z)\otimes U. \tag{5.1}$$

Note that from this definition $\mathrm{CNOT}_{12} = C\text{-}X_{12}$. Using Eq. 5.1, it is straightforward to

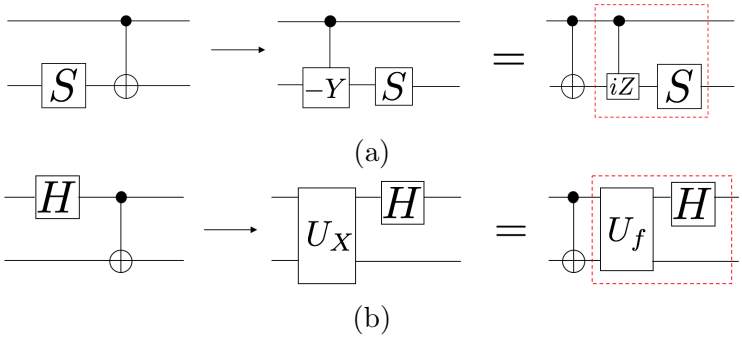


Figure 5.4: (a) Propagation of $I \otimes S$ through a CNOT gate. (b) Propagation of $H \otimes I$ through a CNOT gate where $U_f = \frac{1}{2}(I+iY)\otimes I + \frac{1}{2}(I-iY)\otimes X$. In both cases, if instead of correcting the Clifford errors prior to the CNOT gate we were to keep track of them using a Clifford frame, the corrections would involve two-qubit gates in addition to the original Clifford corrections.

show the following relations

$$(S \otimes I)C\text{-}X_{12} = C\text{-}X_{12}(S \otimes I), \tag{5.2}$$
$$(I \otimes S)C\text{-}X_{12} = C\text{-}Y_{12}(I \otimes S), \tag{5.3}$$
$$(I \otimes H)C\text{-}X_{12} = C\text{-}Z_{12}(I \otimes H), \tag{5.4}$$
$$(H \otimes I)C\text{-}X_{12} = U_X(H \otimes I), \tag{5.5}$$

where we defined $U_X \equiv \frac{1}{2}(I+X)\otimes I + \frac{1}{2}(I-X)\otimes X$.

From Fig. 5.4 and Eq. 5.2 to Eq. 5.5, it can be seen that propagating Clifford corrections through CNOT gates can result in both single and two-qubit Clifford corrections. By keeping track of logical Clifford corrections in classical software, these could grow due to other CNOT gates resulting in a generic Clifford correction. Furthermore, when propagating Clifford corrections through non-Clifford gates (such as the T gate), the output can potentially be outside of the Clifford hierarchy. These could then propagate through the remainder of the logical circuit resulting in a unitary gate correction expressed as a product of several T gates.

5.3.1 Clifford propagation through CNOT gates

We first address the propagation of logical Clifford corrections (expressed in tensor product form) through CNOT gates. The goal is to prevent a two-qubit correction from spreading

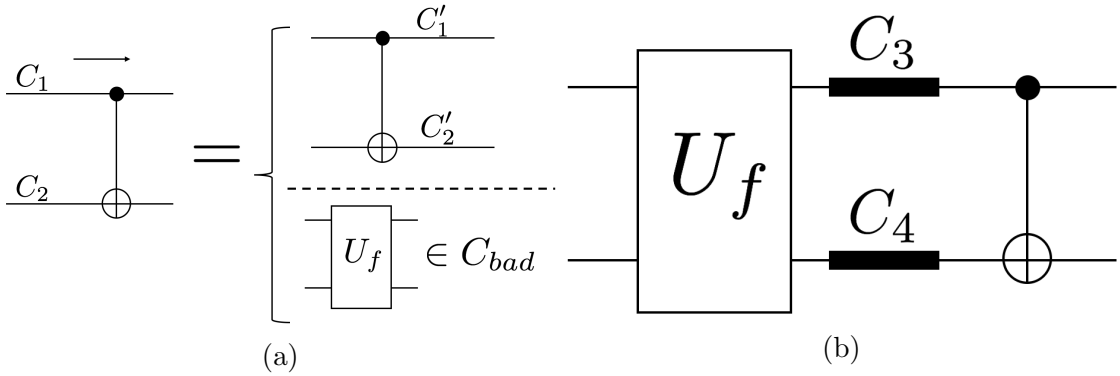


Figure 5.5: (a) Propagating input Clifford gates $C_1 \otimes C_2$ across a CNOT (part of a quantum algorithm) leads to two possible outcomes, one in C_{good} (defined in Eq. 5.7) and the other in C_{bad} (defined in Eq. 5.8). (b) Buffers are introduced to learn if the outcome in Fig. 5.5a belongs to C_{good} or C_{bad}. Rounds of error correction in the buffers introduce the corrections $C_3 \otimes C_4$. If the outcome in Fig. 5.5a belongs to C_{bad}, we apply a CNOT correction following the buffers. The protocol is repeated until the resulting gate belongs to C_{good}.

to multiple code blocks in order to avoid performing a generic Clifford correction. After the application of a logical CNOT gate (as part of a quantum algorithm), we can place a buffer before the next logical two-qubit gate (or non-Clifford gate) in order to determine what the Clifford frame was right before the application of the logical CNOT. Note that during the application of the buffer, repeated rounds of error correction are performed to prevent the build-up of a large number of errors, producing additional Clifford gates.

From Eq. 5.2 to Eq. 5.5, upon propagating the input Clifford gates $C_1 \otimes C_2$ through a logical CNOT gate part of a quantum algorithm using a Clifford frame, two outcomes are possible. In the first case, we can have

$$C\text{-}X_{12}(C_1 \otimes C_2) = (C'_1 \otimes C'_2)C\text{-}X_{12}, \tag{5.6}$$

for some Clifford gates C'_1 and C'_2. In particular, we define

$$C_{good} = (C_1 \otimes C_2)C\text{-}X_{12}. \tag{5.7}$$

If Eq. 5.6 is not satisfied for $C_1 \otimes C_2$, then the output will belong to the set C_{bad} defined to be

$$C_{bad} = \mathcal{P}_2^{(2)} \setminus C_{good}, \tag{5.8}$$

where $\mathcal{P}_2^{(2)}$ is the two-qubit Clifford group. Performing a computer search, we found that out of the $24^2 = 576$ possible input Clifford gates (expressed in tensor product form), 64 will satisfy Eq. 5.6.

After the application of the CNOT gate part of the quantum algorithm, the buffers will introduce the Clifford corrections $C_3 \otimes C_4$ arising from the repeated rounds of error correction. Once the Clifford frame prior to applying the CNOT gate is known, we will be able to determine if the output from the propagation of the Clifford frame belongs to C_{good} or C_{bad}. If it belongs to C_{good}, then no further operations are required. In the case where it belongs to C_{bad}, we perform a logical CNOT correction after the buffer. A second set of buffers is introduced to determine if the resulting gate belongs to C_{good} or C_{bad}. We can repeat this process recursively until the resulting gate belongs to C_{good}.

Assuming the input Clifford gates and buffer Clifford corrections are chosen uniformly at random, we performed a simulation to estimate the transition probability from $C_{bad} \to C_{good}$. Performing 5×10^5 simulations, we found that the $C_{bad} \to C_{good}$ transition occurs with probability $\frac{1}{12}$. Thus, when computing in a random Clifford frame, each logical CNOT requires on average $12 + 1$ logical CNOTs.

Lastly, we point out that for noise models where the Clifford corrections arising from the buffer are biased towards the Pauli gates, it would be more advantageous to apply $C_1^\dagger \otimes C_2^\dagger$ (where C_1 and C_2 are the input Clifford gates in Fig. 5.5a) after the following initial CNOT correction. More specifically, if the probability of acquiring a non-trivial Clifford correction in the buffer is ε, then the $C_{bad} \to C_{good}$ transition probability becomes $1 - \varepsilon\frac{11}{12}$.

5.3.2 Clifford propagation through T gates

We now address the propagation of Clifford corrections through a logical T gate, which is a non-Clifford gate. When applying a logical T gate in a quantum algorithm, we could also place a buffer before the next logical gate part of the quantum algorithm to learn the Clifford frame immediately before applying the T gate. If the output correction is non-Clifford, we can perform appropriate corrections in order to restore the Clifford frame. If successful, this would guarantee that the input correction to the next gate would belong to the Clifford group.

Suppose the input to the logical T gate is a Clifford correction C_1, so the resulting gate is TC_1. On the one hand, if $TC_1 = \tilde{C}_1 T$ for some Clifford $\tilde{C}_1$—or equivalently $TCT^\dagger \in \mathcal{P}_1^{(2)}$—then no further operations are necessary. Only gates in the set generated

by $C_1 \in \langle S, X\rangle$ satisfy this property. In fact, if $C_1 \in \langle S, X\rangle$, then $TC_1T^\dagger \in \langle S, X\rangle$. Thus, we define

$$C_- = \langle S, X\rangle, \quad \text{and} \quad C_+ = \mathcal{P}_1^{(2)} \setminus C_-. \tag{5.9}$$

Note that $|C_-| = 8$ while $|C_+| = 24 - 8 = 16$. Once we learn that the input Clifford correction C_1 to the logical T gate belongs to C_-, then no further operations are necessary to restore the Clifford frame. If the buffer Clifford operations are uniformly distributed over the Clifford group, this occurs with probability $\frac{1}{3}$.

On the other hand, when we learn that the input Clifford correction C_1 to the logical T gate belongs to C_+, we apply another logical T in the hope to restore the Clifford frame. Since an additional Clifford gate C_2 was accumulated during the buffer, the resulting gate is TC_2TC_1. When $C_2 \in C_-$, then $TC_2TC_1 = \tilde{C}_2T^2C_1 \in \mathcal{P}_1^{(2)}$, since $T^2 = S$ is a Clifford transformation. At this stage, we have returned to a Clifford frame, but have removed the desired logical T gate: we are thus back at our starting point and can try anew.

Once again, whenever the Clifford corrections occurring during the buffer are biased to the Pauli group (e.g., when the probability of an error is low), before applying another T gate correction to restore the algorithm T gate, we should apply the Clifford transformation $(\tilde{C}_2SC_1)^\dagger$. In this way, we would increase the probability of being in a state of the form CT where $C \in C_-$ (thus restoring the Clifford frame). To take this possibility into account, we will henceforth assume that the Clifford corrections arising from the buffer belong to C_- with probability $1 - p$ and to C_+ with probability p.

Define $\mathcal{T}^{(0)} = \mathcal{P}_1^{(2)}$ to be the set of single-qubit Clifford gates, and define

$$\mathcal{T}^{(k)} = \prod_{j=1}^{k}(TC_j) \quad \text{where} \quad C_j \in C_+. \tag{5.10}$$

Every time we learn that the previous buffer Clifford was in C_+, we apply a T gate and wait for another buffer. Since this buffer will belong to C_- with probability $1 - p$ and to C_+ with probability p, each step of the above protocol can be seen as a step in a random walk over the sets $\mathcal{T}^{(k)}$ with transition $\mathcal{T}^{(k)} \to \mathcal{T}^{(k+1)}$ occurring with probability p and transition $\mathcal{T}^{(k)} \to \mathcal{T}^{(k-1)}$ occurring with probability $1 - p$. Every time $\mathcal{T}^{(0)}$ is reached, there is a probability $1 - p$ that a logical T gate is successfully realized at the next step.

To summarize, when attempting to implement a logical T gate starting in a Clifford frame $\mathcal{T}^{(0)}$, we either succeed with probability $1 - p$ or end up applying a $\mathcal{T}^{(1)}$ gate with probability p. In the latter case, we enter a random walk over $k \in \mathbb{N}$, and our goal is to return to $k = 0$. This clearly requires an odd number of steps. The one-step process

$1 \to 0$ occurs with probability $1-p$. The three step process $1 \to 2 \to 1 \to 0$ occurs with probability $p(1-p)^2$. Generalizing to $t = 2j+1$ time steps, where $j \in \mathbb{N}$, the number of paths that lead to a gate in $\mathcal{T}^{(0)}$ is given by the j'th Catalan number $K_j = \frac{1}{j+1}\binom{2j}{j}$ [145]. Therefore, the probability of returning to a gate in $\mathcal{T}^{(0)}$ after $t = 2j+1$ time steps is given by

$$P_{2j+1} = \frac{1}{j+1}\binom{2j}{j} p^j (1-p)^{j+1}. \tag{5.11}$$

Using the generating function for the Catalan number $c(x) = \sum_{j\geq 0} K_j x^j = (1 - \sqrt{1-4x})/(2x)$, the total probability that the random walk terminates is given by

$$\sum_{k\geq 0} P_{2k+1} = \min\left\{\frac{1-p}{p}, 1\right\}. \tag{5.12}$$

If $p > 1/2$, then with finite probability the random walk will not terminate whereas if $p \leq 1/2$, the random walk is guaranteed to terminate. This means that we must choose Clifford gates from C_- with probability greater than $1/2$. The latter condition can be satisfied in cases where Clifford corrections arising from the buffer are biased towards gates belonging to C_-, or in particular if they are biased towards Pauli gates or the identity. When the buffers are chosen uniformly over the Clifford group, then $p = 16/24 = 2/3 > 1/2$ so with some finite probability the procedure will not terminate.

We conclude this section by calculating the probability of obtaining a gate in $\mathcal{T}^{(0)}$ within n time steps, which we define as $F(p,n)$. This quantity gives the number of expected T gate corrections that need to be applied in order to restore the Clifford frame after propagation through a logical T gate. The probability $F(p,n)$ is obtained by summing Eq. 5.11 with a cut-off of n time-steps. The result is given by

$$\begin{aligned} F(p,n) &= \sum_{k=0}^{n} P_{2k+1} \\ &= 1 - f(p,n), \end{aligned} \tag{5.13}$$

where

$$\begin{aligned} f(p,n) = \frac{1-p}{2+n}\binom{2(n+1)}{n+1}(p(1-p))^{n+1} \times \\ \times \ {}_2F_1(1, \frac{3}{2}+n; 3+n; 4p(1-p)), \end{aligned} \tag{5.14}$$

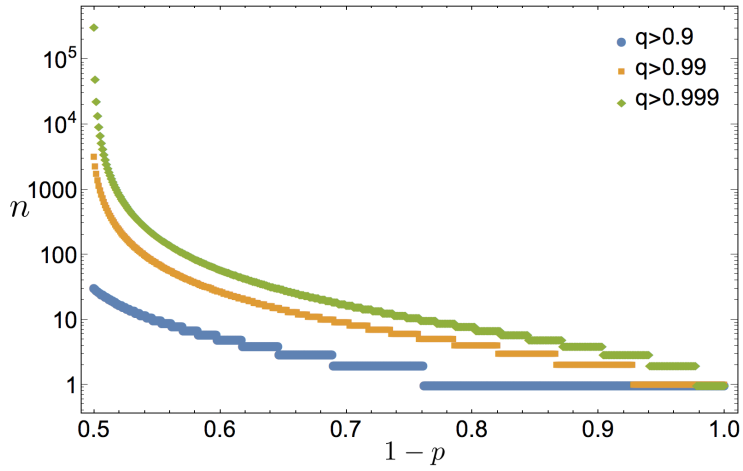


Figure 5.6: For a fixed value of $1-p$, we plot the value of n such that $F(p,n) > q$. We give plots for $q = 0.9$, $q = 0.99$ and $q = 0.999$. Hence, each curve corresponds to the expected number of T gate corrections that are required for obtaining a gate in $\mathcal{T}^{(0)}$ with probability greater than q.

and ${}_2F_1(a,b;c;z)$ is the Hypergeometric function defined in [146]. Plots of Eq. 5.13 are given in Fig. 5.6.

We now obtain an upper bound on the number of time steps n that are necessary to restore the Clifford frame with probability $q = 1-\varepsilon$. In other words, we would like to obtain an upper bound on n such that $F(p,n) > 1-\varepsilon$. We first obtain a lower bound for the function $f(p,n)$ by using the following inequalities

$$\binom{2(n+1)}{n+1} \geq \left(\frac{2(n+1)}{(n+1)}\right)^{n+1} = 2^{n+1}, \tag{5.15}$$

$${}_2F_1(1, \frac{3}{2}+n; 3+n; 4p(1-p)) \geq 2p+1. \tag{5.16}$$

Inserting Eq. 5.15 and Eq. 5.16 into Eq. 5.13, we obtain

$$F(p,n) \leq 1 - \frac{(1-p)(2p+1)}{n+2}[2p(1-p)]^{n+1}. \tag{5.17}$$

Setting $F(p,n) > 1-\varepsilon$, we obtain

$$\frac{[2p(1-p)]^{n+1}}{n+2} \leq \frac{\varepsilon}{(1-p)(2p+1)} \tag{5.18}$$

For $n \geq 1$, we have $1/(n+2) > (\frac{1}{2})^{n+1}$, so Eq. 5.18 becomes

$$[p(1-p)]^{n+1} \leq \frac{\varepsilon}{(1-p)(2p+1)} \tag{5.19}$$

which shows that $n \sim |\log \varepsilon|$—the T gate overhead for restoring the Clifford frame scales logarithmically in ε.

We now consider regimes where the noise is below the fault-tolerance threshold of a code (say $p \lesssim 10^{-2}$ as is required for the surface code [7]). In such regimes, corrections based on syndrome measurement outcomes will be significantly biased towards the identity, or more generally towards Pauli operators. More specifically, suppose that for a given noise model and code, a Pauli correction leads to a logical error rate δ_P. Applying Clifford corrections can only improve this logical error rate to $\delta_C \leq \delta_P$ since they include Pauli corrections. But in a Clifford correction scheme, non-Pauli corrections are only used when Pauli corrections are not optimal, which occurs at a rate δ_P, which is small below threshold. Consequently, in the case where a correction from C_{bad} was applied, the expected number of T gate corrections to return to the Clifford frame (as shown in Fig. 5.6) would be very close to one since $p \lesssim \delta_P$. Thus, we conclude that the Clifford gates can be used with the scheme proposed here at a negligible cost.

We conclude this section by mentioning that Clifford frames are also useful in randomized benchmarking schemes where random Clifford gates are applied to transform a given noise channel into an effective depolarizing channel [144, 147, 148]. In these schemes, once the random Clifford gates have been applied, they must be propagated in classical software through a sequence of logical gates that are part of the quantum circuit of interest (which typically includes Clifford and T gates) and conjugate Clifford gates are subsequently applied after the Clifford frame has been restored. The techniques presented in this section can be used to restore the Clifford frame after propagation through the logical gates and could thus be an attractive approach for performing randomized benchmarking using Clifford gates. Note that if only random Pauli operators were used, the noise would be transformed to a Pauli channel (but not in general a depolarizing channel).

5.4 Summary and outlook

In this chapter, we considered performing fault-tolerant quantum computation when measurement times and/or decoding times are much slower than gate times.

This was realized by providing a new scheme for performing error correction using the Pauli frame by placing buffers after the application of a non-Clifford gate. We showed that

the Pauli frame can always be restored by applying logical Clifford corrections prior to the application of the next two-qubit or non-Clifford gate part of a quantum algorithm.

Given that logical Clifford corrections can significantly increase a code's threshold value (from Ref. [21] and Chapter 2), in the remainder of this chapter, we showed how to perform fault-tolerant error correction in the Clifford frame. We performed an in-depth study of the propagation of logical Clifford corrections through logical CNOT and T gates, and the same idea can be generalized to other universal gate sets.

For the propagation through CNOT gates, we placed buffers at strategic locations to ensure that the output Clifford corrections could be expressed in tensor product form. To achieve this, we found that on average 12 logical CNOT corrections would be required when Clifford corrections arising from buffers are chosen uniformly at random. This ensures that two-qubit Clifford corrections would not propagate through the remainder of the circuit yielding a generic Clifford correction.

We also used buffers to keep track of the Clifford corrections propagating through T gates. We showed that in certain conditions, by applying enough T gate corrections, the Clifford frame could be restored with probability arbitrarily close to 1. Furthermore, to restore the Clifford frame with probability at least $1 - \varepsilon$, we showed that the number of T gate corrections scales as $\log(1/\varepsilon)$.

While we have shown that Clifford corrections can produce a higher error threshold, the impact and applicability of Clifford corrections remains largely unexplored. Our original motivation for the current work was to determine if one of the key tricks used in FT schemes — Pauli frames — could be generalized to Clifford corrections. Having found that it can indeed be generalized at a negligible cost below threshold, we pave the way to future investigations of Clifford corrections.

Chapter 6

Conclusion

With many quantum technologies reaching a level of maturity where quantum error correction and fault-tolerance can begin to make an impact, it is important to have schemes which can cope with constraints imposed by particular architectures. For instance, some architectures might place limitations on the number of qubits that can be made available, the number of times ancilla qubits can be repeatedly measured could be limited, some gates could only be implemented via nearest neighbor interactions, measurement latencies (the time for the measurement outcome to be made available to the outside world) could be much slower than single and two-qubit gate times and much more. In this thesis, we present new schemes to address some of the mentioned constraints with the main goal of making quantum error correction fault-tolerance as accessible as possible in both near and long term implementations.

In Chapter. 2 we provided a new decoding algorithm which adapts to general Markovian noise models allowing for higher threshold values compared to more standard decoders. In particular, when applied to concatenated codes, the infidelity can be computed efficiently as a function of the number of concatenated levels allowing for exact solutions to a code's threshold value. We applied our decoding algorithm to study the performance of various codes under combined amplitude and phase-damping channels, coherent error channels and correlated noise channels. An interesting feature of our algorithm was the ability to study how non-Pauli recovery operators (and for codes which can implement non-Pauli transversal gates) could further suppress the logical noise rate in certain regimes. For instance, it was shown that when using the 5-qubit code under coherent noise, errors can be exponentially suppressed for certain rotation axes regardless of the rotation angle about the axis. Lastly, we showed that in many cases, our algorithm could still give enhanced performance when the underlying noise model was not perfectly known. We believe that

our work provides new insights into the error correcting capabilities of quantum error correcting codes. Further, we show that adapting a decoder to more realistic noise models can substantially improve a code's performance providing further motivation for tailoring decoders to noise observed in experimental settings.

Performing fault-tolerant error correction using flag qubits was first introduced by Chao and Reichardt [57] with applications to distance-three perfect codes (or perfect CSS codes). The idea of flag qubits is to catch high weight errors arising from fewer faults in the syndrome extraction circuits. Further, flag qubits allow fault-tolerant error correction protocols to be implemented with very small qubit overhead. Using flag qubits, in Chapter 3 we developed a new fault-tolerant error correction protocol which can be applied to arbitrary distance codes. There are several reasons why such generalizations are important. For instance, it was shown that fault-tolerant error correction using Shor error correction (reviewed in Sec. 1.2.1) combined with LDPC (low density parity check) codes satisfying certain properties can achieve constant overhead [54]. However, our new fault-tolerant error correction protocol uses fewer qubits than Shor error correction and does not require the verification of ancilla qubits. Additionally, we showed how fault-tolerant error correction could be performed using fewer syndrome measurement repetitions compared to previous methods used in Shor error correction. We gave a sufficient condition which makes it easier to verify if a code family satisfies the condition for our flag-fault-tolerant error correction scheme. We proved that the family of surface codes, color codes with a hexagonal lattice and quantum Reed-Muller codes satisfy the sufficient condition and provided syndrome extraction circuits for each code family. Lastly, we believe our protocol could be suitable for early fault-tolerant experiments which only have access to a very small number of qubits.

The decoding schemes presented in Chapter 2 required some knowledge of the noise model afflicting the quantum system and assumed that gates and measurements could be implemented perfectly. In realistic quantum devices, fully characterizing the noise model can be a significant challenge. Furthermore, gates and measurements can introduce additional errors into the system. In Chapter 4, we used state of the art machine learning techniques to find improved decoders in low noise rate fault-tolerant regimes which could be relevant for early fault-tolerant experiments. Our techniques require no knowledge of the noise model and we showed how they could be implemented in the lab. The fault-tolerant error correction schemes considered were the surface code as well as Steane and Knill error correction. For both distance-three and five codes, we showed improvements in the pseudo-thresholds that were obtained under the same depolarizing noise channel as the one used in Chapter 3. We believe that even bigger improvements could be obtained if more realistic noise models were considered. We further gave estimates of the classical resources that would be required in order to perform active error correction. However,

neural decoders do have significant limitations. First, training a neural network becomes much harder (and requires more experiments) as the physical noise rate becomes smaller. The reason is that the machine learning techniques developed so far do not work well with sparse data. In particular, we believe that it is unrealistic to train a neural-network at large noise rates and apply the trained network to lower noise rates since in reality, the neural-network would be trained with the device given by the experimentalist. Another issue is that in order to achieve good performance, the size of the training set needs to grow significantly for higher distance codes. Hence we believe that neural decoders could be relevant in early fault-tolerant experiments, but large scale quantum computations will require methods that are scalable. Nevertheless, we believe important decoding insights could be obtained with trained neural networks applied to small distance codes and could help guide more scalable solutions.

In Chapter 5 we considered the problem of slow error diagnostics, which incorporates both long measurement latencies as well as slow classical decoding times of error syndromes. The problem of slow error diagnostics has been considered previously in the context of Pauli frames [44, 20]. We first provided a generalization of the previous methods which apply regardless of how non-Clifford gates are implemented and to any family of stabilizer codes. In scenarios where non-Pauli recovery operators are used, one cannot use Pauli frames to keep track of errors in classical software. Motivated by the improved thresholds obtained via non-Pauli recover operators shown in Chapter 2, we developed a theory of Clifford frame error correction and proved that it can be implemented with negligible overhead in fault-tolerant noise rate regimes. Further, we believe our scheme can have applications in other areas such as randomized benchmarking and where Clifford frames are used as an accessory in measurement-based quantum computing.

References

[1] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. Perfect quantum error correcting code. *Phys. Rev. Lett.*, 77:198–201, Jul 1996.

[2] Multiple-particle interference and quantum error correction. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996.

[3] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:2493, 1995.

[4] Yu Tomita and Krysta M. Svore. Low-distance surface codes under realistic quantum noise. *Phys. Rev. A*, 90:062320, Dec 2014.

[5] H. Bombin and M. A. Martin-Delgado. Topological quantum distillation. *Phys. Rev. Lett.*, 97:180501, Oct 2006.

[6] Adam Paetznick and Ben W. Reichardt. Fault-tolerant ancilla preparation and noise threshold lower bounds for the 23-qubit golay code. *Quant. Inf. Compt.*, 12:1034–1080, 2011.

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

[8] P.W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. *Proceedings., 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[9] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[10] D. Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997.

[11] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. *Proceedings of Symposia in Applied Mathematics*, 68:13–58, 2010.

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

[13] Andrew W. Steane. Enlagement of calderbank-shor-steane quantum codes. *IEEE. Trans.Inform. Theory*, 45(7):2492–2495, 1999.

[14] Héctor Bombín. Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes. *New J. Phys.*, 17(8):083002, 2015.

[15] Aleksander Kubica and Michael E. Beverland. Universal transversal gates with color codes: A simplified approach. *Phys. Rev. A*, 91:032330, 2015.

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

[17] Peter W. Shor. Fault-tolerant quantum computation. *Proceedings., 37th Annual Symposium on Foundations of Computer Science*, pages 56–65, 1996.

[18] Panos Aliferis, Daniel Gottesman, and John Preskill. Quantum accuracy threshold for concatenated distance-3 codes. *Quantum Info. Comput.*, 6(2):97–165, March 2006.

[19] Tomas Jochym-O'Connor and Raymond Laflamme. Using concatenated quantum codes for universal fault-tolerant quantum gates. *Phys. Rev. Lett.*, 112:010505, 2014.

[20] David P. DiVincenzo and Panos Aliferis. Effective fault-tolerant quantum computation with slow measurements. *Phys. Rev. Lett.*, 98:020501, Jan 2007.

[21] Christopher Chamberland, Joel J. Wallman, Stefanie Beale, and Raymond Laflamme. Hard decoding algorithm for optimizing thresholds under general markovian noise. *Phys. Rev. A*, 95:042332, 2017.

[22] P. Iyer and D. Poulin. Hardness of decoding quantum stabilizer codes. *IEEE Transactions on Information Theory*, 61(9):5209–5223, Sept 2015.

[23] Daniel Gottesman. The heisenberg representation of quantum computers, talk at. In *International Conference on Group Theoretic Methods in Physics*. Citeseer, 1998.

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

[25] Mauricio Gutiérrez and Kenneth R. Brown. Comparison of a quantum error-correction threshold for exact and approximate errors. *Phys. Rev. A*, 91:022335, Feb 2015.

[26] Mauricio Gutiérrez, Conor Smith, Livia Lulushi, Smitha Janardan, and Kenneth R. Brown. Errors and pseudothresholds for incoherent and coherent noise. *Phys. Rev. A*, 94:042338, Oct 2016.

[27] Michael R. Geller and Zhongyuan Zhou. Efficient error models for fault-tolerant architectures and the pauli twirling approximation. *Phys. Rev. A*, 88:012314, Jul 2013.

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

[29] David Poulin. Optimal and efficient decoding of concatenated quantum block codes. *Phys. Rev. A*, 74:052333, Nov 2006.

[30] Jesse Fern. Correctable noise of quantum-error-correcting codes under adaptive concatenation. *Phys. Rev. A*, 77:010301, Jan 2008.

[31] Andrew S. Darmawan and David Poulin. Tensor-network simulations of the surface code under realistic noise. *Phys. Rev. Lett.*, 119:040502, Jul 2017.

[32] K Kraus. *States, Effects, and Operations*. Springer-Verlag Berlin Heidelberg, 1983.

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

[34] Panos Aliferis, Daniel Gottesman, and John Preskill. Accuracy threshold for post-selected quantum computation. *Quant. Inf. Comput.*, 8:181–244, 2008.

[35] Christopher Chamberland, Tomas Jochym-O'Connor, and Raymond Laflamme. Thresholds for universal concatenated quantum codes. *Phys. Rev. Lett.*, 117:010501, 2016.

[36] Christopher Chamberland, Tomas Jochym-O'Connor, and Raymond Laflamme. Overhead analysis of universal concatenated quantum codes. *Phys. Rev. A*, 95:022313, 2017.

[37] Veronika Baumann and Časlav Brukner. Appearance of causality in process matrices when performing fixed-basis measurements for two parties. *Phys. Rev. A*, 93:062324, Jun 2016.

[38] Shelby Kimmel, Marcus P. da Silva, Colm A. Ryan, Blake R. Johnson, and Thomas Ohki. Robust extraction of tomographic information via randomized benchmarking. *Phys. Rev. X*, 4:011050, Mar 2014.

[39] Carlo Cafaro and Peter van Loock. Approximate quantum error correction for generalized amplitude-damping errors. *Phys. Rev. A*, 89:022316, Feb 2014.

[40] Joseph Emerson, Marcus Silva, Osama Moussa, Colm Ryan, Martin Laforest, Jonathan Baugh, David G. Cory, and Raymond Laflamme. Symmetrized characterization of noisy quantum processes. *Science*, 317(5846):1893–1896, 2007.

[41] Richard Kueng, David M. Long, Andrew C. Doherty, and Steven T. Flammia. Comparing experiments to the fault-tolerance threshold. *Phys. Rev. Lett.*, 117:170502, Oct 2016.

[42] Christopher Chamberland and Michael E. Beverland. Flag fault-tolerant error correction with arbitrary distance codes. *Quantum*, 2:53, February 2018.

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

[44] E. Knill. Scalable quantum computing in the presence of large detected-error rates. *Phys. Rev. A*, 71:042322, Apr 2005.

[45] Sergey Bravyi and Alexei Kitaev. Quantum codes on a lattice with boundary. *arXiv:quant-ph/9811052*, 1998.

[46] Eric Dennis, Alexei Kitaev, Andrew Landhal, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43:4452–4505, 2002.

[47] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 176–188. ACM, 1997.

[48] John Preskill. Reliable quantum computers. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1969):385–410, 1998.

[49] Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. Resilient quantum computation. *Science*, 279:342–345, 1998.

[50] David S. Wang, Austin G. Fowler, and Lloyd C. L. Hollenberg. Surface code quantum computing with error rates over 1 *Phys. Rev. A*, 83:020302, Feb 2011.

[51] Robert G. Gallager. *Low density parity check codes*. PhD thesis, MIT, 1960.

[52] Alexey A. Kovalev and Leonid P. Pryadko. Fault tolerance of quantum low-density parity check codes with sublinear distance scaling. *Phys. Rev. A*, 87:020304, Feb 2013.

[53] J. P. Tillich and G. Zmor. Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Transactions on Information Theory*, 60(2):1193–1202, Feb 2014.

[54] Daniel Gottesman. Fault-tolerant quantum computation with constant overhead. *arXiv: quant-ph/1310.2984*, 2013.

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

[56] Jesse Fern. An upper bound on quantum fault tolerant thresholds. *arXiv:quant-ph/0801.2608*, 2008.

[57] Rui Chao and Ben W. Reichardt. Quantum error correction with only two extra qubits. *Phys. Rev. Lett.*, 121:050502, Aug 2018.

[58] Rui Chao and Ben W. Reichardt. Fault-tolerant quantum computation with few qubits. *arXiv:quant-ph/1705.05365*, 2017.

[59] Theodore J. Yoder and Isaac H. Kim. The surface code with a twist. *Quantum*, 1:2, April 2017.

[60] Panos Aliferis and Andrew Cross. Subsystem fault tolerance with the bacon-shor code. *Phys. Rev. Lett.*, 98:220502, 2007.

[61] A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2 – 30, 2003.

[62] Xiao-Gang Wen. Quantum orders in an exact soluble model. *Phys. Rev. Lett.*, 90:016803, Jan 2003.

[63] Jonas T. Anderson, Guillaume Duclos-Cianci, and David Poulin. Fault-tolerant conversion between the steane and reed-muller quantum codes. *Phys. Rev. Lett.*, 113:080501, 2014.

[64] Theerapat Tansuwannont, Christopher Chamberland, and Debbie Leung. Flag fault-tolerant error correction for cyclic css codes. *arXiv:quant-ph/1803.009758*, 2018.

[65] Andrew J. Landahl, Jonas T. Anderson, and Patrick R. Rice. Fault-tolerant quantum computing with color codes. *arXiv:1108.5738*, 2011.

[66] Hayato Goto. Minimizing resource overheads for fault-tolerant preparation of encoded states of the steane code. *Scientific Reports*, (6):19578, 2016.

[67] Christopher Chamberland and Pooya Ronagh. Deep neural decoders for near term fault-tolerant experiments. *Quantum Science and Technology*, 3(4):044002, 2018.

[68] Intel Press Kit. https://newsroom.intel.com/press-kits/quantum-computing/. Accessed: 2018-02-16.

[69] IBM Q Experience. https://quantumexperience.ng.bluemix.net/qx/devices. Accessed: 2018-02-16.

[70] Riggeti QPU. http://pyquil.readthedocs.io/en/latest/qpu_overview.html. Accessed: 2018-02-16.

[71] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, R. Barends, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. Fowler, B. Foxen, R. Graff, E. Jeffrey, J. Kelly, E. Lucero, A. Megrant, J. Mutus, M. Neeley, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, H. Neven, and J. M. Martinis. A blueprint for demonstrating quantum supremacy with superconducting qubits. *ArXiv e-prints*, September 2017.

[72] Christophe Vuillot. Is error detection helpful on ibm 5q chips ? *arXiv:quant-ph/1705.08957*, 2017.

[73] Emanuel Knill. Fault-tolerant postselected quantum computation: schemes. *arXiv:quant-ph/0402171*, 2004.

[74] Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. Threshold accuracy for quantum computation. *arXiv: quant-ph/9610011*, 1996.

[75] Adam Paetznick and Ben W. Reichardt. Universal fault-tolerant quantum computation with only transversal gates and error correction. *Phys. Rev. Lett.*, 111:090505, Aug 2013.

[76] Héctor Bombín. Gauge color codes: optimal transversal gates and gauge fixing in topological stabilizer codes. *New J. Phys.*, 18:043038, 2016.

[77] Theodore J. Yoder, Ryuji Takagi, and Isaac L. Chuang. Universal fault-tolerant gates on concatenated stabilizer codes. *Phys. Rev. X*, 6:031039, Sep 2016.

[78] Andrew S. Darmawan and David Poulin. An efficient general decoding algorithm for the surface code. *arXiv:1801.01879*, 2018.

[79] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

[80] Austin G. Fowler, Adam C. Whiteside, Angus L. McInnes, and Alimohammad Rabbani. Topological code autotune. *Phys. Rev. X*, 2:041003, Oct 2012.

[81] Giacomo Torlai and Roger G. Melko. Neural decoder for topological codes. *Phys. Rev. Lett.*, 119:030501, Jul 2017.

[82] S. Krastanov and L. Jiang. Deep neural network probabilistic decoder for stabilizer codes. *Scientific Reports*, (7):11003, 2017.

[83] Savvas Varsamopoulos, Ben Criger, and Koen Bertels. Decoding small surface codes with feedforward neural networks. *Quantum Science and Technology*, 3(1):015004, 2018.

[84] Paul Baireuther, Thomas E. O'Brien, Brian Tarasinski, and Carlo W. J. Beenakker. Machine-learning-assisted correction of correlated qubit errors in a topological code. *Quantum*, 2:48, January 2018.

[85] Nikolas P. Breuckmann and Xiaotong Ni. Scalable neural network decoders for higher dimensional quantum codes. *arXiv:quant-ph/1710.09489*, 2017.

[86] Nishad Maskara, Aleksander Kubica, and Tomas Jochym-O'Connor. Advantages of versatile neural-network decoding for topological codes. *arXiv:1802.08680*, 2018.

[87] Austin G. Fowler. Minimum weight perfect matching of fault-tolerant topological quantum error correction in average o(1) parallel time. *Quantum Info. Comput.*, 15(1-2):145–158, January 2015.

[88] Austin G. Fowler, Adam C. Whiteside, and Lloyd C. L. Hollenberg. Towards practical classical processing for the surface code: Timing analysis. *Phys. Rev. A*, 86:042313, Oct 2012.

[89] Guillaume Duclos-Cianci and David Poulin. Fast decoders for topological quantum codes. *Phys. Rev. Lett.*, 104:050504, Feb 2010.

[90] Guillaume Duclos-Cianci and David Poulin. Fault-tolerant renormalization group decoder for abelian topological codes. *Quant. Inf. Comput.*, 14(9 & 10):0721–0740), 2014.

[91] Sergey Bravyi and Jeongwan Haah. Quantum self-correction in the 3d cubic code model. *Phys. Rev. Lett.*, 111:200501, Nov 2013.

[92] James R. Wootton and Daniel Loss. High threshold error correction for the surface code. *Phys. Rev. Lett.*, 109:160503, Oct 2012.

[93] Nicolas Delfosse and Naomi H. Nickerson. Almost-linear time decoding algorithm for topological codes. *arXiv:quant-ph/1709.06218*, 2017.

[94] J. Conrad, C. Chamberland, N. P. Breuckmann, and B. M. Terhal. The small stellated dodecahedron code and friends. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 376(2123), 2018.

[95] Barbara M. Terhal. Quantum error correction for quantum memories. *Rev. Mod. Phys.*, 87:307–346, Apr 2015.

[96] Christopher Chamberland, Pavithran Iyer, and David Poulin. Fault-tolerant quantum computing in the Pauli or Clifford frame with slow error diagnostics. *Quantum*, 2:43, January 2018.

[97] P. Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. On universal and fault-tolerant quantum computing: A novel basis and a new constructive proof of universality for shor's basis. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 486–494. IEEE, 1999.

[98] Panos Aliferis and Barbara M. Terhal. Fault-tolerant quantum computation for local leakage faults. *Quant. Inf. Comput.*, 7:139–156, 2007.

[99] Carlos Mochon. Anyon computers with smaller groups. *Phys. Rev. A*, 69:032306, Mar 2004.

[100] Yasunari Suzuki, Keisuke Fujii, and Masato Koashi. Efficient simulation of quantum error correction under coherent error based on the nonunitary free-fermionic formalism. *Phys. Rev. Lett.*, 119:190503, Nov 2017.

[101] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[102] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information science and statistics. Springer, 2013.

[103] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. *Efficient BackProp*, pages 9–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[104] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[105] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[106] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *Proceedings of the 17th International Conference on Artificial Neural Networks*, ICANN'07, pages 220–229, Berlin, Heidelberg, 2007. Springer-Verlag.

[107] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[108] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling, 2016.

[109] D. Gillick, C. Brunk, O. Vinyals, and A. Subramanya. Multilingual Language Processing From Bytes. *ArXiv e-prints*, November 2015.

[110] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[111] Jurgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649, Washington, DC, USA, 2012. IEEE Computer Society.

[112] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[113] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization*, 19(4):1574–1609, January 2009.

[114] Léon Bottou and Yann L. Cun. Large scale online learning. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 217–224. MIT Press, 2004.

[115] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[116] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[117] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. Technical report, 2012.

[118] J. Mockus. *Bayesian approach to global optimization: theory and applications*. Mathematics and its applications (Kluwer Academic Publishers).: Soviet series. Kluwer Academic, 1989.

[119] Ruben Martinez-Cantin. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, 15:3915–3919, 2014.

[120] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[121] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, March 2008.

[122] G Wendin. Quantum information processing with superconducting circuits: a review. *Reports on Progress in Physics*, 80(10):106001, 2017.

[123] IBM QISKit. https://github.com/QISKit/ibmqx-backend-information/tree/master/backends/ibmqx3. Accessed: 2018-03-27.

[124] Norbert M. Linke, Dmitri Maslov, Martin Roetteler, Shantanu Debnath, Caroline Figgatt, Kevin A. Landsman, Kenneth Wright, and Christopher Monroe. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, 2017.

[125] M. Reiher, N. Wiebe, K. M. Svore, D. Wecker, and M. Troyer. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Science*, 114:7555–7560, July 2017.

[126] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. Vazir Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan,

R. Walter, W. Wang, E. Wilcox, and D. H. Yoon. In-Datacenter Performance Analysis of a Tensor Processing Unit. *ArXiv e-prints*, April 2017.

[127] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.

[128] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic. *Digital Integrated Circuits*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2008.

[129] Behrooz Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, Oxford, UK, 2000.

[130] G. Bewick, P. Song, G. De Micheli, and M. J. Flynn. Approaching a nanosecond: a 32 bit adder. In *Proceedings 1988 IEEE International Conference on Computer Design: VLSI*, pages 221–226, Oct 1988.

[131] S. Naffziger. A sub-nanosecond 0.5 /spl mu/m 64 b adder design. In *1996 IEEE International Solid-State Circuits Conference. Digest of TEchnical Papers, ISSCC*, pages 362–363, Feb 1996.

[132] Wayne Wolf. *FPGA-Based System Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[133] Shanzhen Xing and W.W.H. Yu. Fpga adders: Performance evaluation and optimal design. 15:24 – 29, 02 1998.

[134] S. Hauck, M. M. Hosler, and T. W. Fry. High-performance carry chains for fpga's. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(2):138–147, April 2000.

[135] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500–503, 2014.

[136] Evan Jeffrey, Daniel Sank, J. Y. Mutus, T. C. White, J. Kelly, R. Barends, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. Megrant, P. J. J. O'Malley, C. Neill, P. Roushan, A. Vainsencher, J. Wenner, A. N. Cleland, and John M. Martinis. Fast accurate state measurement with superconducting qubits. *Phys. Rev. Lett.*, 112:190504, May 2014.

[137] J. R. Petta, A. C. Johnson, J. M. Taylor, E. A. Laird, A. Yacoby, M. D. Lukin, C. M. Marcus, M. P. Hanson, et al. Coherent manipulation of coupled electron spins in semiconductor quantum dots. *Science*, 309(5744):2180–2184, 2005.

[138] M. Veldhorst, J. C. C. Hwang, C H. Yang, A. W. Leenstra, B. de Ronde, J. P. Deholla, J. T. Muhonen, F. E. Hudson, K. M. Itoh, A. Morella, and A. S. Dzurak. An addressable quantum dot qubit with fault-tolerant control-fidelity. *Nature nanotechnology*, 9(12):981–985, 2014.

[139] J. Stehlik, Y.-Y. Liu, C. M. Quintana, C. Eichler, T. R. Hartke, and J. R. Petta. Fast charge sensing of a cavity-coupled double quantum dot using a josephson parametric amplifier. *Phys. Rev. Applied*, 4:014018, Jul 2015.

[140] S. Olmschenk, D. Hayes, D. N. Matsukevich, P. Maunz, D. L. Moehring, K. C. Younge, and C. Monroe. Measurement of the lifetime of the $6p\ ^2P^o_{1/2}$ level of yb^+. *Phys. Rev. A*, 80:022502, Aug 2009.

[141] H. Häffner, C.F. Roos, and R. Blatt. Quantum computing with trapped ions. *Physics Reports*, 469(4):155 – 203, 2008.

[142] S.J. Devitt, A.G. Fowler, T. Tilma, W.J. Munro, and K. Nemoto. Classical processing requirements for a topological quantum computing system. *Int. J. Quant. Inf.*, 8:1–27, 2010.

[143] Torsten Karzig, Christina Knapp, Roman M. Lutchyn, Parsa Bonderson, Matthew B. Hastings, Chetan Nayak, Jason Alicea, Karsten Flensberg, Stephan Plugge, Yuval Oreg, Charles M. Marcus, and Michael H. Freedman. Scalable designs for quasiparticle-poisoning-protected topological quantum computation with majorana zero modes. *Phys. Rev. B*, 95:235305, Jun 2017.

[144] Joel J. Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Phys. Rev. A*, 94:052325, Nov 2016.

[145] Thomas Koshy. *Catalan Numbers with Applications*. Oxford University Press, Oxford, England, 2008.

[146] George E Andrews, Richard Askey, and Ranjan Roy. Special functions (encyclopedia of mathematics and its applications vol 71), 1999.

[147] Easwar Magesan, J. M. Gambetta, and Joseph Emerson. Scalable and robust randomized benchmarking of quantum processes. *Phys. Rev. Lett.*, 106:180504, May 2011.

[148] Easwar Magesan, Jay M. Gambetta, and Joseph Emerson. Characterizing quantum gates via randomized benchmarking. *Phys. Rev. A*, 85:042311, Apr 2012.

[149] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error rate. *SIAM Journal on Computing*, 38(4):1207–1282, 2008.

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

[151] Vladimir Kolmogorov. Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, 2009.

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

[153] A. Jamiołkowski. Linear transformations which preserve trace and positive semidefiniteness of operators. *Reports on Mathematical Physics*, 3(4):275 – 278, 1972.

[154] Christopher J. Wood, Jacob D. Biamonte, and David G. Cory. Tensor networks and graphical calculus for open quantum systems. *Quantum Info. Comput.*, 15(9-10):759–811, July 2015.

APPENDICES

Appendix A

Quantum error correction appendix

A.1 Obtaining the α and β coefficients in closed form

In this section we provide an alternative derivation of the α and β coefficients found in Eq. 2.18 and Eq. 2.19. The latter coefficients will be given in terms of the symplectic vector representation of Pauli operators. For the bit strings $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, we will write

$$Z(a)X(b) = (Z^{a_1} \otimes \ldots \otimes Z^{a_n})(X^{b_1} \otimes \ldots \otimes X^{b_n}). \tag{A.1}$$

Since the α coefficient is related to the overall sign of the operator $S_k\overline{\tau}$ (see Eq. 2.6 and Eq. 2.17), the goal is to obtain an expression relating the overall sign of $S_k\overline{\tau}$ to its symplectic vector representation. An operator $S_k \in \mathcal{S}$ can always be written as a product of the codes stabilizer generators so that

$$S_k = g_{j_1} \ldots g_{j_k}, \tag{A.2}$$

where

$$g_{j_i} = Z(a_{j_i})X(b_{j_i}). \tag{A.3}$$

Defining

$$\overline{a} = a_{j_1} + \ldots + a_{j_k} \pmod 2 \tag{A.4}$$

$$\overline{b} = b_{j_1} + \ldots + b_{j_k} \pmod 2, \tag{A.5}$$

we commute all the Z operators in Eq. A.2 to the left, allowing us to write S_k as in Eq. A.1

$$S_k = (-1)^{f(a_{j_1},\ldots,a_{j_k};b_{j_1},\ldots,b_{j_k})} Z(\overline{a})X(\overline{b}). \tag{A.6}$$

The overall sign can be obtained from the function f, which is given by

$$f(a_{j_1},\ldots,a_{j_k};b_{j_1},\ldots,b_{j_k}) = \sum_{l=1}^{k-1}\sum_{t=l+1}^{k} b_{j_l}a_{j_t}. \tag{A.7}$$

Writing the logical Pauli operator $\overline{\tau}$ as

$$\overline{\tau} = Z(\tau_z)X(\tau_x), \tag{A.8}$$

$S_k\overline{\tau}$ can then be written as

$$S_k\overline{\tau} = (-1)^{f(a_{j_1},\ldots,a_{j_k};b_{j_1},\ldots,b_{j_k})+\overline{b}\cdot\tau_z} Z(\overline{a}+\tau_z)X(\overline{b}+\tau_x). \tag{A.9}$$

For any $\mu_j \in \{I, X, Y, Z\}$, we can write μ_j in terms of X and Z Pauli operators:

$$\mu_j = (-i)^{a_j b_j} Z^{a_j} X^{b_j}, \tag{A.10}$$

allowing us to write

$$Z(\overline{a}+\tau_z)X(\overline{b}+\tau_x) = i^{(\overline{a}+\tau_z)\cdot(\overline{b}+\tau_x)}\mu_1 \otimes \ldots \otimes \mu_n. \tag{A.11}$$

It is important to note that the dot product in the factor of i is *not* added modulo 2.

Using Eq. A.11, we have

$$S_k\overline{\tau} = (-1)^{f(a_{j_1},\ldots,a_{j_k};b_{j_1},\ldots,b_{j_k})+\overline{b}\cdot\tau_z} i^{(\overline{a}+\tau_z)\cdot(\overline{b}+\tau_x)}\mu_1 \otimes \ldots \otimes \mu_n. \tag{A.12}$$

Since the α coefficient takes into account the overall sign of the product between elements in the stabilizer group and the logical operators, we have

$$\alpha^{\tau}_{\phi(S_k\overline{\tau})} = \frac{1}{2^{\frac{n}{2}-1}}(-1)^{f(a_{j_1},\ldots,a_{j_k};b_{j_1},\ldots,b_{j_k})+\overline{b}\cdot\tau_z} i^{(\overline{a}+\tau_z)\cdot(\overline{b}+\tau_x)}, \tag{A.13}$$

where the normalization factor arises from choosing a trace orthonormal basis in the sum of E_τ.

Given a recovery map R_l for the syndrome measurement l, we can write it in terms of its symplectic vector representation as

$$R_l = Z(a_l)X(b_l). \tag{A.14}$$

From Eq. 2.15 and Eq. 2.17, the β coefficient corresponding to the recovery map R_l can be obtained by commuting R_l to the left of $S_k\overline{\tau}$ and using $R_l^\dagger R_l = I$. Doing so, we find that

$$\beta^{\tau}_{\phi(S_k\overline{\tau})}(R_l) = \alpha^{\tau}_{\phi(S_k\overline{\tau})}(-1)^{a_l\cdot(\overline{b}+\tau_x)+b_l\cdot(\overline{a}+\tau_z)}. \tag{A.15}$$

Appendix B

Fault-tolerant error correction appendices

B.1 Proof that the flag t-FTEC protocol satisfies the fault-tolerance criteria of Def. 5

Consider the flag t-FTEC protocol described in Sec. 3.4.1.

Claim 1. *If the flag t-FTEC condition is satisfied, then both fault-tolerance criteria of Def. 5 will be satisfied.*

Proof. First note that the protocol always terminates. As was shown in the arguments leading to Eq. 3.5 presented in Sec. 3.4.1, the maximum number of syndrome measurement rounds is $\frac{1}{2}(t^2 + 3t + 2)$.

To prove fault-tolerance, in what follows we assume that there are at most t-faults during the protocol. Also, we define a benign fault to be a fault that either leaves all syndrome measurements in the protocol unchanged.

By repeating the syndrome measurement using t-flag circuits, the following cases exhaust all possible errors for the occurrence of at most t faults.

Case 1: The same syndrome is measured $t - n_{diff} + 1$ times in a row and there are no flags.

At any time during the protocol, if there are no flags, there can be at most $t - n_{\text{diff}}$ remaining faults that occur (since it is guaranteed that there were at least n_{diff} faults).

Therefore, if the same syndrome was measured $t - n_{\text{diff}} + 1$ times in a row, at least one round (say r) had to have been fault-free yielding the correct syndrome corresponding to the data qubit errors present at that time. Applying $E_{\min}(s)$ will remove those errors. Furthermore, since all syndrome measurements are identical and there are no flags, there can be at most $t - n_{\text{diff}}$ errors which are introduced on the data blocks from faults during the $t - n_{\text{diff}} + 1$ syndrome measurement rounds (excluding round r). Since none of the errors change the syndrome, after applying the correction, the output state can differ from the input codeword by an error of weight at most $t - n_{\text{diff}}$ (if the total number of faults and input errors was t). For input states afflicted by an error of arbitrary weight, the output state will differ from a valid codeword (but not necessarily the input codeword) by an error of weight at most $t - n_{\text{diff}}$. Thus both conditions of Def. 5 are satisfied.

Case 2: There are no flags and $n_{diff} = t$.

The only way that $n_{\text{diff}} = t$ is if there were t-faults that each changed the syndrome measurement outcome. Further since there were no flags, an error E afflicting the data qubits must satisfy $\text{wt}(E) \leq t$. Thus repeating the syndrome measurement using non-flag circuits will correctly identify and remove the error in the case where the number of input errors and faults is t or project the system back to the code space (to a possibly different codeword) if there were t faults and the input state was afflicted by an error of arbitrary weight .

Case 3: A set of t circuits $\{C(g_{i_1}), \cdots, C(g_{i_t})\}$ flagged.

Since t circuits $\{C(g_{i_1}), \cdots, C(g_{i_t})\}$ flagged, then no other faults can occur during the protocol. Hence, when repeating the syndrome measurement using non-flag circuits, the measured syndrome will correspond to an error $E_r \in \tilde{E}_t^t(g_{i_1}, \cdots g_{i_t}, s)$. Since from the flag t-FTEC condition all elements of $\tilde{E}_t^t(g_{i_1}, \cdots g_{i_t}, s)$ are logically equivalent, the product of errors resulting from the flag circuits $\{C(g_{i_1}), \cdots, C(g_{i_t})\}$ will be corrected.

Note that for an input error E_{in} of arbitrary weight and since the final round must be error free, applying a correction a correction from the set $\tilde{E}_t^t(g_{i_1}, \cdots g_{i_t}, s)$ is guaranteed to return the system to the codespace. Thus both conditions of Def. 5 are satisfied.

Case 4: The m circuits $\{C(g_{i_1}), \cdots, C(g_{i_m})\}$ flagged with $1 \leq m < t$, $n_{diff} = t - m$.

Here we can assume that at any point during the protocol and after the j'th flag, the syndrome never repeated more than $t - j - n_{\text{diff}}$ times. Otherwise case 5 of the protocol would already have occurred.

As m circuits $\{C(g_{i_1}), \cdots, C(g_{i_m})\}$ have flagged and $n_{\text{diff}} = t - m$, then there can be no more faults. The final syndrome measurement using non-flag circuits will yield a syndrome corresponding to an error in the set $\tilde{E}_t^m(g_{i_1}, \cdots g_{i_m}, s)$ (and all elements are

logically equivalent from the flag t-FTEC condition). Applying a recovery operator chosen from this set will thus remove the errors afflicting the data. If the input state differs from a valid codeword by an error of arbitrary weight, by definition of $\tilde{E}_t^m(g_{i_1}, \cdots g_{i_m}, s)$ the output state will be a valid codeword.

Case 5: The m circuits $\{C(g_{i_1}), \cdots, C(g_{i_m})\}$ flagged with $1 \leq m < t$, $n_{same} = t - m - n_{diff} + 1$.

Given that m circuits $\{C(g_{i_1}), \cdots, C(g_{i_m})\}$ flagged, there are r remaining faults that don't result in a flag with $n_{\text{diff}} \leq r \leq t - m$. In this case, after the m'th flag, the syndrome measurement was repeated using t-flag circuits $t - m - n_{\text{diff}} + 1$ times in a row and all syndromes were the same. It is thus guaranteed that at least one of the syndrome measurements s was fault-free and correctly identified the errors arising from the flags and errors causing the syndrome to change giving n_{diff} (along with some error E which did not cause the circuits to flag with $\text{wt}(E) \leq t - m - n_{\text{diff}}$). Consequently, if there are no errors on the input state, the overall error on the data will be EE_r with $E_r \in \bigcup_{j=0}^{t-m-n_{\text{diff}}} \tilde{E}_t^{t-j-n_{\text{diff}}}(g_{i_1}, \cdots, g_{i_m}, s)$. Since all elements in $\bigcup_{j=0}^{t-m-n_{\text{diff}}} \tilde{E}_t^{t-j-n_{\text{diff}}}(g_{i_1}, \cdots, g_{i_m}, s)$ are logically equivalent from the flag t-FTEC condition, by choosing a correction from this set, the output state can differ from the input codeword by an error of at most weight $t - m - n_{\text{diff}}$.

If there is an input error of arbitrary weight, then again one of the $t - m - n_{\text{diff}} + 1$ rounds will have the correct syndrome s. The actual state of the data qubits after the protocol can differ from the state which had the correct syndrome by an error of weight at most $t - m - n_{\text{diff}}$. Therefore applying any correction with syndrome s will return the system to the code space up to an error of weight at most $t - m - n_{\text{diff}}$.

□

B.2 Fault-tolerant state preparation and measurement using flag t-FTEC

In this section we show how to fault-tolerantly prepare a logical $|\overline{0}\rangle$ state and how to perform fault-tolerant measurements for codes that satisfy the flag t-FTEC condition of Sec. 3.4. Note that there are several methods that can be used for doing so. Here we follow a procedure similar to that shown in [11] when performing Shor EC. However, compared to Shor EC, the flag t-FTEC protocol requires fewer qubits. Furthermore, postselection is not necessary.

Consider an n-qubit stabilizer code C with stabilizer group $\mathcal{S} = \langle g_1, \cdots, g_{n-k} \rangle$ that can correct up to t errors. Notice that the encoded $|\overline{0}\rangle$ state is a $+1$ eigenstate of the logical $\overline{Z}$ operator and all of the codes stabilizer generators. For k encoded qubits, $|\overline{0}\rangle$ would be $+1$ eigenstate of $\{\overline{Z}_1, \cdots \overline{Z}_k\}$ and all of the codes stabilizers. For notational simplicity, in what follows we assume $k = 1$.

The state $|\overline{0}\rangle$ is a stabilizer state completely specified by the full stabilizer generators of $\mathcal{S}$ and $\overline{Z}$. We can think of $\mathcal{S}' = \langle g_1, \cdots g_{n-1}, \overline{Z} \rangle$ as a stabilizer code with zero encoded qubits and a $2^0 = 1$ dimensional Hilbert space. Thus any state which is a $+1$ eigenstate of all operators in $\mathcal{S}'$ will correspond to the encoded $|\overline{0}\rangle$ state.

Now, suppose we prepare $|\overline{0}\rangle_{\text{in}}$ using a non-fault-tolerant encoding and perform a round of flag t-FTEC using the extended stabilizers $\langle g_1, \cdots g_{n-1}, \overline{Z} \rangle$. Then by the second criteria of Def. 5, the output state $|\overline{0}\rangle_{\text{out}}$ is guaranteed to be a valid codeword with at most t single-qubit errors. But for the extended stabilizers $\langle g_1, \cdots g_{n-1}, \overline{Z} \rangle$ there is only *one* valid codeword which corresponds to the encoded $|\overline{0}\rangle$ state. In fact, by the second criteria of Def. 5, any n-qubit input state prepared using non-fault-tolerant circuits is guaranteed to be an encoded $|\overline{0}\rangle$ state if there are no more than t faults in the EC round.

We point out that the flag t-FTEC condition of Sec. 3.4.1 is trivially satisfied for $\mathcal{S}'$ since the codes logical operators are now stabilizers. In other words, if two errors belong to the set $\tilde{E}_t^m(g_{i_1}, \cdots, g_{i_k}, s)$, then their product will always be a stabilizer. Therefore, the flag t-FTEC protocol can always be applied for the code $\mathcal{S}'$.

To summarize, the encoded $|\overline{0}\rangle$ state can be prepared by first preparing any n-qubit state using non-fault-tolerant circuits followed by applying a round of flag t-FTEC using the extended stabilizers $\langle g_1, \cdots g_{n-1}, \overline{Z} \rangle$. This guarantees that the output state will be the encoded $|\overline{0}\rangle$ state with at most t single-qubit errors.

Now suppose we want to measure the eigenvalue of a logical operator $\overline{P}$ where P is a Pauli. If C is a CSS code and the logical operator being measured is X or Z, one could measure the eigenvalue by performing the measurement transversally. So suppose C is not a CSS code. From [11] we require that performing a measurement with s faults on an input state with r errors ($r + s \leq t$) is equivalent to correcting the r errors and performing the measurement perfectly. The protocol for fault-tolerantly measuring the eigenvalue of $\overline{P}$ is described as follows:

1. Perform flag t-FTEC.
2. Use a t-flag circuit to measure the eigenvalue of $\overline{P}$.
3. Repeat steps 1 and 2 $2t + 1$ times and take the majority of the eigenvalue of $\overline{P}$.

Step 1 is used to remove input errors to the measurement procedure. However during error correction, a fault can occur which could cause a new error on the data. Thus by repeating the measurement without performing error correction, the wrong state would be measured each time if there were no more faults. But repeating the syndrome $2t+1$ times, it is guaranteed that at least $t+1$ of the syndrome measurements had no faults and that the correct eigenvalue of $\overline{P}$ was measured. Thus taking the majority of the measured eigenvalues will give the correct answer.

Note that during the fault-tolerant measurement procedure, if there is a flag either during the error correction round or during the measurement of $\overline{P}$, when error correction is performed one corrects based on the possible set of errors resulting from the flag.

B.3 Candidate general w-flag circuit construction

In this section we provide a candidate general w-flag circuit construction for measuring the stabilizer $Z^{\otimes w}$. Although we do not provide a rigorous proof that our construction results in a w-flag circuit, we give several arguments as evidence that it satisfies all the criteria of a w-flag circuit. An illustration of the circuit construction (for $w = 12$) is given in Fig. B.1 and the description for how the circuit is constructed for arbitrary w is provided in the caption.

In what follows, we can restrict our attention to the case in which all v faults occur on CNOT gates in the circuit. The effect on the measurement outcomes and data qubits due to a set of v faults that include faults at idle and measurement locations can always occur due to at most v faults at CNOT locations only (as every qubit is involved in at least one CNOT). Moreover, we can assume that for CNOT_{fm} gates, the faults belong to the set $\{IZ, ZI, ZZ\}$ since X errors would never propagate to the data or affect the measurement outcome of a flag qubit. For CNOT_{dm} gates, we can assume that faults belong to the set $\{XZ, XI\}$. We only consider Z errors on the target qubit of a CNOT_{dm} for the same reason that was given for CNOT_{fm} gates. For the control qubit, an X error guarantees that the weight of the data qubit error increases even after the application of a satbilizer (since we are measuring $Z^{\otimes w}$).

We will use the following useful terminology: we say that a single-qubit Pauli at a time step in the circuit propagates to a qubit at a particular time-step if it would do so in the fault-free circuit. Given a single-qubit Pauli at a time step in the circuit, we say that another qubit is affected by the Pauli if it propagates to that qubit in any time step.

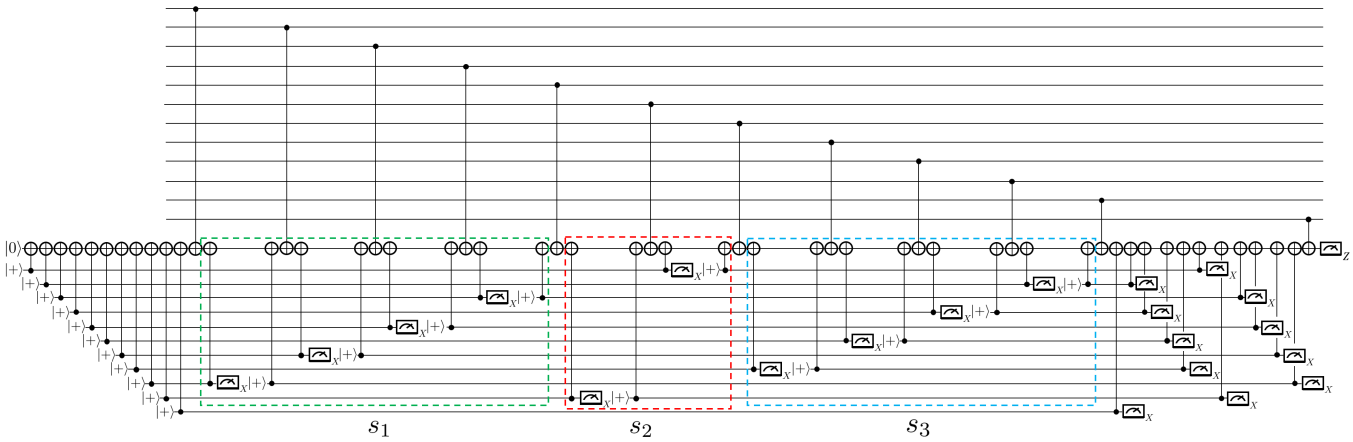


Figure B.1: Illustration of the general w-flag circuit construction for $w = 12$. In general, the circuit requires $w - 1$ flag qubits and is implemented using $7w - 8$ time steps. The circuit consists of two families of CNOT_{fm} gates. For the first family, with the first set of CNOT_{fm} gates located before the first CNOT_{dm} gate, the partnering CNOT_{fm} gates are divided into three sets s_1, s_2 and s_3 which are enclosed in the green, red and blue dashed boxes. In general, s_1 and s_3 both contain $(w - 4)/2$ CNOT_{fm} gates. In s_1, the j'th control qubit is at position $w + 2(j + 1)$ and in s_3 it is at position $w + 2j + 1$ with $j \in \{1, 2, \cdots, (w - 4)/2\}$ In s_2, the control qubits are always located at the $w + 2$'th and $2w - 1$'th qubits. Lastly, note that qubits are reused for implementing the second family of CNOT_{fm} gates. The partnering CNOT_{fm} gates are located in between the $w - 1$ and w'th CNOT_{dm} gates following an identical pattern as in s_1, s_2 and s_3 (in s_1 and s_3 the CNOT's are implemented in reverse order).

We now provide arguments for why the circuit is a w-flag circuit. First, note that every CNOT_{fm} gate comes as part of a pair with the measurement qubit being the target qubit. This ensures that when the circuit is fault-free, it implements a projective measurement of $Z^{\otimes w}$ without flagging. Next, notice that apart from the last two CNOT_{dm} gates, each CNOT_{dm} gate is followed by two CNOT_{fm} gates, one with its partnering CNOT_{fm} located before the first CNOT_{dm} and the other partner is located in between the last two CNOT_{dm} gates. Thus if there is a single Z error on the measurement qubit which propagates to any of the data qubits, the circuit will flag.

In all circuits considered in this section, s_0 will correspond to the sequence of CNOT_{fm} gates that come before the first CNOT_{dm} gate. First consider the shorter circuit construction using only the first family of CNOT_{fm} gates from the construction in Fig. B.1 (see the example in Fig. B.2). We can separate the set of all locations into subsets including two CNOT_{fm} gates and one CNOT_{dm} gate as shown in Fig. B.3 (apart from the last CNOT_{dm}). This circuit segment can increase the weight of the data error by at most one. There are four cases with inputs on the measurement qubit before the first CNOT_{fm} and CNOT_{dm} being $\{(I,I),(I,Z),(Z,I),(Z,Z)\}$. Note that if the following property held for each segment, then the circuit would be w-flag: for all inputs to the segment, if the weight of the data error increases and there are no faults in the segment, the segment flags. Unfortunately, for the input (Z,Z), this is not the case. Both input Z must come from at least two faults.

Note that if v faults results in a data qubit error of weight greater than v without causing the circuit in Fig. B.2 to flag, there must be either an IZ fault followed by no fault in a consecutive pair of CNOT_{fm} gates belonging to s_0 or a ZZ fault followed by two CNOT_{fm} gates that don't fail in s_0.

Moreover, a poor choice of ordering of the CNOT_{fm} gates in s_1, s_2 and s_3 can result in four faults causing a weight $\frac{w}{2}+1$ error on the data without causing the circuit to flag. Therefore, the ordering of the CNOT_{fm} gates in the sets s_1, s_2 and s_3 is chosen such that most Z errors in s_0 that first propagate to flag qubits connected to gates in s_1, will then propagate to flag qubits in s_3 and vice-versa. Typically, if a Z error propagates through multiple CNOT_{dm} gates in s_1, then unless CNOT_{fm} gates in s_3 fail, the flag qubits affected by the Z error would flag. Furthermore, the total number of required failures for gates in s_3 to cancel the Z errors will typically be equal to the number of times the Z error propagated to the data.

There are however cases which don't flag in which v faults in the circuit construction presented in Fig. B.2 lead to more than v errors on the data qubit, such as the example given in the figure. All such problematic cases that we found had a Z error on the target

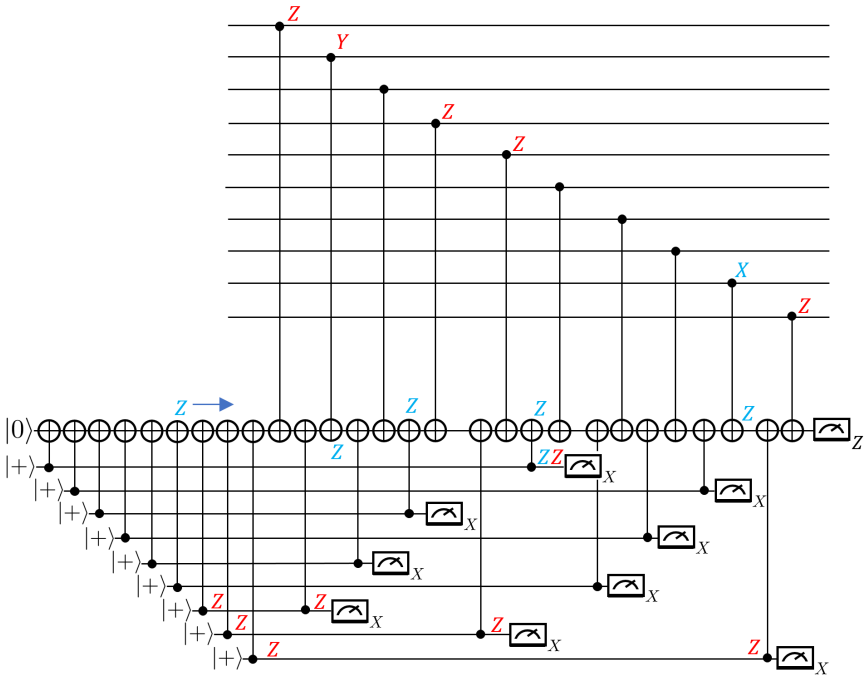


Figure B.2: Example of five faults that lead to an error of weight six on the data without causing a flag when only the first family of CNOT_{fm} gates are used in the construction of Fig. B.1 (here $w = 10$). Errors arising from faults are shown in blue and the resulting errors after propagating through the CNOT gates are shown in red.

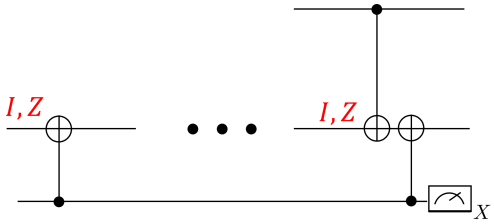


Figure B.3: Illustration of a pair of CNOT_{fm} gates as well as a segment of CNOT_{dm} followed by CNOT_{fm} gate. The first CNOT_{fm} gate belongs to the sequence of CNOT_{fm} gates that come before the first CNOT_{dm} gate (see Fig. B.1).

qubit in one of the last few CNOT_{fm} gates in s_0, followed by a Z error on the target qubit in one of the first few CNOT_{dm} gates in s_1. Then further Z errors occur throughout the remainder of the circuit which propagate to the data while preventing the flag qubits affected by the previous errors from flagging. Further, a Z error on the control qubit of the second CNOT_{fm} in s_2 cancels the Z which propagates to the flag qubit coupled to that CNOT_{fm} gate.

This particular problematic fault pattern would lead to flags if it occurred within the full circuit construction of Fig. B.1 (if the additional locations of the larger circuit do not fail). As this was the only type of problematic fault pattern that we found, one would hope that all problematic fault patterns are rendered non problematic provided no additional locations fail. Since the additional CNOT_{fm} gates always occur immediately after one of the original CNOT_{fm} gates (or after the last CNOT_{fm} gate), as far as the flag properties of the original circuit are concerned, no new problematic fault patterns are introduced.

We conclude this section by noting that our candidate general w-flag circuit construction requires $w-1$ flag qubits and is implemented in $7w-8$ time steps. This is clearly not optimal in general since for example, as shown in Fig. 3.9a, a w-flag circuit was found (for $w=6$) which requires only three flag qubits instead of five and the circuit is implemented in 14 time steps instead of 34. It is thus still an open problem to find optimal w-flag circuits for arbitrary w.

B.4 Quantum Reed-Muller codes

In this section we first describe how to construct the family of quantum Reed-Muller codes $\text{QRM}(m)$ with code parameters $[\![2^m-1, k=1, d=3]\!]$ following Ref. [63]. We then show that the family of $\text{QRM}(m)$ codes satisfy the sufficient flag 1-FTEC condition of Sec. 3.4.2.

Reed-Muller codes of order m (RM$(1, m)$) are defined recursively from the following generator matrices: First, RM$(1, 1)$ has generator matrix

$$G_1 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \tag{B.1}$$

and RM$(1, m + 1)$ has generator matrix

$$G_{m+1} = \begin{pmatrix} G_m & G_m \\ 0 & 1 \end{pmatrix}, \tag{B.2}$$

where 0 and 1 are vectors of zeros and ones in Eq. B.2. The dual of RM$(1, m + 1)$ is given by the higher order Reed-Muller code RM$(m - 2, m)$. In general, the generator matrices for higher-order Reed-Muller codes RM(r, m) are given by

$$H_{r,m+1} = \begin{pmatrix} H_{r,m} & H_{r,m} \\ 0 & H_{r-1,m} \end{pmatrix}. \tag{B.3}$$

with

$$H_{2,1} = H_{1,1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \tag{B.4}$$

The X stabilizer generators of QRM(m) are derived from shortened Reed-Muller codes where the first row and column of G_m are deleted. We define the resulting generator matrix as $\overline{G}_m$. The Z stabilizer generators are obtained by deleting the first row and column of $H_{m-2,m}$. Similarly, we define the resulting generator matrix as $\overline{H}_{m-2,m}$.

As was shown in [63], rows$(\overline{G}_m) \subset$ rows$(\overline{H}_{m-2,m})$ and each row has weight 2^{m-1}. Therefore, all the X-type stabilizer generators of QRM(m) have corresponding Z-type stabilizers. By construction, the remaining rows of $\overline{H}_{m-2,m}$ will have weight 2^{m-2}. Furthermore, every weight 2^{m-2} row has support contained within some weight 2^{m-1} row of the generator matrix $\overline{H}_{m-2,m}$. Therefore, every Z-type stabilizer generator has support within the support of an X generator.

FTEC scheme	Noise model	Number of qubits	Time steps (T_{time})	Pseudo-threshold
Full Steane-EC	$\tilde{p} = p$	≥ 171	15	$p_{\text{pseudo}} = 3.50 \times 10^{-3}$
Full Steane-EC	$\tilde{p} = p/100$	≥ 171	15	$p_{\text{pseudo}} = 1.05 \times 10^{-3}$
Flag-EC $[\![19, 1, 5]\!]$	$\tilde{p} = p$	22	$504 \leq T_{\text{time}} \leq 960$	$p_{\text{pseudo}} = 1.14 \times 10^{-5}$
Flag-EC $[\![19, 1, 5]\!]$	$\tilde{p} = p/100$	22	$504 \leq T_{\text{time}} \leq 960$	$p_{\text{pseudo}} = 7.74 \times 10^{-5}$

Table B.1: Pseudo-threshold results for the Full Steane and flag 2-FTEC protocol applied to the $[\![19, 1, 5]\!]$ code. Since the Steane error correction protocol is non-deterministic, the number of qubits will depend on how many times the encoded states are rejected. For low error rates, the states are accepted with high probability so that the average number of qubits is ≈ 171. Our three qubit flag error correction protocol requires at most six rounds of syndrome measurements, with each round using flag circuits requiring 168 time steps and the round using non-flag circuits requiring 120 time steps. However, for low noise rates, the average number of time steps will be close to 504 (since at least three rounds are required for the protocol to be fault-tolerant).

B.5 Implementation of Steane error correction

In this section we describe how to implement Steane error correction and discuss its fault-tolerant properties. We also provide a comparison of a version of Steane error correction with flag 2-FTEC protocol described in Sec. 3.2.2 applied to the $[\![19, 1, 5]\!]$ code.

Steane error correction is a fault-tolerant scheme that applies to the Calderbank-Shor-Steane (CSS) family of stabilizer codes [43]. In Steane error correction, the idea is to use encoded $|\overline{0}\rangle$ and $|\overline{+}\rangle = (|\overline{0}\rangle + |\overline{1}\rangle)/\sqrt{2}$ ancilla states to perform the syndrome extraction. The ancilla's are encoded in the same error correcting code that is used to protect the data. The X stabilizer generators are measured by preparing the encoded $|\overline{0}\rangle$ state and performing transversal CNOT gates between the ancilla and the data, with the ancilla acting as the control qubits and the data acting as the target qubits. After applying the transversal CNOT gates, the syndrome is obtained by measuring $|\overline{0}\rangle$ transversally in the X-basis. The code construction for CSS codes is what guarantees that the correct syndrome is obtained after applying a transversal measurement (see [11] for more details).

Similarly, the Z-stabilizer generators are measured by preparing the encoded $|\overline{+}\rangle$, applying CNOT gates transversally between the ancilla and the data with the data acting as the control qubits and the ancilla's acting as the target qubits. The syndrome is then obtained by measuring $|\overline{+}\rangle$ transversally in the Z-basis.

The above protocol as stated is not sufficient in order to be fault-tolerant. The reason

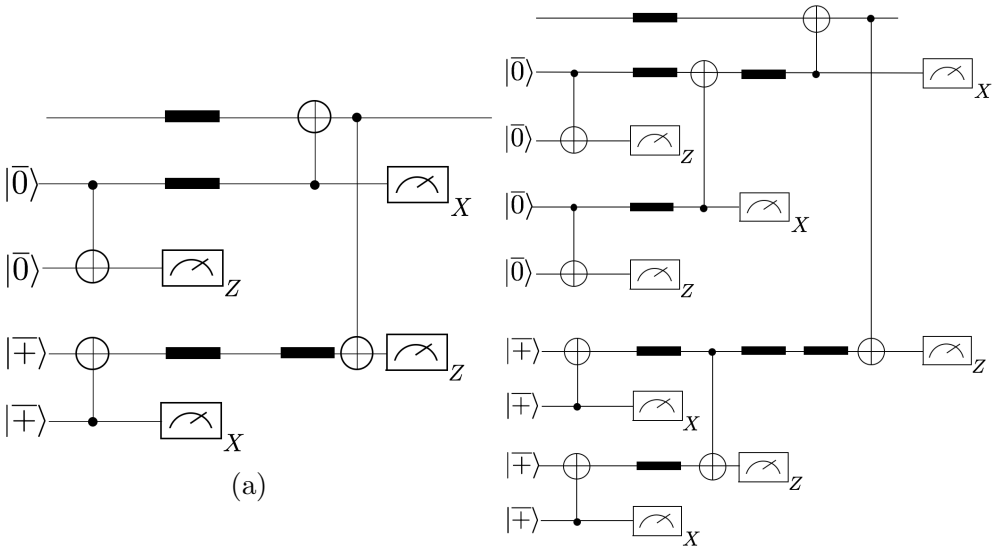
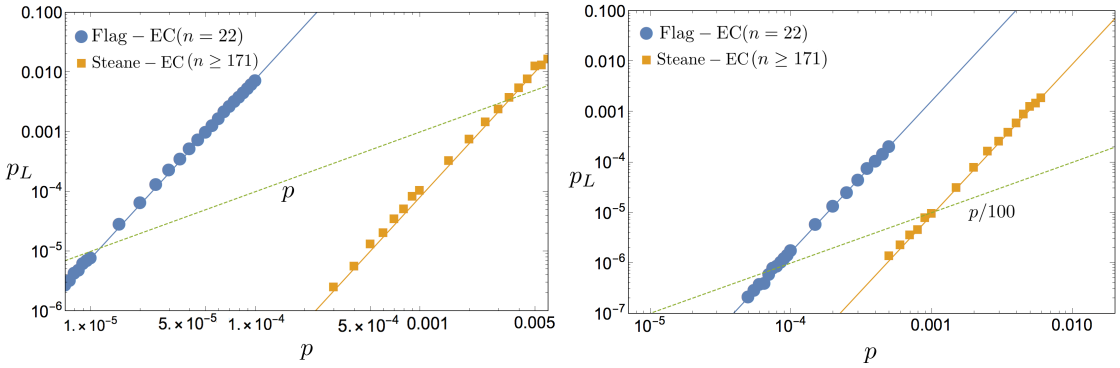


Figure B.4: (a) Fault-tolerant Steane error correction circuit for distance-three CSS codes. Each line represents an encoded qubit. The circuit uses only two encoded $|\overline{0}\rangle$ and $|\overline{+}\rangle$ ancilla states (encoded in the same error correcting code which protects the data) to ensure that faults in the preparation circuits of the ancilla's don't spread to the data block. (b) Fault-tolerant Steane error correction circuit which can be used for any distance-three CSS stabilizer code encoding the data. There are a total of eight encoded ancilla qubits instead of four. The dark bold lines represent resting qubits. Note that the circuit in Fig. B.4b could in some cases be used for higher distance CSS codes with appropriately chosen circuits for $|\overline{0}\rangle$ and $|\overline{+}\rangle$ ancilla states (see Ref. [6]).



(a) (b)

Figure B.5: Logical failure rate of the full fault-tolerant Steane error correction approach of Fig. B.4b and the flag 2-FTEC protocol of Sec. 3.2.2 applied to the $[\![19, 1, 5]\!]$ code. In (a) idle qubits are chosen to fail with a total probability $\tilde{p} = p$ while in (b) idle qubits fail with probability $\tilde{p} = p/100$. The intersection between the dashed curve and solid lines represent the pseudo-threshold of both error correction schemes.

is that in general the circuits for preparing the encoded $|\overline{0}\rangle$ and $|\overline{+}\rangle$ are not fault-tolerant in the sense that a single error can spread to a multi-weight error which could then spread to the data block when applying the transversal CNOT gates. To make the protocol fault-tolerant, extra $|\overline{0}\rangle$ and $|\overline{+}\rangle$ ancilla states (which we call verifier qubits) are needed to check for multi-weight errors at the output of the ancilla states.

For the $|\overline{0}\rangle$ ancilla, multiple X errors can spread to the data if left unchecked. Therefore, another encoded $|\overline{0}\rangle$ ancilla is prepared and a transversal CNOT gate is applied between the two states with the ancilla acting as the control and the verifier state acting as target. Anytime X errors are detected the state is rejected and the error correction protocol starts over. Further, if the verifier qubit measures a -1 eigenvalue of the logical Z operator, the ancilla qubit is also rejected. A similar technique is used for verifying the $|\overline{+}\rangle$ state (see Fig. B.4a).

For the $[\![7, 1, 3]\!]$ Steane code, an error $E = Z_i Z_j$ can always be written as $E = \overline{Z} Z_k$ where $\overline{Z}$ is the logical Z operator (this is not true for general CSS codes). But $|\overline{0}\rangle$ is a $+1$ eigenstate of $\overline{Z}$. Therefore, we don't need to worry about Z errors of weight greater than one occurring during the preparation of the $|\overline{0}\rangle$ state.

In Ref. [18] it was shown that unlike for the $[\![7, 1, 3]\!]$ code, for general CSS codes, the encoded ancilla states need to be verified for both X and Z errors in order for Steane error correction to satisfy the fault-tolerant properties of Def. 5. We show the general distance-

three fault-tolerant scheme in Fig. B.4b. Note that the circuit in Fig. B.4a will only satisfy the fault-tolerant criteria of Def. 5 for perfect distance-three CSS codes (see [18] for more details).

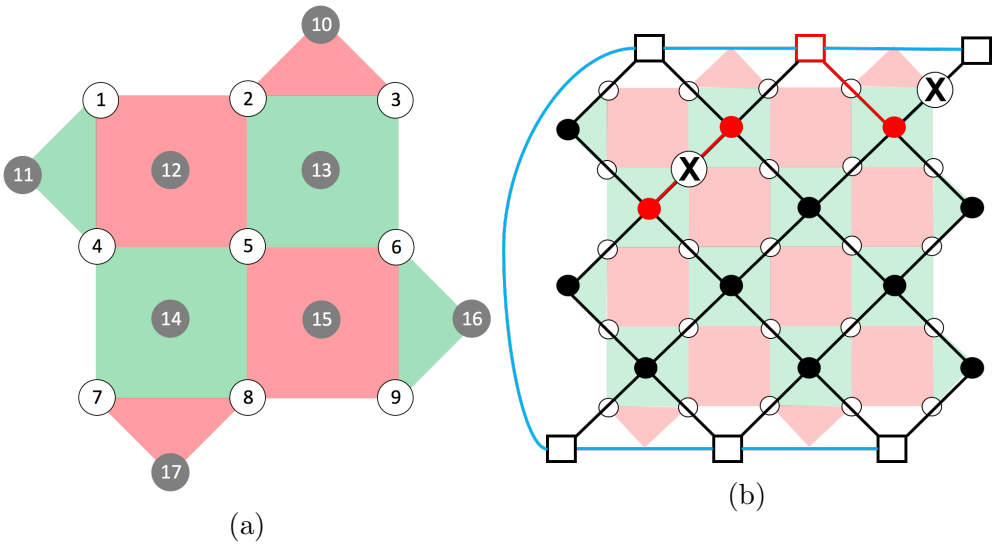
In Sec. 3.6.2 we computed logical failure rates for Steane error correction applied to the $[\![19, 1, 5]\!]$ code using the circuit of figure Fig. B.4a in order to minimize the number of physical qubits. However, since the $[\![19, 1, 5]\!]$ code is not a perfect CSS code, only the circuit in Fig. B.4b satisfies all the criteria of Def. 5. This explains why the leading order contributions to the logical failure was of the form $p_{\mathrm{L}} = c_1 p^2 + c_2 p^3 + \mathcal{O}(p^4)$ instead of $p_{\mathrm{L}} = cp^3 + \mathcal{O}(p^4)$ (which would be the case for a distance-5 code).

In Fig. B.5 we applied Steane error correction using the circuit of Fig. B.4b to achieve the full error correcting capabilities of the $[\![19, 1, 5]\!]$ code. We used methods presented in [6, 36] in order to obtain the encoded $|\overline{0}\rangle$ state (since the $[\![19, 1, 5]\!]$ code is self-dual, the $|\overline{+}\rangle$ state is obtain by interchanging all physical $|0\rangle$ and $|+\rangle$ states and reversing the direction of the CNOT gates). Note that not all $|\overline{0}\rangle$ and $|\overline{+}\rangle$ circuits had the same sequence of CNOT gates. This was to ensure that a single fault in two different preparation circuits, i.e. for $|\overline{0}\rangle$ and for $|\overline{+}\rangle$, would not lead to uncorrectable X or Z errors that would go undetected by the verifier ancillas and at the same time propagate to the data block. The results are compared with the flag 2-FTEC protocol of Sec. 3.2.2 applied to the $[\![19, 1, 5]\!]$ for the noise models where idle qubits fail with probability $\tilde{p} = p$ and $\tilde{p} = p/100$. In both cases the logical failure rates have a leading order p^3 contribution (which is determined from finding the best fit curve to the data). The pseudo-threshold results are given in Tab. B.1.

As can be seen, the full Steane-EC protocol using the circuit of Fig. B.4b achieves significantly lower logical failure rates compared to Steane-EC using the circuit in Fig. B.4a at the cost of using a minimum of 171 qubits compared to a minimum of 95 qubits. In contrast, the flag 2-FTEC scheme of Sec. 3.2.2 has a pseudo-threshold that is one to two orders of magnitude lower than than the full Steane-EC scheme but requires only 22 qubits.

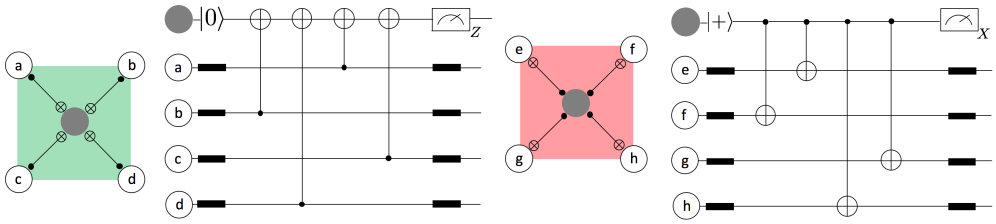
B.6 Implementation of Surface code error correction

We consider the rotated surface code [61, 4, 45, 46, 7, 62] as shown in Fig. B.6a, which has $n = d^2$ data qubits for distance d. Although we are concerned with error correction under the circuit level noise model described in Sec. 3.1.1, it is useful to build intuition by first considering the idealized noise model in which stabilizer measurements are perfect, and single qubit X errors occur with probability $2p/3$ (Z errors can be treated in the same way). An X type error E occurs with probability $\mathcal{O}(p^{\mathrm{wt}(E)})$, and has syndrome $s(E)$.



(a) (b)

Figure B.6: (a) The $d = 3$ surface code, with data qubits represented by white circles. The X (Z) stabilizer generators are measured with measurement ancillas (gray) in red (green) faces (b) For perfectg measurements, the graph G_{2D} used to correct X type errors (here for $d = 5$) consists of a black node for each Z-stabilizer, and a black edge for each data qubit in the surface code. White boundary nodes and blue boundary edges are added. Black and blue edges are given weight one and zero respectively. In this example, a two qubit X error has occurred causing three stabilizers to be violated (red nodes). A boundary node is also highlighted and a minimum weight correction (red edges) which terminates on highlighted nodes is found. The algorithm succeeds as the error plus correction is a stabilizer.



(a) (b)

Figure B.7: Circuits for measuring (a) Z-type, and (b) X-type generators. Identity gates (black rectangles) are inserted in the Z-type stabilizer measurement circuits to ensure that all measurements are synchronized. Note that unlike in [7], to be consistent with the other schemes in this chapter, we assume that we can prepare and measure in both the X and Z basis.

The minimum weight X-type correction can be found efficiently for the surface code in terms of the graph $G_{2\mathrm{D}}$ shown in Fig. B.6b. The graph $G_{2\mathrm{D}}$ has a bulk node (black circle) for each Z stabilizer generator, and a bulk edge (black) for each data qubit. A bulk edge coming from a bulk node corresponds to the edge's data qubit being in the support of the node's stabilizer. The graph also contains boundary nodes (white boxes) and boundary edges (blue), which do not correspond to stabilizers or data qubits. Each bulk and boundary edge is assigned weight one and zero respectively. The minimum weight decoder is then implemented as follows. After the error E is applied, the nodes corresponding to unsatisfied stabilizers are highlighted. If an odd number of stabilizers was unsatisfied, one of the boundary nodes is also highlighted. Highlighted nodes are then efficiently paired together by the minimum weight connections in the graph, by Edmonds' algorithm [79, 151]. The correction C is applied to the edges in the connection. Note that any single $\mathcal{O}(p)$ fault in this noise model corresponds to a weight one edge on the graph.

For circuit noise, we introduce a measurement qubit for each stabilizer generator, as represented by gray circles in Fig. B.6a, and circuits must be specified to implement the measurements, such as those in Fig. B.7. The performance of the code is sensitive to the choice of circuit [4], for example a poor choice could allow a single fault to cause a logical failure for $d = 3$ for any choice of decoder.

To implement the decoder, first construct a new three dimensional graph $G_{3\mathrm{D}}$ by stacking d copies of the planar graph $G_{2\mathrm{D}}$ that was shown in Fig. B.6b, and adding new bulk (boudnary) edges to connect bulk (boudnary) nodes in neighboring layers. We also add additional diagonal edges such that any single $\mathcal{O}(p)$ fault in the measurement circuits corresponds to a weight-one edge in $G_{3\mathrm{D}}$ (see Fig. B.8). For simplicity, we do not involve

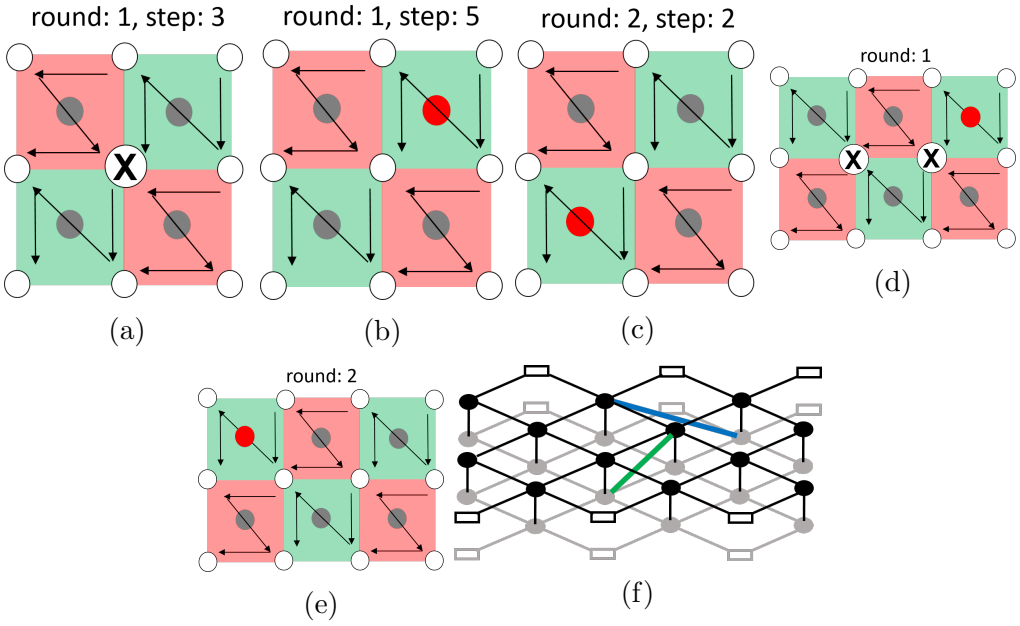


Figure B.8: Examples of a single fault leading to diagonal edges in G_{3D}. Dark arrows represent the CNOT sequence. (a) An X error occurs during the third time step in the CNOT gate acting on the central data qubit. (b) During the fifth time step of this round, the X error is detected by the Z type measurement qubit to the top right. (c) The X error is not detected by the bottom left Z type stabilizer until the following round. (d) An XX error occurs on the third CNOT of an X measurement circuit, which is detected by the Z measurement to the right. (e) Detection by the left Z stabilizer does not occur until the next round. (f) The corresponding edges in G_{3D}, green for (a-c), and blue for (d-e). Here we show two rounds of the graph ignoring boundary edges.

further possible optimizations such as setting edge weights based on precise probabilities and including X-Z correlations [50].

All simulations of the surface code are performed using the circuit noise model in Sec. 3.1.1, with the graph G_{3D} described above as follows (to correct X errors):

1. Data acquisition: Stabilizer outcomes are stored over d rounds of noisy error correction, followed by one round of perfect error correction. The net error E applied over all d rounds is recorded.

2. Highlight nodes: Nodes in the graph G_{3D} are highlighted if the corresponding Z-type

FTEC scheme	Noise model	Number of qubits	Time steps (T_{time})	Pseudo-threshold
Flag-EC $[\![7,1,3]\!]$	$\tilde{p}=p$	9	$36 \leq T_{\text{time}} \leq 108$	$p_{\text{pseudo}} = 3.39 \times 10^{-5}$
Flag-EC $[\![7,1,3]\!]$		11	$34 \leq T_{\text{time}} \leq 104$	$p_{\text{pseudo}} = 2.97 \times 10^{-5}$

Table B.2: Pseudo-thresholds and circuit depth for flag-EC protocols using two and four ancilla qubits applied to the $[\![7,1,3]\!]$ code. The results are presented for the noise models where $\tilde{p}=p$ and $\tilde{p}=p/100$.

stabilizer outcome changes in two consecutive rounds. [1]

3. Minimum weight matching: Find a minimal edge set forming paths that terminate on highlighted nodes. Highlight the edge set.

4. Vertical collapse: The highlighted edges in $G_{3\text{D}}$ are mapped edges in the planar graph $G_{2\text{D}}$, and are then added modulo 2.

5. Correction: The X-type correction C_X is applied to highlighted edges in $G_{2\text{D}}$.

The Z correction C_Z is found analogously. Finally, if the residual Pauli $R = EC_XC_Z$ is a logical operator, we say the protocol succeeded, otherwise we say it failed.

[1] For an odd number of highlighted vertices, highlight the boundary vertex.

B.7 Compact implementation of flag error correction

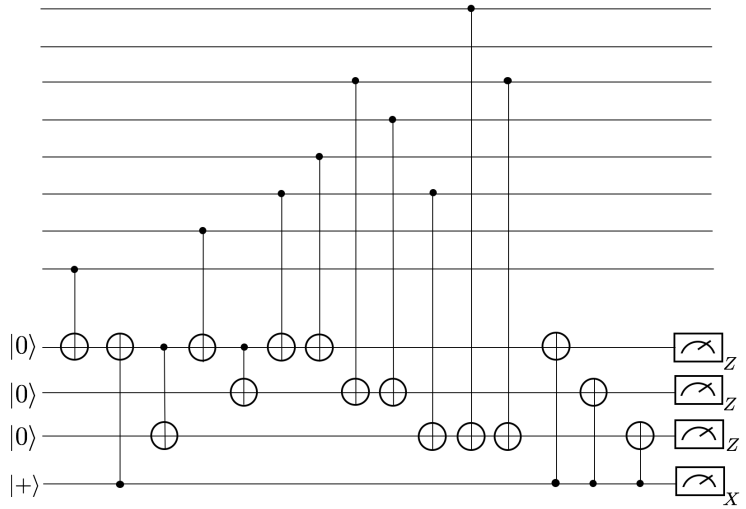


Figure B.9: Circuit for measuring the Z stabilizer generators of the $[\![7,1,3]\!]$ code using one flag qubit and three measurement qubits. The circuit is constructed such that any single fault at a bad location leading to an error of weight greater than one will cause the circuit to flag. Moreover, any error that occurs when the circuit flags due to a single fault has a unique syndrome.

In [57], it was shown that by using extra ancilla qubits in the flag-EC protocol, it is possible to measure multiple stabilizer generators during one measurement cycle which could reduce the circuit depth. Note that for the Steane code, measuring the Z stabilizers using Fig. 3.1b requires only one extra time step. In this section we compare logical failure rates of the $[\![7,1,3]\!]$ code using the flag-EC method of Sec. 3.2.1 which requires only two ancilla qubits and a flag-EC method which uses four ancilla qubits but that can measure all Z stabilizer generators in one cycle (see Fig. B.9). All X stabilizers are measured in a separate cycle.

Logical failure rates for $\tilde{p} = p$ are shown in Fig. B.10. Pseudo-thresholds and the number of time steps required to implement the protocols are given in Tab. B.2. Note that measuring stabilizers using two ancilla's requires at most two extra time steps. Furthermore, the extra ancilla's for measuring multiple stabilizers result in more idle qubit locations compared to using only two ancilla qubits. With the added locations for errors to be introduced, the flag error correction protocol using only two ancilla's achieves a *higher*

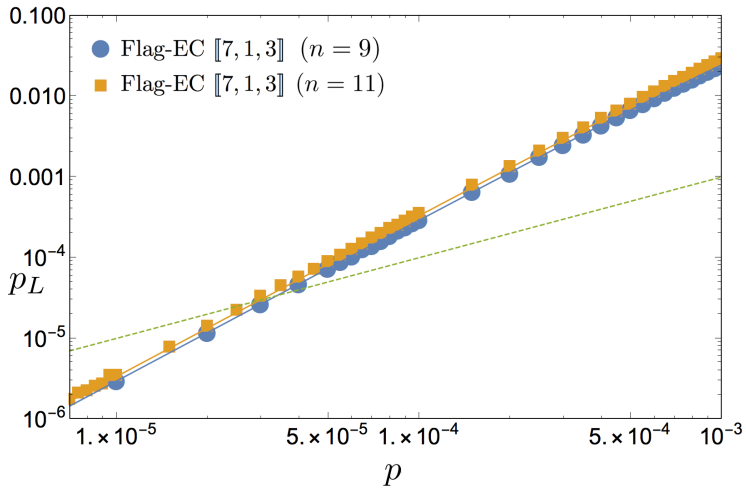


Figure B.10: Logical failure rates of the flag 1-FTEC protocols using two and four ancilla qubits applied to the $[\![7,1,3]\!]$ Steane code.

pseudo-threshold compared to the protocol using more ancilla's. Thus assuming that reinitializing qubits can be done without introducing many errors into the system, FTEC using fewer qubits could achieve lower logical failure rates compared to certain schemes using more qubits.

B.8 Stabilizer generators for the distance-five color codes

In Tab. B.3 we provide stabilizer generators for the $[\![19,1,5]\!]$ and $[\![17,1,5]\!]$ color codes.

$[\![19,1,5]\!]$ code	$[\![17,1,5]\!]$ code
$Z_1Z_2Z_3Z_4,\ \ X_1X_2X_3X_4$	$Z_1Z_2Z_3Z_4,\ \ X_1X_2X_3X_4$
$Z_1Z_3Z_5Z_7,\ \ X_1X_3X_5X_7$	$Z_1Z_3Z_5Z_6,\ \ X_1X_3X_5X_6$
$Z_{12}Z_{13}Z_{14}Z_{15},\ \ X_{12}X_{13}X_{14}X_{15}$	$Z_5Z_6Z_9Z_{10},\ \ X_5X_6Z_9Z_{10}$
$Z_1Z_2Z_5Z_6Z_8Z_9,\ \ X_1X_2X_5X_6X_8X_9$	$Z_7Z_8Z_{11}Z_{12},\ \ X_7X_8X_{11}X_{12}$
$Z_6Z_9Z_{16}Z_{19},\ \ X_6X_9X_{16}X_{19}$	$Z_9Z_{10}Z_{13}Z_{14},\ \ X_9X_{10}X_{13}X_{14}$
$Z_{16}Z_{17}Z_{18}Z_{19},\ \ X_{16}X_{17}X_{18}X_{19}$	$Z_{11}Z_{12}Z_{15}Z_{16},\ \ X_{11}X_{12}X_{15}X_{16}$
$Z_{10}Z_{11}Z_{12}Z_{15},\ \ X_{10}X_{11}X_{12}X_{15}$	$Z_8Z_{12}Z_{16}Z_{17},\ \ X_8X_{12}X_{16}X_{17}$
$Z_8Z_9Z_{10}Z_{11}Z_{16}Z_{17}$	$Z_3Z_4Z_6Z_7Z_{10}Z_{11}Z_{14}Z_{15}$
$Z_5Z_7Z_8Z_{11}Z_{12}Z_{13}$	$X_3X_4X_6X_7X_{10}X_{11}X_{14}X_{15}$
$X_5X_7X_8X_{11}X_{12}X_{13}$	
$X_8X_9X_{10}X_{11}X_{16}X_{17}$	
$\overline{X} = X^{\otimes 19}, \overline{Z} = Z^{\otimes 19}$	$\overline{X} = X^{\otimes 17}, \overline{Z} = Z^{\otimes 17}$

Table B.3: Stabilizer generators for the $d = 5$ ($[\![19, 1, 5]\!]$ code) and $d = 5$ ($[\![17, 1, 5]\!]$ code) family of color codes [5]. The last row illustrates representatives of the codes logical operators.

Appendix C

Error correction for input Clifford errors

In this appendix we will explain why a generic $n-$qubit Clifford error on an encoded state transforms into a logical Clifford operation and a physical Pauli operation, upon doing stabilizer measurements.

Before proceeding, we provide a few definitions. Let $\bar{\rho}$ be an encoded state of a stabilizer code with stabilizer $\mathcal{S}$, that undergoes an error described by some CPTP map $\mathcal{E}$, i.e, $\bar{\rho} \mapsto \mathcal{E}(\bar{\rho})$. Let Π_0 denote the projector onto the code space and consequently Π_s denote the projector onto the syndrome space of s. Let T_s be a Pauli operation that takes a state from the syndrome s subspace to the code space, i.e, $\Pi_s = T_s \cdot \Pi_0 \cdot T_s$. Upon measuring the stabilizer generators to obtain a syndrome s and applying the corresponding T_s operation, the noisy state can be projected back to the code space,

$$\mathcal{E}(\bar{\rho}) \mapsto T_s \Pi_s \; \mathcal{E}(\bar{\rho}) \; \Pi_s T_s \; . \tag{C.1}$$

Since the resulting state is in the codespace, it can be decoded back to a single qubit state ϕ, given by

$$\phi = \sum_a \mathrm{Tr} \left(T_s \Pi_s \; \mathcal{E}(\bar{\rho}) \; \Pi_s T_s \cdot \bar{P}_a \right) P_a \tag{C.2}$$

Let us denote the combined effect of the encoder, noise model, the projection to code space and the decoder by a quantum channel called the *effective projection*. The effective projection can be seen as acting directly on ρ to result in ϕ [152].

However, in order to extract the (single qubit or logical) CPTP map defining the effective projection channel, we must make use of another tool, an isomorphism between channels and states, called the Choi-Jamiłowski isomorphism [153, 154]. Under this isomorphism, a single qubit CPTP channel $\mathcal{E}$ is mapped to a unique bipartite quantum state $\mathcal{J}(\mathcal{E})$ called the *Choi matrix* corresponding to $\mathcal{E}$, in the following manner.

$$\mathcal{J}(\mathcal{E}) = \frac{1}{4}\sum_{i,j} \mathcal{E}(P_i) \otimes P_j^T \tag{C.3}$$

Furthermore, when the quantum channel is expressed as a process matrix Λ, where $\Lambda_{i,j} = \mathrm{Tr}\left(\mathcal{E}(P_i)\cdot P_j\right)$, where $P_i, P_j \in \{I, X, Y, Z\}$ are Pauli matrices, it follows that

$$\Lambda_{ij} = \mathrm{Tr}\left(\mathcal{J}(\mathcal{E})\cdot(P_j \otimes P_i^T)\right) \ . \tag{C.4}$$

Lastly, note that when $\mathcal{E}(\rho) = C\cdot\rho\cdot C^\dagger$ where C is a Clifford operation, the corresponding process matrix $\Lambda(\mathcal{E})$ is given by

$$\begin{aligned}\Lambda(\mathcal{E})_{ij} &= \mathrm{Tr}\left(C\cdot P_i\cdot C^\dagger\cdot P_j\right)\\ &= \mathrm{Tr}\left(P_{\sigma(i)}\cdot P_j\right) = \delta_{\sigma(i),j}\end{aligned} \tag{C.5}$$

where σ is a permutation of $\{1,2,3,4\}$ that depends on the Clifford operation. Hence the process matrix of a Clifford channel is a permutation matrix.

We now have all the necessary ingredients to prove our claim. Note that if ρ in Eq. C.1 is the Bell state and $\mathcal{E}$ acts on one half of the encoded Bell state, then using Eq. C.3, we know that ϕ in Eq. C.2 is simply the Choi matrix corresponding to the effective projection channel. From Eq. C.4, it follows that the process matrix for the effective projection channel, denoted by $\Lambda^{(1)}$, is given by

$$\begin{aligned}\Lambda^{(1)}_{ij} &= \mathrm{Tr}\left(T_s\Pi_s\ \mathcal{E}(\Pi_0\cdot\bar{P}_i)\ \Pi_s T_s\Pi_0\bar{P}_j\right)\\ &= \mathrm{Tr}\left(\mathcal{E}(\Pi_0\cdot\bar{P}_i)\cdot\Pi_s\cdot\bar{P}_j\right)\\ &= \sum_{\substack{l\\ P_l\in\bar{P}_i\cdot\mathcal{S}}}\ \sum_{\substack{k\\ P_k\in\bar{P}_j\cdot\mathcal{S}}} c_k\Lambda_{lk}\\ &= \sum_{\substack{l\\ P_l\in\bar{P}_i\cdot\mathcal{S}}}\ \sum_{\substack{k\\ P_k\in\bar{P}_j\cdot\mathcal{S}}} c_k\delta_{l,\sigma(k)}\end{aligned} \tag{C.6}$$

where $c_k \in \{+1, -1\}$ and in the last step, we have used the fact that Λ is the process matrix of a Clifford operation, Eq. C.5. That $\Lambda^{(1)}$ is also a permutation matrix follows

from two properties – (i) Every row of $\Lambda^{(1)}$ has a unique non-zero column and (ii) every column of $\Lambda^{(1)}$ has a unique non-zero row. We will show (i) explicitly in what follows, the proof for (ii) is almost identical. Suppose that there are two columns j' and j'' such that $\Lambda_{ij'} \neq 0$ and $\Lambda_{ij''} \neq 0$. Along with Eq. C.6, it implies that there exists $P_{k'} \in \bar{P}_{j'} \cdot \mathcal{S}$ and $P_{k''} \in \bar{P}_{j''} \cdot \mathcal{S}$ such that $\delta_{l,\sigma(k')} = \delta_{l,\sigma(k'')}$. Hence, it must be that $\sigma(k') = \sigma(k'')$, in other words, $j' = j''$.

Hence the effective projection channel is indeed a Clifford operation, in other words, any $n-$qubit Clifford operation on the physical qubits can be promoted to a logical Clifford operation and a physical Pauli error (T_s in Eq. C.1), by a syndrome measurement.