Quantum Error Mitigation and Error Correction for Practical Quantum Computation

Armands Strikis

Mansfield College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2023

This thesis is dedicated to
Daina Kārkliņa
(1950-2020)

Acknowledgements

The last three years have been the most stimulating, adventurous and cheerful years of my life. None of that would be true without the immense support and friendship of the following people whom I cannot thank enough.

First and foremost, my biggest gratitude goes to my supervisor *Simon Benjamin*. It is hard to describe the magnitude of the positive impact Simon had on my life. He made the doctoral journey an utmost pleasure to the point of making my friends envious as I was telling stories about his supervision. But who could blame them, Simon is not only an excellent mentor but also a natural leader and a brilliant scientist. It is incredible that one person can fulfil all of these roles, and I am very thankful that I could learn from him.

For the other half of my academic success, I have to thank my collaborators. *Ying Li* for patiently guiding me through the murky waters of QEM. *Ben Brown* for hosting dozens of discussions on the most diverse aspects of QEC. And *Lucas Berent* for his endurance to have day-long discussions until our brains shut down.

I want to thank current and past members of the *QTechTheory* group. Special thanks to *Cica Gustiani*, *Tim Chan* and *Arthur Rattew* for endless office discussions and *Zhenyu Cai*, *Hamza Jnane* and *Adam Siegel* for all the fun we had in Ghent. I am glad the world didn't implode when the pickiest eater met the least one.

In addition to the excellent academic cohort, I was fortunate to have one of the finest group of friends. I thank *Joseph McGovern* and *Thomas Han* for keeping me sane during the covid year and after, my current and recent housemates *Alex Cliffe* and *Oliwier Melon* for being overall awesome friends and *David Rytz* and *Tom McAuliffe* for unexpected invites to random events. I can't leave this without acknowledging *MCBC* which kept me physically active while churning more drama than any telenovela. But for the most fun I had in a year, I have to say thanks to *Lily Mott*. The range of places, foods, adventures and emotions I experienced cannot be matched; and yet there are so many more turtles to see!

Finally, I want to thank my family and my dear Latvian friend *Lauris Stasjuns*. Thank you for having my back in all of my ventures.

Abstract

We are rapidly entering the era of potentially useful quantum computation. To keep on designing larger and more capable quantum computers, some form of algorithmic noise management will be necessary. In this thesis, I propose multiple practical advances in quantum error mitigation and error correction. First, I present a novel and intuitive way to mitigate errors using a strategy that assumes no or very minimal knowledge about the nature of errors. This strategy can deal with most complex noise profiles, including those that describe severe correlated errors. Second, I present proof that quantum computation is scalable on a defective planar array of qubits. This result is based on a two-dimensional surface code architecture for which I showed that a finite rate of fabrication defects is not a fundamental obstacle to maintaining a non-zero error-rate threshold. The same conclusions are supported by extensive numerical studies. Finally, I give a new perspective on how to view and construct quantum error-correcting codes tailored for modular architectures. Following a given recipe, one can design codes that are compatible with the qubit connectivity demanded by the architecture. In addition, I present several product code constructions, some of which correspond to the latest developments in quantum LDPC code design. These and other practical advancements in quantum error mitigation and error correction will be crucial in guiding the design of emerging quantum computers.

Contents

List of Publications

This thesis is based on the following publications.

1. A. Strikis, D. Qin, Y. Chen, S. C. Benjamin and Y. Li, "*Learning-based quantum error mitigation*", PRX Quantum 2, 040330 (2021).

2. A. Strikis, S. C. Benjamin, and B. J. Brown, "*Quantum computing is scalable on a planar array of qubits with fabrication defects*", Phys. Rev. Applied 19, 064081 (2023).

3. A. Siegel, A. Strikis, T. Flatters, S. Benjamin, "*Adaptive surface code for quantum error correction in the presence of temporary or permanent defects*", (2022), arXiv:2211.08468. (Accepted in Quantum)

4. A. Strikis, L. Berent, "*Quantum Low-Density Parity-Check Codes for modular architectures*", PRX Quantum 4, 020321 (2023).

In a few places of this thesis, I use text from the publications verbatim; such text was originally authored by me.

Chapter 1

Introduction

The development of quantum technologies may revolutionise the field of computation and deliver important applications for drug discovery, materials design, cryptography as well as to fundamental science. Realising this promise, however, is a difficult task. Most quantum applications require computing systems on the scale of thousands of qubits. Moreover, these qubits must be controlled with fidelity such that billions of quantum operations can be performed with a low overall error chance. For example, one of the newest versions of Shor's algorithm requires 6000 qubits with 3 billion reliable operations to factor a 2048-bit integer [1]. Since quantum systems are very susceptible to errors which destroy the quantum properties, the execution of such algorithms is incredibly challenging.

It is clear that the first stage solution is to devise better and more noise-resilient quantum platforms from the engineering point of view. This is a slow and expensive endeavour and novel engineering advancements are extremely hard to come by. A complementary approach, emerging from the theory community, is to create a defensive layer against the errors sitting above the hardware level. The community mainly differentiates between two such forms of protection – quantum error mitigation (QEM) and quantum error correction (QEC). The latter is believed to be the path to scalable fault-tolerance while the former could find its uses in small to medium size systems.

Today, leading tech companies such as Google, IBM and Microsoft and many university teams have joined the race to build the first useful quantum computer.

Even with limited devices, many teams have successfully demonstrated implementations of promising error mitigation and correction strategies, see for example [2, 3]. Among many platforms, it is not yet clear which approach is best but, ultimately, some form of error management will be necessary. Many experimental teams are in the early stages of planning their quantum computing architectures, hence, current advances in QEM or QEC can greatly influence the way in which the quantum computer is designed. Therefore, it is very timely to put more effort into designing novel and better-performing error mitigation and correction strategies. These developments will be crucial to speeding up the advent of scalable and practical quantum computation.

In this thesis, I present several advancements towards the practical uses of selected QEM and QEC methods. In Chapter 3, I discuss a novel and intuitive error mitigation protocol that extends the basic probabilistic error cancellation method. This protocol has a new learning component, which allows one to perform reliable error mitigation with little or no knowledge of the pre-existing error model. Moreover, it can deal with complex noise profiles such as spatially or temporally correlated errors.

In Chapter 4, I present proof that quantum computation is scalable on a planar array of qubits with a finite rate of fabrication defects. Assuming a surface code architecture, these defects effectively create punctures in the code, and therefore, limit the performance of the code. By suitably redesigning the stabilisers around these punctures, I show that the physical error rate threshold does not vanish in the asymptotic limit. The second part of this chapter describes joint-authored work presenting several numerical simulations that support the theoretical results.

Finally, in Chapter 5, I propose a new perspective on how to view and construct quantum LDPC codes for modular architectures. By viewing the intra- and inter-modular connectivity as two separate codes, one can construct a product code that fully respects the architectural constraints.

Each of these chapters has an individual introduction to the subject and contains a short literature review. The background that is shared between two or more

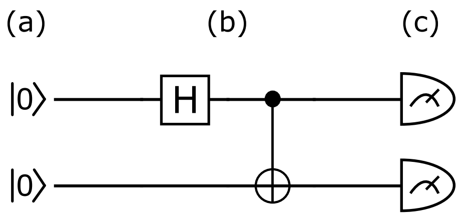


Figure 1.1: An example quantum circuit. Qubits are represented by lines and the circuit is read from left to right. (a) represents state initialisation, (b) unitary quantum gates and (c) single-qubit measurements. The exact meaning of circuit elements will be introduced later.

chapters is summarised in this and the next chapter. Specifically, I introduce parts of general and noisy quantum computation in the rest of this chapter and quantum error correction in Chapter 2. Both chapters introduce the notation that is assumed throughout the thesis.

1.1 Quantum computation

As I suggested above, quantum computation may turn out to be very impactful in multiple areas of society. In the rest of this chapter, I will briefly introduce aspects of quantum computation and later consider the effect of noise on it. Note that, in this thesis, I will only consider digital quantum computation and leave out the more specialised variant of analogue quantum computation. As is customary, I will represent it with a universal and intuitive framework of quantum circuits.

A quantum circuit is loosely defined as a sequence of discrete operations on *qubits*. See Fig. 1.1 for an example of a two-qubit circuit. These operations constitute qubit state initialisation, unitary quantum gates, i.e. operations that transform the state of a qubit from one to another, and measurements. A qubit is a two-dimensional mathematical object whose state is defined by a normalised vector

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \tag{1.1}$$

where α, β are complex numbers called amplitudes. The vector is usually expressed in the computational basis $|0\rangle = (1 \quad 0)^T$ and $|1\rangle = (0 \quad 1)^T$ as $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ and $\frac{\beta/|\beta|}{\alpha/|\alpha|}$ is called the relative phase, or phase for short.

The adjoint of $|\psi\rangle$ is denoted as $\langle\psi| = (\alpha^* \quad \beta^*)$ and serves as a linear functional that acts on any vector $|\phi\rangle$ as $\langle\psi|\phi\rangle \in \mathbb{C}$. Therefore, the vectors are part of a two-dimensional complex Hilbert space $\mathcal{H}_2$ - a vector space equipped with an inner product, where the correspondence is $(\cdot,\cdot) = \langle\cdot|\cdot\rangle$. A Hilbert space of multiple qubits is defined as a tensor product of single-qubit spaces $\mathcal{H} = \mathcal{H}_2 \otimes \mathcal{H}_2 \otimes \ldots$. Similarly, a multi-qubit state can be written as a vector $|\phi\rangle \in \mathcal{H}$, and we say it is in a product form if it can be written as $|\phi\rangle = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \ldots$ (or in a shorter notation $|\phi\rangle = |\psi_1\psi_2\ldots\rangle$), where $|\psi\rangle_i$ represent single qubit states. With these definitions, a quantum state initialisation corresponds to setting a qubit(s) in a state $|\psi\rangle$. This state often has a simple-to-create product form. Unitary gates correspond to unitary matrices $U_1U_2\ldots = U$ being applied to the initial state, that is $U|\psi\rangle$, and measurements are destructive actions in which one of the basis states of $U|\psi\rangle$ is measured with probability $|a|^2$, where a is the amplitude of the respective state. Note that measurements can be performed over different bases.

In general, the main subroutine of all quantum algorithms is the execution of one or more quantum circuits. Since it is difficult to simulate most such circuits classically, it is widely believed that there exist at least a few quantum algorithms that offer speedups over their classical counterparts. There is much more to say about quantum algorithms, see for example [4], however, the results and ideas presented in this thesis will be mostly agnostic of them.

As we will later see, when dealing with noise, a slightly different perspective of qubits and their operations is advantageous. This perspective is the framework of ensembles of quantum states [5]. An ensemble is a collection $\{p_i, |\psi_i\rangle\}$ which describes a mixture of quantum states $|\psi_i\rangle$. Here, p_i is the probability that the quantum system is in the state $|\psi_i\rangle$. Bear in mind this is a purely classical probability distribution and should not be confused with a superposition of quantum states. The ensemble can be written in a compact form as a Hermitian operator called the density operator

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|\,. \tag{1.2}$$

Any density operator satisfies two properties -

1. $\mathrm{Tr}(\rho) = 1$, ensuring that $\sum_i p_i = 1$, that is, the system is always found in one of the states $|\psi_i\rangle$,

2. ρ is positive semi-definite, $\rho \geq 0$, ensuring that probabilities p_i are nonnegative.

Due to the spectral theorem, the same conclusions may be drawn when relating eigenvalues of density operators to these requirements [6].

Using the framework of density operators, we can generalise quantum operations as follows. Any evolution of a mixture of quantum states given by a density operator ρ can be written as a CPTP (completely positive and trace-preserving) map

$$\mathcal{E} : \rho \to \mathcal{E}(\rho) = \rho' \tag{1.3}$$

which is called a *quantum channel*. These generalised channels include unitary gates, measurements and even operations like partial trace. Every CP map can be expressed in terms of Kraus operators $\{K_j\}$ as

$$\mathcal{E}(\rho) = \sum_j K_j \rho K_j^\dagger, \tag{1.4}$$

which satisfy $\sum_j K_j^\dagger K_j \leq 1$ with the equality corresponding to a CPTP map [5]. Quantum channel formalism is very useful for the discussion of noisy quantum computation as it captures all of the characteristics of the noise in a very compact way. This is slightly more difficult to achieve using the circuit framework alone. However, one can still think of these maps being applied to a quantum circuit in place of unitary gates. In that case, the initial quantum state is described with some initial density operator.

In the following subsection, I will use quantum channel formalism to describe noisy quantum computation and provide some examples. In turn, all noise models in this thesis will be defined as noise channels applied at some specified locations in the quantum circuit.

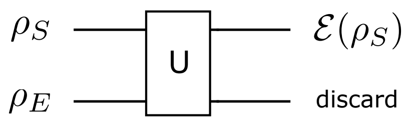


Figure 1.2: In this circuit, lines represent the system's state ρ_S and the environment's state ρ_E respectively. The joint state ρ_{ES} may undergo some unitary evolution U. After that, the description of the system's evolution is obtained by discarding the environmental degrees of freedom.

1.2 Noisy quantum computation

Interactions with the environment are a large part of the study of quantum theory and give profound insights as to why we see the everyday world in a classical setting rather than quantum. The effects of environmental interactions decohere a quantum state and reduce the system to a macroscopic classical state that we are used to observing in everyday situations [7]. Therefore, to employ quantum computers for potentially powerful large-scale quantum algorithms, we must manage noise in these systems. This will be the overarching topic of this thesis.

Noise in quantum systems can be induced by many physical processes and theoretical limits - thermal fluctuations of the system or environment, unwanted coupling between qubits, imperfect system control and even shot noise, to name a few, all lead to erroneous computation outcomes. The Kraus operator formalism introduced above allows us to characterise noise in both a closed and an open system. Certain kinds of imperfect control, e.g. from systematic over-rotation of a quantum gate, will lead to a unitary noise, while a non-trivial coupling between the system and environment generates a non-unitary noise.

To describe any such noise, it suffices to consider a system-environment joint state ρ_{ES} undergoing some unwanted unitary evolution U written as $U\rho_{ES}U^\dagger$. We can always expand the system-environment space such that it forms a closed system and, as per Schrödinger equation, any evolution defined on it is unitary. Then, the evolution of the system alone $\mathcal{E}(\rho_S)$ is obtained by tracing out the environment with a partial trace as

$$\mathcal{E}(\rho_S) = \mathrm{Tr}_E(U\rho_{ES}U^\dagger), \tag{1.5}$$

where the CPTP map $\mathcal{E}(\rho_S)$ defines the Kraus operators. See Fig. 1.2 for the process in a circuit format. Going further, I will only consider the state of interest ρ_S and, hence, denote it without the subscript S.

Often the goal in quantum computation is to either find the value of some observable or produce an output state at the end of a specified circuit. For example, the former could include finding the energy expectation value E of a Hamiltonian H that describes a chemical system [8]. Using the formalism of density operators, this is written as

$$E = \mathrm{Tr}(\rho_{\mathrm{out}} H) = \sum_i p_i \langle\psi_i| H |\psi_i\rangle , \tag{1.6}$$

where ρ_{out} is the ideal output state produced by the circuit. If the output state is not an eigenstate of H, then we might expect to receive a different result for each individual shot of the circuit. The average of these results is an unbiased estimate of the expectation value as long as all circuit operations are executed perfectly. The presence of noise could lead one to either produce the wrong output state ρ'_{out} or measure the wrong observable H'. In these cases, the estimator of the expectation value is highly likely to be biased. If the bias is too large, the computation might be rendered worthless and, hence, reducing noise and consequently the bias to some tolerated level is crucial. Similarly, if the task of quantum computation is to produce a classical string of bits, for example, by measuring all qubits in $|0\rangle / |1\rangle$ basis needed in Shor's factoring algorithm [9], then noisy operations might result in the incorrect outcomes of one or more qubit measurements.

To better visualise the impact of noise on quantum computation, let us consider some common noise channels. As mentioned before, quantum states decohere over time, which means that when we express the state ρ in the computational basis, its off-diagonal elements are suppressed. This is often modelled by a dephasing channel, which, in some sense, characterises the system's ability to remain quantum. For a single qubit, the dephasing channel is given as

$$\mathcal{D}_{\mathrm{Ph}}(\rho) = (1-\epsilon)\, I\rho I + \epsilon\, Z\rho Z, \tag{1.7}$$

where I and Z represent unitary identity and phase-flip operations respectively, and $\epsilon \in [0, 1/2]$ describes the severity of the channel. The severities of general error channels are associated with the error rates that are frequently quoted by experimental teams. The generalisation to a multi-qubit system is straightforward, and a two-qubit example can be found in Chapter 3. Operations I and Z are often expressed as unitary matrices in $|0\rangle / |1\rangle$ basis as

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

and together with matrices expressing bit-flip (X) and bit+phase-flip (Y) operations

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

they form the basis for size 2×2 complex matrices. The set of these four matrices are called Pauli matrices and they will play an important part throughout this thesis. Now, the above dephasing channel can be understood as follows. With probability $1 - \epsilon$ nothing happens to the quantum state, i.e. the identity channel is applied, while with probability ϵ the sign of the relative phase is flipped. Over the whole ensemble, these phase-flips result in losing the ability to extract information about the phase and when $\epsilon = 1/2$ the state has lost all of its information about it. This is what is meant by saying that the state ρ decoheres.

Using the same 4 Pauli matrices, it is easy to express another common noise channel which is called the depolarising channel. The depolarising channel is well suited to model a random effect of noise. For example, the authors of 2019 Google's supremacy experiment argued that, for a sufficiently random ensemble of quantum circuits, the average noise is well modelled by a depolarising channel after every gate [10]. Acting on a single qubit the depolarising channel is written as

$$\mathcal{D}_{\text{Pol}}(\rho) = (1 - \epsilon)\, I \rho I + \frac{\epsilon}{3}\, (X \rho X + Y \rho Y + Z \rho Z), \tag{1.8}$$

where again $\epsilon \in [0, 3/4]$ corresponds to the noise severity and I, X, Y, Z are the Pauli matrices. Similarly to before, this can be understood as leaving the state as it is with probability $1 - \epsilon$ or applying one of the non-trivial Pauli operations each with probability $\frac{\epsilon}{3}$. This channel corresponds to losing every kind of information about the system. Whenever $\epsilon = 3/4$, the resulting state is a fully mixed state ($\rho = \text{diag}(1/2, 1/2)$), and hence, all information about the initial state has been lost. Both of the noise models, that is, the dephasing and depolarising channels, are instances of Pauli error channels and are often used for modelling errors in individual circuit runs. These channels are crucial in testing the performance of quantum error correction protocols since they allow one to efficiently model quantum noise on classical software [11, 12]. I will expand on this point later when discussing quantum error correction.

Another common noise channel that is not expressed with Pauli matrices is the amplitude damping channel which is used to model qubit relaxation. Many quantum platforms, e.g. ion trap systems [13], have their qubits based on multi-level systems with different energies. Since lower energy levels are generally preferred, a relaxation to these levels takes place in a finite amount of time. For a single qubit, the amplitude damping channel is expressed in terms of Kraus operators as

$$\mathcal{D}_{\text{Am}}(\rho) = K_0 \rho K_0^\dagger + K_1 \rho K_1^\dagger, \tag{1.9}$$

where $K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix}$ and $K_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix}$ with γ describing the probability of relaxation. Here it is assumed that the $|0\rangle$ state is energetically preferred over the state $|1\rangle$. Interesting to note that if the qubit relaxation does not occur (the operator K_0 is implemented), the probability that the state is in the excited state $|1\rangle$ is lowered. Since no energy was transferred to the environment, the state is therefore more likely to be in the lower energy state [5].

All of the presented noise channels and more contribute to the errors in quantum computation. Knowing exactly the noise model might allow one to combat the effects of it. For example, protocols based on spin-echo or dynamical decoupling are frequently used to minimise the effects of decoherence [14, 15]. However, for large-scale quantum computation, a higher level of abstraction is necessary. As briefly mentioned above, quantum error mitigation (QEM) and quantum error correction (QEC) are two categories of algorithmic tools that provide such protection.

In the next chapter, I will briefly introduce QEC and leave the introduction to QEM to Chapter 3.

Chapter 2

Error correction

In this chapter, I will introduce several aspects of error correction. This will allow me to define notation and introduce preliminaries that will be assumed in the later chapters. I start with a brief introduction to classical error correction before turning to quantum error correction and fault-tolerance.

2.1 Classical error correction

Classical error correction first appeared as a means to communicate reliably, and one could say its origin is as far back as human civilisation goes. If a person is trying to convey a message through a noisy channel, for example by shouting across a rapid river, it may be wise to repeat the message multiple times. The action introduces redundancy that helps the receiver decipher the message. However, as a scientific field, classical error correction is believed to have started in the 1940s with pioneering work in the area by Richard Hamming. In 1950 he formulated what became known as the Hamming(7,4) code, a linear error-correcting protocol (code) that encodes 4 protected (logical) bits into 7 physical (data) bits [16]. That is, the code uses 3 additional bits of redundancy to protect the information encoded in the logical bits. This can be applied to either noisy communication or erroneous computation. The basis for error correction is to use this type of redundancy.

Codes like these are commonplace everywhere. For example, QR codes that are commonly used to access digital information are protected with an error-correcting code. Even if some part of the QR code is missing or is damaged, the information is still accessible. Similarly, barcodes on most items in a store are protected by error-detecting codes. These codes are simpler than error-correcting codes, but they still help to find whether an error has occurred while transmitting the information. If the barcode scanner has detected an error, the scan will fail and needs to be repeated. Let me now formalise some of these ideas by giving a rapid introduction to binary linear error-correcting codes. Since they are operationally close to the quantum stabiliser codes, they will form an elegant pathway towards introducing quantum error correction.

A classical binary linear code encoding k logical bits into n data bits is a k-dimensional subspace $\mathcal{C} \subseteq \mathbb{F}_2^n$, where $\mathbb{F}_2^n$ is a binary field with elements representing the bit-strings of all data bits. Each logical bit-string is associated with a vector in $\mathcal{C}$. These vectors are called codewords and $\mathcal{C}$ is called the codespace. The codespace is designed such that when a single or a few bit-flip error happens, any codeword is mapped to a vector outside of the codespace. This is crucial to detecting and correcting errors. The minimum number of bit-flip errors that take one codeword to another is called the distance of the code d. This implies that the code can reliably correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors on individual data bits. In compact notation, the code is denoted as an $[n, k, d]$ code. The codespace is often described as $\ker H$ of a binary matrix H called the parity check matrix, that is

$$\mathcal{C} = \{\, x \in \mathbb{F}_2^n \mid Hx = 0 \,\} . \tag{2.1}$$

For example, the aforementioned Hamming(7,4) code has a parity check matrix

$$H = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} , \tag{2.2}$$

while the distance $d = 3$ repetition code has a parity check matrix

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} . \tag{2.3}$$

The latter code encodes a single logical bit into three data bits with mappings $0 \to 000$ and $1 \to 111$. Each column j of the parity check matrix corresponds to a

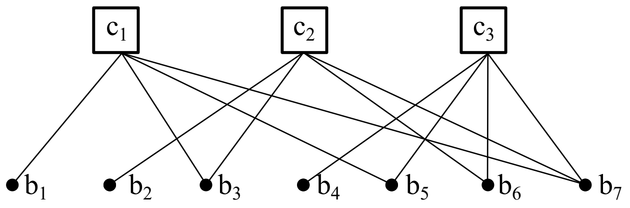


Figure 2.1: Bipartite Tanner graph of a classical Hamming(7,4) code. The squares represent parity checks c_i, while dots - data bits b_j. A check and a data bit are connected by a line iff the bit is in the support of the check.

data bit b_j and each row i to a parity check c_i. We say that a data bit b_j is in the support of a parity check c_i iff $H_{ij} = 1$. These checks are used to detect and find errors since only codewords satisfy all of them (see Eq. 2.1). For example, the check corresponding to the first row of Eq. 2.3 is only satisfied when the first two data bits have even parity, i.e. 00 or 11. If errors on some initial codeword map it to $x \notin \mathcal{C}$, the bit-string $s = Hx$ (called the syndrome) is by definition non-trivial. From it, a likely error $\overline{E}$ can be inferred such that $s = H(x + \overline{E}) = 0$ (i.e. $x + \overline{E}$ is a codeword again). This is usually done with classical software called the decoder. We say that the decoder has succeeded if the codeword $x + \overline{E}$ is the initial codeword, otherwise, it has failed and we result in a logical failure. In the latter case, we have lost the information we wanted to protect. Note, however, that larger codes with greater distances do not necessarily imply a lower logical failure rate. Distance is only one of several factors that influence the logical failure rate. For example, one should also consider how many error configurations of weight $\geq d$ induce a logical error.

Alternatively, one can describe a classical code with its *Tanner graph* - a bipartite graph whose biadjacency matrix is H. For example, the Tanner graph of Hamming(7,4) code is shown in Fig. 2.1. These graphs allow one to better visualise the connectivity between parity checks and data bits which is paramount in quantum error correction.

2.2 Quantum error correction

Quantum error correction is conceptually different from classical error correction for several reasons. First of all, since the space of qubit states is continuous, there is a continuum of bit-flip type errors that one needs to consider. Additionally, phase-flip type errors also need to be corrected. Surprisingly, discrete bit-flip and phase-flip error correction is sufficient to correct an arbitrary error. This comes from the digitisation of noise during the syndrome measurements that I will discuss later. I will denote single-qubit bit-flip and phase-flip errors as X and Z errors respectively, since they correspond to the actions of X/Z Pauli matrices. Second, measuring a quantum state is a destructive operation, and therefore, detecting errors needs to be done in a way that does not extract any information about the quantum state we want to protect. With all of these considerations, the first quantum code was proposed by Peter Shor in 1995 called Shor's 9 qubit code [17].

Similar to the classical case, a QEC code $\mathcal{C}$ encoding k logical qubits in n data qubits is a 2^k-dimensional subspace of the Hilbert space of all data qubits, i.e. $\mathcal{C} \subseteq \mathcal{H}_2^n$. The vectors of the codespace $\mathcal{C}$ are called codewords (or logical states) and the minimum number of errors that takes any codeword to a different codeword is called the distance d. This implies that the code can reliably correct up to $\left\lfloor \frac{d-1}{2} \right\rfloor$ errors on individual data qubits. In compact notation, a quantum code is denoted as an $[\![n, k, d]\!]$ code.

2.2.1 Stabiliser codes

Specifying the codespace of a quantum code is a tedious task, and instead one can define a QEC code by a stabiliser group S [18]. It is an abelian group whose elements M, called stabilisers (or parity checks), are operators that fix (map to itself) any codeword $|\psi\rangle_L$. Hence, S defines the code $\mathcal{C}$ in the following way

$$\mathcal{C} = \{|\psi\rangle_L : M|\psi\rangle_L = |\psi\rangle_L, \forall M \in S\}. \tag{2.4}$$

In most cases, S is taken as an abelian subgroup of an n-qubit Pauli group

$$\mathcal{P}_n = \langle i, X_j, Z_j \mid j \in [n] \rangle = \left\{ \phi \bigotimes_{j=0}^{n} P_j \right\},$$

where $\phi \in \{\pm 1, \pm i\}$ and P is a single qubit Pauli operator $P \in \{I, X, Y, Z\}$. A peculiar thing about Pauli operators is that they either commute or anti-commute,

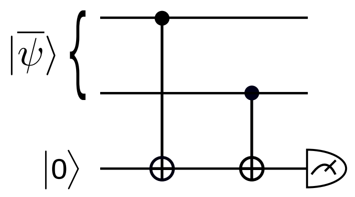


Figure 2.2: A circuit to extract the eigenvalue of ZZ stabiliser. The third (bottom) qubit is an ancilla qubit that is entangled with the data qubits of the code and finally measured. The entangling gates are CNOT gates that are explicitly defined later in the chapter.

therefore, it is easy to see why Eq. 2.4 defines a necessarily abelian group. For if any two operators $P, Q \in S$ anti-commuted we would have

$$|\psi\rangle_L = PQ\,|\psi\rangle_L = -QP\,|\psi\rangle_L = -\,|\psi\rangle_L \tag{2.5}$$

for all codewords $|\psi\rangle_L$. This can only be satisfied when $|\psi\rangle_L = 0$, implying that there is no protected codespace at all. Any Pauli operator that commutes with all elements of S, but is not contained in S, is called a logical operator. These are operators that take a codeword to a different one.

The stabiliser group forms the basis for syndrome extraction to detect and correct errors. This is done by measuring the eigenvalues of stabiliser operators over the data qubit space, e.g. see Fig. 2.2 where a circuit to extract the eigenvalue of a two-qubit $Z \otimes Z \equiv ZZ$ stabiliser is given. We say that the qubit is in the support of a stabiliser if the stabiliser acts non-trivially on it. Analogously to the classical case, a negative eigenvalue outcome of any stabiliser indicates an error. In fact, it is sufficient to only measure the elements of the stabiliser generating set $\mathcal{S} = \langle S_1, \ldots, S_m \rangle$. These are called stabiliser generators, and the collection of measurement outcomes is called the syndrome. Given m independent generators, the common $+1$ eigenspace (codespace) of all elements of S has size $\mathcal{C} = \mathcal{H}_2^{n-m}$ and encodes $k = n - m$ qubits. Simply put, every additional stabiliser generator halves the Hilbert space size of data qubits. Importantly, when measuring the eigenvalue of any stabiliser operator, no information about the logical state is extracted as every operator stabilises all logical states in the same fashion. This ensures that the logical state does not collapse as part of the measurements.

Just like in the classical case, to infer a likely error E, classical software called the decoder is used. There are many decoders that are used in practice, e.g. see [19–21], however, most of them operate by finding the smallest weight error E_{min} that fully explains the syndrome data. Such decoders have good performance since majority of errors in nature are localised, e.g. two errors on non-interacting qubits are less likely than a single error. The design and choice of the decoder are intricately tied to the quantum code and the details of the error model one assumes.

Earlier I mentioned that only X and Z Pauli errors need to be corrected due to the digitisation of noise during the stabiliser measurements. To briefly explain this, consider a unitary error operation $R_x(\theta) = \cos(\theta/2)I - i\sin(\theta/2)X$ on the top qubit of the ZZ stabiliser circuit given in Fig. 2.2. The three-qubit state before both CNOT gates is given as

$$(\cos(\theta/2)I \otimes I \otimes I - i\sin(\theta/2)X \otimes I \otimes I) \left|\overline{\psi}\right\rangle \left|0\right\rangle, \tag{2.6}$$

while after them

$$\begin{aligned}(\cos(\theta/2)I \otimes I \otimes I - i\sin(\theta/2)X \otimes I \otimes X) \left|\overline{\psi}\right\rangle \left|0\right\rangle &= \\ = \left|\overline{\psi}\right\rangle \otimes \cos(\theta/2)\left|0\right\rangle - (X \otimes I)\left|\overline{\psi}\right\rangle \otimes i\sin(\theta/2)\left|1\right\rangle. \end{aligned} \tag{2.7}$$

When we measure the bottom qubit in $\left|0\right\rangle / \left|1\right\rangle$ basis, its value will be 0 with probability $\cos^2(\theta/2)$ or 1 with probability $\sin^2(\theta/2)$. Respectively, the measurement collapses the encoded state to either $\left|\overline{\psi}\right\rangle$ or $X \otimes I \left|\overline{\psi}\right\rangle$. The former outcome indicates no error while the latter is equivalent to a discrete bit-flip (X) error on the first qubit. This generalises to arbitrary errors and digitises them on individual qubits as I, X, Y, Z errors during the Pauli stabiliser measurements. Since identity error corresponds to a trivial error and Y error is expressed as $Y = iXZ$, correcting X and Z errors is sufficient. This also implies it is easy to correct errors as any Pauli operator is its own inverse. A correction of an error is equivalent to an application of the same error. In practice, the inferred errors are not corrected on the experimental device but are only noted in software, they are used once a logical qubit measurement needs to be interpreted.

Stabiliser generators	
S_1	$X{\otimes}I{\otimes}X{\otimes}I{\otimes}X{\otimes}I{\otimes}X$
S_2	$I{\otimes}X{\otimes}X{\otimes}I{\otimes}I{\otimes}X{\otimes}X$
S_3	$I{\otimes}I{\otimes}I{\otimes}X{\otimes}X{\otimes}X{\otimes}X$
S_4	$Z{\otimes}I{\otimes}Z{\otimes}I{\otimes}Z{\otimes}I{\otimes}Z$
S_5	$I{\otimes}Z{\otimes}Z{\otimes}I{\otimes}I{\otimes}Z{\otimes}Z$
S_6	$I{\otimes}I{\otimes}I{\otimes}Z{\otimes}Z{\otimes}Z{\otimes}Z$

Table 2.1: Stabiliser generators S_i for a 7 qubit Steane code. Both X and Z denote single-qubit Pauli matrices. The tensor product sign $\otimes$ together with the identity matrices are often omitted when the respective data qubit is specified, e.g. $S_1 = X{\otimes}I{\otimes}X{\otimes}I{\otimes}X{\otimes}I{\otimes}X$ is written as $S_1 = X_1X_3X_5X_7$.

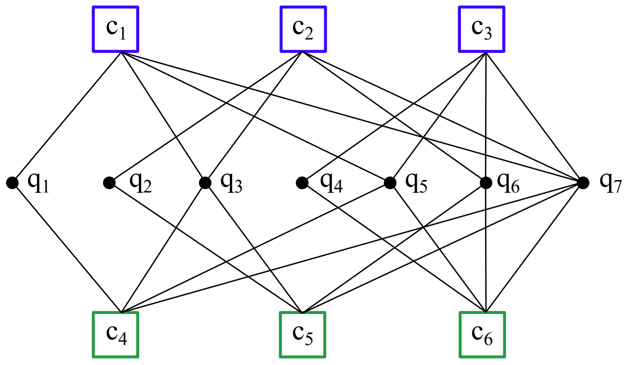


Figure 2.3: Tripartite quantum Tanner graph of a 7 qubit Steane code. Data qubits are denoted by q_i, while c_j denote the X (blue) and Z (green) parity checks (stabilisers). Edges are drawn between them iff the data qubit is in the support of the stabiliser.

Similar to the classical case, we say the decoder fails and we have created a logical error whenever the correction together with the initial error leads to a logical operator. Such an operator is undetected as it commutes with all stabilisers, and, therefore, we have irretrievably lost information about the logical state.

2.2.2 CSS codes

An important subclass of stabiliser codes are Calderbank-Shor-Steane (CSS) codes, which will be considered in all chapters of this thesis. These are stabiliser codes where the non-identity operators for each stabiliser generator are either all X or all Z single-qubit Pauli operators. The two types of stabilisers are called X and Z stabilisers respectively. The commutation is ensured by every pair of Z and X stabilisers sharing an even number of qubits in their support. In one of the later chapters, I will use CSS codes in relation to classical error-correcting codes,

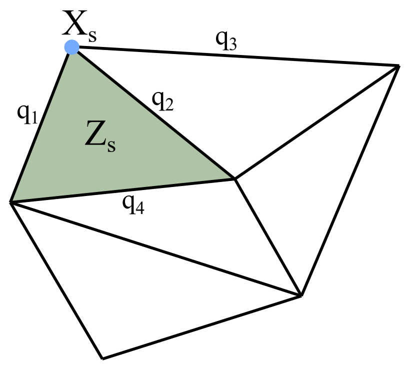


Figure 2.4: Many quantum CSS codes can be compactly represented as a tessellation in some dimensional space. Data qubits q_i are placed on edges, while X_s and Z_s stabilisers are placed on vertices and faces respectively. Some qubits and stabilisers are highlighted. Note that this is a pseudo-random tessellation.

where the relationship comes from the fact that these codes detect X and Z errors separately. As an example, a 7 qubit Steane code is a CSS code with stabiliser generators given in Table. 2.1. Let me introduce a couple of other representations using the Steane code as an example.

Similar to classical codes, quantum CSS codes can be represented with a Tanner graph. In this case, it is a tripartite graph where two sets of nodes correspond to X and Z stabilisers and the final set corresponds to the data qubits. An edge is drawn if the qubit is in the non-trivial (non-identity) support of the stabiliser. The Tanner graph of the Steane code is shown in Fig. 2.3.

Another, similar representation of a CSS code is to associate a tessellation in some dimensional space. In two dimensions, qubits are placed on edges, X (resp. Z) stabilisers on vertices (resp. faces), see Fig. 2.4. This perspective allows one to confirm that the stabilisers commute since any vertex share exactly two or zero edges with any face. However, some codes that are represented in this way do not form a proper tessellation in the graph-theoretic sense. For example, if one wanted to represent a 7 qubit Steane code (Table. 2.1) in the same fashion, there would exist hyperedges that connect to either 1 or 3 vertices. It is worth mentioning that there is another representation involving homological algebra that is handy when analysing and constructing CSS codes. However, I leave the introduction of this perspective to a later chapter in the thesis.

2.2.3 Simulation of quantum codes

The most robust way to test the performance of an error-correcting code would be to use it in an experiment. This, however, is not often possible since current quantum computing devices are too small to test codes of larger sizes. The current efforts have only reached demonstrations of codes with at most $n = 25$ data qubits [3, 22, 23]. Instead, simulation of quantum codes on classical devices is adopted, and, fortunately, this is not as difficult as simulating quantum computers themselves. Pauli stabiliser measurement circuits (e.g. Fig. 2.2) contain only a very specific set of gates - the Clifford gates. They are defined on n as

$$\mathbb{C}_n = \{V \in U(2^n) \mid V\mathcal{P}_nV^\dagger = \mathcal{P}_n\}, \tag{2.8}$$

where $U(2^n)$ describes all unitary gates on n qubits. In simple terms, Clifford gates are those that map a Pauli group to itself. If, in addition, the simulated error model can be constructed from Clifford gates, then the code can be efficiently simulated on a classical device as per the Gottesman-Knill theorem [11, 12]. It states that any Clifford circuit, i.e. a circuit consisting of only Clifford gates, can be simulated on a classical computer in a polynomial time with respect to the circuit size.

The popular error models for simulations include the following:

1. **IID X/Z noise.** Each stabiliser measurement round every data qubit independently undergoes a phase- or bit-flip error with some probability p. The model can be made stronger by adding that in each round the stabiliser measurement outcome is flipped with probability q, imitating a noisy measurement. In this case, it is called the phenomenological noise model.

2. **Depolarising model.** Each stabiliser measurement round every data qubit undergoes a depolarising noise channel with some severity p. This model may be supplied with noisy measurements as described above.

3. **Circuit noise.** This is the most comprehensive error model in which every circuit operation, i.e. state initialisation, measurement, single and two-qubit gates and sometimes even identity channels, is considered erroneous. Gates in

the stabiliser readout circuits are modelled as perfect gates with a Pauli error channel applied after. The measurement at the end of the circuit is modelled in the same way as in the phenomenological noise model.

All of these example models can be simulated with Clifford circuits and therefore the Gottesman-Knill theorem applies. That being said, the decoder can become the bottleneck for simulating larger codes or running them on a quantum computer. For example, a large class of codes are called quantum LDPC (low-density parity-check) codes (see Chapter 5) for which the running time of a state-of-the-art decoder that is reliable and practical scales as n^3. The repeated decoding task becomes quickly impractical and expensive for $n \geq 10^5$.

So far, we have considered using QEC codes to protect the encoded information from general errors. Sometimes this is referred to as memory QEC. To implement large-scale quantum algorithms, in addition to reliable storage, we also require the capability to manipulate quantum information in a fault-tolerant way. This is called fault-tolerant quantum computation.

2.3 Fault-tolerant quantum computation

I would say that there is not a strictly agreed definition of fault-tolerance as most QEC practitioners have a slightly altered view of what they consider as fault-tolerant. However, the main idea, that most of the definitions are built around, can be encapsulated with two points. First, there exists a QEC code that can achieve some target logical failure rate. Second, logical information can be manipulated with a universal set of *fault-tolerant gates* or operations in general. Satisfying these two points, one is able to correct errors as quickly as they are introduced and perform an arbitrary quantum computation with a vanishing probability of failure. Let me now expand on the second point. The first point I leave to Chapter 4 where I introduce the threshold theorem and describe the necessary assumptions to achieve an arbitrarily low logical failure rate.

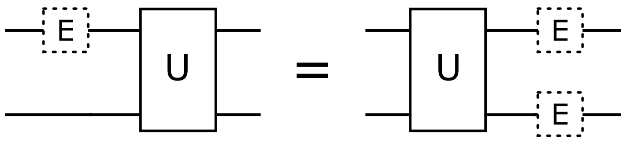


Figure 2.5: A single unitary gate U applied after a single error E can be equivalent to two errors happening after the application of U. We say that the unitary gate has spread the error.

First of all, logical operations are analogous to the operations defined on a quantum circuit of qubits. Instead of acting on physical qubits, they act on the logical qubits encoded in some code. The logical qubit states are the codewords and, therefore, logical gates map one codeword to another. Now, a fault-tolerant gate is defined as a logical gate that does not spread errors within the same code (or code block). By spreading errors I mean the following. Consider a logical state $|\psi\rangle$ with a single qubit error X and a two-qubit gate U that is applied after, that is $U(X \otimes I)|\psi\rangle$. We say that U spreads or propagates the error if

$$U(X \otimes I)|\psi\rangle = (X \otimes X)U|\psi\rangle = (X \otimes X)|\phi\rangle\,, \tag{2.9}$$

where $|\phi\rangle = U|\psi\rangle$. In words, the action of U on $|\psi\rangle$ spreads the singe-qubit error to another qubit in the system. If both of these qubits are within the same QEC code block, then U is not a fault-tolerant gate. In fact, the RHS of Eq. 2.9 does not necessarily need to have two X errors, any two non-identity Pauli errors would imply the propagation of an error. Fig. 2.5 shows the process in the circuit format. The generalisation from fault-tolerant gates to operations is straightforward, however, there is much more to say about fault-tolerant stabiliser measurements. For this, I refer the reader to [5] (Section 10.6).

Transversal gates are a class of logical gates that are intrinsically fault-tolerant. These logical gates are implemented by implementing the respective physical gate on all data qubits separately or, in the case of multi-qubit logical gates, between the respective data qubits of two or more separate codes. Fig. 2.6 shows implementations of two transversal logical gates for the 7 qubit Steane code - a single-qubit Hadamard gate

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{2.10}$$

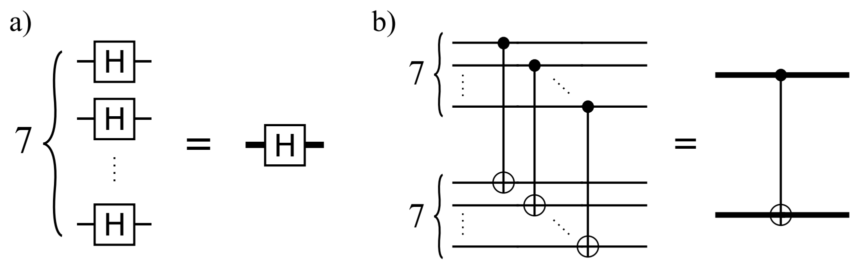


Figure 2.6: Fault-tolerant implementation of two logical gates for a 7 qubit Steane code. (a) A Hadamard gate and (b) a CNOT gate. In both, the narrow lines denote the data qubits and the thick line denotes the logical qubit.

and a two-qubit controlled-NOT (CNOT) gate

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2.11}$$

It is worthwhile noting that these two gates together with a phase-gate

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{2.12}$$

generate the Clifford group, i.e. $\mathbb{C}_n = \langle H_i, S_i, \mathrm{CNOT}_{ij} \rangle$, where the subscripts index the qubit/s the gate is acting on. All of the above matrices are expressed in the computational basis.

Unfortunately, every code has a limited set of transversal gates. By Eastin-Knill theorem [24], there does not exist a quantum error-correcting code that has a universal set of transversal gates. Therefore, one needs to find other ways to complete the universal set of fault-tolerant gates to implement an arbitrary computation in a fault-tolerant way. As an example, in Chapter 4, I will introduce a code called the surface code for which the logical gates H, S and CNOT can be applied transversally. Therefore, any logical Clifford gate can be implemented fault-tolerantly. With such Clifford gates, it is possible to perform magic state distillation [25] and teleportation [26] circuits to implement a $T = \sqrt{S}$ gate in a fault-tolerant fashion. Since any unitary gate can be decomposed into a set of Clifford + T gates [5], this constitutes a universal gate set.

With this, I conclude the presentation of preliminaries and notation used in this thesis. More specialised introductions are given at the beginning of each thesis chapter.

The following chapter is based on an article titled "*Learning-based quantum error mitigation*" [27]. The paper was authored by Dayue Qin, Yanzhu Chen, Simon C Benjamin, Ying Li and myself. Both I and DQ shared the first author role. YL conceived the main idea of using learning-based techniques for performing quantum error mitigation and provided operational proofs. This chapter will focus on my contributions: I devised and explored a practical approach that we call the significant error approach. In addition, I performed most of the numerical simulations found in [27]. For other ideas that were explored by my co-authors please see [27]. All authors contributed to the writing of the manuscript. SCB and YL supervised the project throughout.

Chapter 3

Learning-based quantum error mitigation

In the previous chapter, I argued that an additional algorithmic layer of protection will be necessary to reduce noise in any large-scale quantum computation. Furthermore, I suggested that quantum error correction (QEC) is a scalable solution and that quantum error mitigation (QEM) might find its uses for medium size systems. In this chapter, I will further expand on the latter and leave the more rigorous subject of QEC for later chapters.

This chapter is structured as follows. First, I will briefly introduce the two most common error mitigation strategies and explore their uses. Second, I will introduce a new variant behind one of the strategies and show how it combines the learning of a system's noise profile with probabilistic error cancellation. Finally, I will develop a practical implementation of the proposed new variant.

3.1 Introduction to quantum error mitigation

In many potential quantum applications, for example, in the use of variational quantum eigensolvers or simulators [28–30], a key task is to evaluate mean values of some observables, i.e. to measure the expected value of one or more qubits at the end of some circuit. As I explained before, the estimator of the mean is usually biased as a result of noise. The role of all error mitigation strategies is to use classical post-processing routines to obtain a less biased or equivalently less noisy estimator of some target computation. This generally involves some guesswork about the characteristics of noise that is present during the computation. Moreover, to obtain such estimators, QEM protocols rely on an increased amount of sampling and/or computations which scale exponentially with the system size. Such an unfavourable scaling limits the practicality of QEM as an all-in-one solution for large-scale quantum computation, however, it may be beneficial in near-term applications or even in fault-tolerant subroutines.

In this section, I will introduce two error mitigation strategies - error extrapolation and probabilistic error cancellation [29, 31, 32]. In the former, the task is to find two or more estimators of an observable from the same computation but using different noise strengths. Then given this set, one may find the estimator at the zero-noise limit by extrapolation. In the latter, an unbiased estimator of some computing result is carefully constructed using the knowledge of the error model of the quantum circuit. The task is performed by randomly sampling a particular set of quantum circuits (derived from the target circuit) and taking a properly weighted average over the obtained measurement outcomes.

3.1.1 Zero-noise extrapolation

Let me start with one of the simplest QEM protocols called zero-noise or Richardson's extrapolation. The protocol was first introduced in 2016 by Y. Li and S. Benjamin [29] and independently in a separate paper by K. Temme, S. Bravyi and J. Gambetta [31]. While both works present the method in a slightly different fashion, the underlying theory of extrapolation is the same.

The protocol requires obtaining a set of noisy estimators $E(\lambda)$ of some observable of interest, where λ is the predicted noise strength. The variable λ, for example, can refer to the average gate error rate over the circuit, and it is preferential that its value is known to high precision. Using the noisy estimators $E(\lambda)$, the goal is to predict the noise-free estimator which I denote as E^*. This is done by extrapolating to $\lambda = 0$ according to a function chosen by the user. More specifically, as derived in [31], the Richardson's extrapolation uses the following

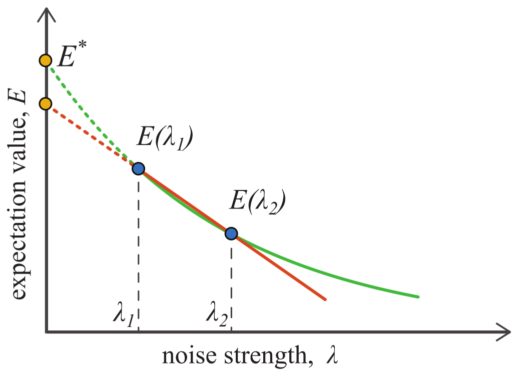


Figure 3.1: Linear (red line) and exponential (green line) variants of Richardson's extrapolation. When the noise is of unital nature the exponential variant heuristically outperforms the linear variant.

degree n polynomial expansion of a noisy estimator

$$E(\lambda) = E^* + \sum_{k=1}^{n} a_k \lambda^k + R_{n+1}, \tag{3.1}$$

where a_k are the constants to be found and R_{n+1} is the remainder that collects higher order terms. The above polynomial (up to some chosen degree n) is then taken as the fitting function for extrapolation, where the remainder R_{n+1} will contribute to the remaining error. Higher n means a more accurate estimate but comes with a drawback that at least $n+1$ samples of $E(\lambda)$ over different values of λ are required to fit the polynomial.

Two variants of this method have appeared in the literature - a linear extrapolation variant, in which we take $n = 1$, and an exponential variant [32–34]. The former only requires performing experiments at two different noise levels (as only two points are needed to fit a straight line), while in the latter case, it depends on the exact form of the exponential function. Surprisingly, due to the nature of random quantum noise being well modelled by unital mappings, such as depolarising or dephasing channels introduced in Chapter 1 [10], the exponential extrapolation variant with as few as two data points is frequently sufficient and heuristically yields a better estimation of E^* than its linear counterpart, see Fig. 3.1. Some reasoning on why unital noise is well modelled by an exponential decay with respect to the noise severity can be found in [32].

Extrapolation QEM may also be applied to algorithmic errors in the same way as to physical ones [35]. Some examples of algorithmic errors include the truncation of Trotterization sequences and inaccurate guesses for noise strengths in the probabilistic noise cancellation scheme that I will describe next [36]. That being said, extrapolation QEM cannot be expected to perfectly mitigate errors in practice. The main two challenges are accurately choosing the curve to fit and uniformly adjusting the noise parameter [37].

3.1.2 Probabilistic error cancellation

Another widely explored QEM strategy is called probabilistic error cancellation or sometimes referred to as "quasi-probability QEM" to emphasise its use of quasi-probability distributions. In this subsection, I will briefly explain the working principles of this method as introduced in 2016 by K. Temme et al. [31] and later by S. Endo et al. [32]. Assume a quantum device has noisy operations $\mathcal{O}$ such as qubit initialisation, one and two-qubit gates and measurements. If the set of available noisy operations is large enough to form a complete basis of CPTP maps acting on some qubit subspace ρ, then any noise-free operation $\mathcal{V}(\rho)$ can be written as a linear combination of these given operations

$$\mathcal{V}(\rho) = \sum_i \eta_i \mathcal{O}_i(\rho). \tag{3.2}$$

Therefore, a noise-free quantum circuit may be decomposed into many noisy circuits each weighted by a factor η_i. Sampling the noisy circuits and performing classical post-processing allows one to find the expectation values as if they were computed using the noise-free circuit.

Finding the set of noisy operations that together describe a noise-free operation may be challenging in general. Indeed, even describing the noisy operations themselves is already a difficult task. For example, take a noisy unitary gate U'. We can model it as $U'(\rho) = NU(\rho)$, where U is the noise-free equivalent of the noisy gate and N is a channel that fits our experimental noise model. To precisely characterise the noise channel some form of tomography must be used [32]. This may be inaccurate [38] or infeasibly costly unless error correlations involve only a few qubits.

Assuming the expensive tomography task has been performed and the channel N is known to a desired precision, one can classically derive its inverse map N^{-1}. Then, the wanted decomposition of the ideal unitary gate U is given as

$$N^{-1}U'(\rho) = N^{-1}NU(\rho) = U(\rho). \tag{3.3}$$

This is exactly the sequence of maps that one should implement in their experiment to perform a noise-free unitary gate. Note that, in general, applying N^{-1} will introduce some additional noise, however, only marginally in most practical cases. As you will find in the example of a depolarising channel below, the inverse map of a noise channel with low severity has a large overlap with the identity channel. Since the identity channel physically requires no application of any operation, the inverse map introduces minimal noise.

Unfortunately, simply cancelling the noise channel using its inverse is not trivial. In general, channel N is not a unitary process hence its inverse is not unitary either. A quantum circuit only admits unitary gates, therefore, to physically implement the inverse map N^{-1}, it first needs to be decomposed into a probabilistic distribution of unitary operations. Then the map is implemented by probabilistically sampling the operations across multiple runs of the circuit. However, even that cannot be done in most cases! Generally, the inverse map N^{-1} is not a quantum channel (a CPTP map). Therefore, it does not have a decomposition into a probabilistic distribution of unitary operations. In simple terms, the distribution has probability coefficients that are negative or larger than one, hence the common name of quasi-probability QEM.

As an example, recall the single-qubit depolarising channel from Chapter 1

$$\mathcal{D}_{\text{pol}} = (1-\epsilon)[I] + \frac{\epsilon}{3} \sum_{\mu \in \{X,Y,Z\}} [\mu], \tag{3.4}$$

where the operators, for example $[X]$, act on the sate ρ as $[X](\rho) = X\rho X^{\dagger}$ and ϵ is the noise severity parameter. Then its classically derived inverse is the following map

$$\mathcal{D}_{\text{pol}}^{-1} = \eta_1[I] + \eta_2 \sum_{\mu \in \{X,Y,Z\}} [\mu], \tag{3.5}$$

where $\eta_1 = 1 + \frac{3\epsilon}{3-4\epsilon}$ and $\eta_2 = -\frac{\epsilon}{3-4\epsilon}$. One can see that the inverse is not defined for $\epsilon = \frac{3}{4}$. At this noise severity the channel describes a fully depolarising map where all information about the quantum state is lost. Since this is an irreversible operation, the inverse of the channel at $\epsilon = \frac{3}{4}$ does not exist.

The coefficients η are called quasi-probabilities, that is, they meet one of the conditions that the probabilities satisfy. Namely, they sum to identity but often fail to meet the requirement that all individual probabilities must be nonnegative. So how does one implement the inverse map in practice given such quasi-probability distribution? The trick is to convert the set of quasi-probabilities η_i into a valid probability vector with coefficients $|\eta_i|/\gamma$. Here, $\gamma = \sum_i |\eta_i| \geq 1$ is called the overhead factor. Then the noise-free operation can be written as

$$U(\rho) = N^{-1}U'(\rho) = \sum_i \eta_i \mathcal{O}_i(\rho) = \gamma \sum_i \frac{\eta_i}{|\eta_i|} \left\{ \frac{|\eta_i|}{\gamma} \mathcal{O}_i(\rho) \right\}, \tag{3.6}$$

where the curly brackets indicate a physically implementable process and everything else describes a classical post-processing step. In the rest of the chapter, I will mostly neglect the noisy unitary gate U' in such decompositions since it has the same action in each term. Only the decomposition of N^{-1} is crucial here.

The physical implementation of the ideal operation (Eq. 3.6) can be done with a straightforward protocol. A noisy operation $\mathcal{O}_i$ is classically sampled with probability $|\eta_i|/\gamma$ and is applied in place of the noisy unitary gate U' for a single run of the target computation. The result of the computation is classically multiplied by the sign of the corresponding η_i. After many runs, the average of all computational results is taken as the error-mitigated result. This, for example, could be the estimator of the noise-free expectation value

$$E = \mathrm{Tr}(A...N^{-1}U'...\rho) = \gamma \sum_i \eta_i/|\eta_i| \{ |\eta_i|/\gamma \mathrm{Tr}(A...\mathcal{O}_i...\rho) \} \tag{3.7}$$

for some observable A. The same protocol can be trivially extended to the case of multiple inverse maps.

In the scenario where the inverse maps are applied perfectly the estimator is unbiased as shown in Fig. 3.2. However, due to the overhead factor γ, the number

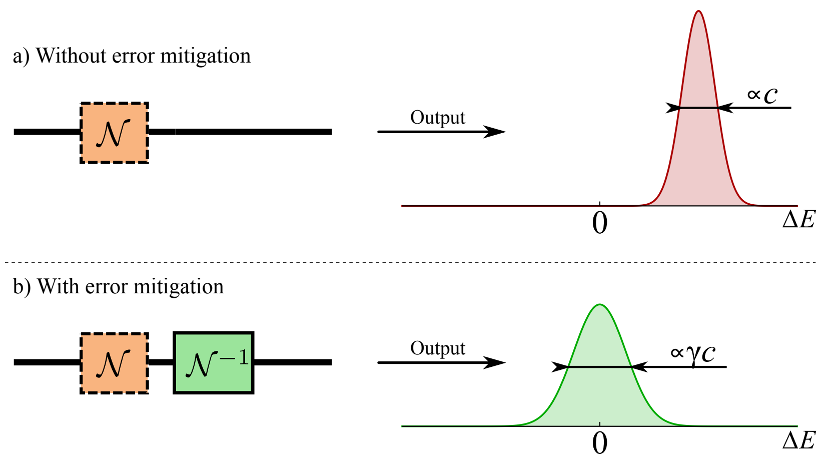


Figure 3.2: Noisy process without (a) and with (b) probabilistic error cancellation and their respective estimators of some expectation value. ΔE characterises the deviation from the ideal expectation value. Introducing error mitigation results in a larger variance of the estimator but minimises its bias.

of samples that are needed to estimate some observable to a given precision is increased by a factor of γ^2 compared to the noise-free case. Since each application of the inverse map multiplicatively increases the overhead factor, the probabilistic noise cancellation QEM strategy scales exponentially with the number of noisy operations in the circuit. Therefore, as mentioned before, it is not a feasible solution for dealing with noise in large-scale quantum computation.

That being said, probabilistic error cancellation has been shown to perform well for small scale experiments on different platforms. For example, it has been successfully demonstrated in systems of superconducting qubits and trapped ions [39, 40].

3.2 Learning-based quantum error mitigation

In this section, I will present a novel and intuitive variant to mitigate errors called learning-based quantum error mitigation (LBQEM). The ideas in this section and the next are based on my and my co-authors' work [27].

To introduce the protocol, consider using probabilistic error cancellation to mitigate errors in quantum computation. Instead of determining the noise model that afflicts the experimental system and decomposing the ideal operations in terms of noisy ones (i.e. by finding the quasi-probability distribution of some set of operations), we propose to determine the decomposition via an *ab initio* learning process. Here, the quasi-probability distribution is found by minimising the error in the final computing result for a set of training circuits. As I will show later, this provides us with a simple and efficient protocol due to its simplicity and, also, the ability to mitigate complex noise profiles. All possible error correlations, i.e. spatial and temporal correlations, are automatically taken into account in the learning process.

A challenge for the learning-based quantum error mitigation at scale is that the ideal computing result for a general training circuit cannot be determined without an ideal execution of the quantum circuit itself. However, as Ying Li has shown (see Subsection 3.2.2), LBQEM is in fact feasible because efficiently simulable Clifford circuits [11, 12, 41] (see Chapter 2) as training tasks are sufficient to find the optimal quasi-probability distribution. The sufficiency is proved under the assumption of negligible single-qubit gate errors. This is a valid assumption since on most quantum platforms, single-qubit gates have the highest fidelity out of all operations. For example, an average single-qubit gate fidelity of 99.9999% has been demonstrated with trapped ions [42] while the record fidelity of two-qubit gates is three orders of magnitude lower at 99.9% [43, 44].

3.2.1 The general protocol

Let me now describe the protocol in detail. Consider a quantum circuit as shown in Fig. 3.3. Here, all qubits are initialised in the state $|0\rangle$ and measured in the computational Z basis at the end. Due to the assumption of ideal single-qubit gates, the errors are only caused by multi-qubit gates, e.g. controlled-NOT and controlled-phase gates, state initialisation and measurement. We collectively denote these operations as frame operations and refer specifically to multi-qubit gates as frame gates.

Suppose that the circuit has n qubits and N layers of frame gates, for example, there are $N = 2$ layers of frame gates in Fig. 3.3 (dashed boxes). For the protocol to work, the only requirement we assume for frame gates is that they are Clifford gates. Between frame operations arbitrary single-qubit unitary gates R_i (blue in Fig. 3.3)

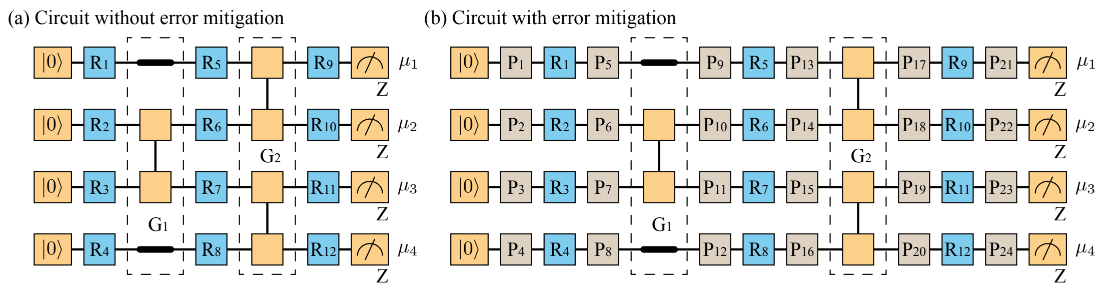


Figure 3.3: Example circuit without (a) and with (b) error mitigation. Each circuit has four qubits and two layers of frame gates G_1 and G_2 (dashed boxes). Note that we do not assume anything about the nature of errors afflicting the frame gates. They might be correlated over many qubits or even have correlations between the dashed boxes (i.e. spatial and temporal errors). Frame operations (orange) include the qubit initialisation, frame gates and measurement. There are three layers of single-qubit unitary gates R_i (blue) in each circuit. (b) To implement error mitigation, two layers of Pauli gates P_i (brown) are introduced before and after each layer of single-qubit gates. Usually, adjacent R_i and P_i gates are combined in their physical implementation. This figure was taken from [27].

can be performed that fully specify the quantum computation. We collectively denote the set of all single-qubit unitary gates by $\boldsymbol{R} = (R_1, R_2, \dots, R_{n(N+1)})$. To perform error mitigation, we introduce a new set of single-qubit Pauli gates P_i before and after each single-qubit unitary gate (see Fig. 3.3(b)). These will act as error-mitigation gates and are collectively denoted by $\boldsymbol{P} = (P_1, P_2, \dots, P_{2n(N+1)})$. A proper combination of these gates will form an inverse noise map as described in the previous section. Both single-qubit unitary gates $\boldsymbol{R}$ and Pauli gates $\boldsymbol{P}$ are treated as variables, that is, they can be replaced by other set of gates if necessary. The frame gates, on the other hand, are always fixed in the same configuration. Let me remark that circuits composed in this way are universal for quantum computing.

Take a circuit with frame operations fixed and with single-qubit gates $\boldsymbol{R}$ and $\boldsymbol{P}$. With this circuit, a key task might be to reliably evaluate the mean value of some computing result. For example, if one wants to estimate the expectation value of observable $Z = \mathrm{diag}(1, -1)$ on the first qubit, then the result we want to compute is $E = 1 - 2\mu_1$, where μ_1 is the state outcome of the first qubit measured in $|0\rangle \,/\, |1\rangle$ basis. Throughout, I will use $E^{\mathrm{ef}}(\boldsymbol{R}, \boldsymbol{P})$ to denote the mean value of such results when the circuit is error-free and $E(\boldsymbol{R}, \boldsymbol{P})$ to denote the mean value from the actual noisy circuit.

As explained above, in probabilistic error cancellation, a linear combination of noisy computing results is used to estimate the error-free result, e.g. some expectation value. In our setup, these results are obtained using different configurations of error-mitigating Pauli gates $\boldsymbol{P}$. Therefore, denoting quasi-probability coefficients by $\eta(\boldsymbol{P})$, the error-mitigated result can be derived as

$$E^{\mathrm{em}}(\boldsymbol{R},\boldsymbol{I})=\sum_{\boldsymbol{P}}\eta(\boldsymbol{P})E(\boldsymbol{R},\boldsymbol{P}), \tag{3.8}$$

where $\boldsymbol{I}$ means that all error-mitigating gates are identity. One can further define the computing error compared to the error-free result as

$$\mathrm{Error}(\boldsymbol{R})\equiv\left|E^{\mathrm{em}}(\boldsymbol{R},\boldsymbol{I})-E^{\mathrm{ef}}(\boldsymbol{R},\boldsymbol{I})\right|, \tag{3.9}$$

The goal is to find the optimal distribution $\eta_{\mathrm{opt}}(\boldsymbol{P})$ such that the error is minimised. As the main idea of LBQEM protocol, the optimisation is done using Clifford training circuits that are constructed by replacing the single-qubit unitary gates $\boldsymbol{R}$ with Clifford gates. The rest of the circuit structure and the gates therein are kept in their original form. Stated formally, consider a loss function in the quadratic form,

$$\mathrm{Loss}\equiv\frac{1}{|\mathbb{T}|}\sum_{\boldsymbol{R}\in\mathbb{T}}\mathrm{Error}(\boldsymbol{R})^2, \tag{3.10}$$

where $\mathbb{T}$ is a set of training computing tasks. To evaluate the loss function, $E(\boldsymbol{R},\boldsymbol{I})$ can be computed using the noisy quantum computer and $E^{\mathrm{ef}}(\boldsymbol{R},\boldsymbol{I})$ using a classical computer. Since Clifford circuits can be efficiently simulated on a classical computer according to the Gottesman-Knill theorem, it is cost-effective to choose the training set $\mathbb{T}$ as a subset of Clifford circuits, i.e. $\mathbb{T}\subseteq\mathbb{C}$, where all $R_i\in\mathbb{C}$ are Clifford gates. By minimising the loss function, one can find the optimal distribution $\eta_{\mathrm{opt}}(\boldsymbol{P})$ for the training set. Then, by applying Pauli gates according to the same distribution $\eta_{\mathrm{opt}}(\boldsymbol{P})$ to our primary circuit(s) with *arbitrary* single-qubit gates $\boldsymbol{R}$, we can find the optimal error-mitigated result $E^{\mathrm{em}}_{\mathrm{opt}}(\boldsymbol{R},\boldsymbol{I})$ (see Eq. 3.8). Recall that $\boldsymbol{R}$ will be non-Clifford in non-trivial quantum computations.

3.2.2 Information completeness

One may wonder whether Clifford circuits as training data provide sufficient information to find the optimal quasi-probability distribution that on average describes an error-free circuit for any $\boldsymbol{R}$. In this subsection, I give a short proof (adapted from my collaborator Ying Li's proof in [27]) showing that Clifford circuits are indeed sufficient under the assumption that single-qubit unitary gates are noise free.

Proposition 3.1. *The training set $\mathbb{T} = \mathbb{C}$ is sufficient for finding an optimal error-mitigating gate distribution $\eta_{\text{opt}}(\boldsymbol{P})$ (if it exists) for any $\boldsymbol{R}$.*

Proof. Use $[R_j](\rho) = R_j \rho R_J^\dagger$ to denote a CPTP map of a unitary gate R_j. These maps can be written as a linear combination of single-qubit Clifford maps, as was shown in [32]. In fact, only a subset of Clifford gates are necessary. Consider a subset $\mathbb{B}_1 \subset \mathbb{C}$ of ten linearly independent single-qubit Clifford gates

$$\begin{aligned} \mathbb{B}_1 \;=\; & \{I, X, Y, Z, \\ & (I+iX)/\sqrt{2}, (I+iY)/\sqrt{2}, (I+iZ)/\sqrt{2}, \\ & (Y+Z)/\sqrt{2}, (Z+X)/\sqrt{2}, (X+Y)/\sqrt{2}\}. \end{aligned} \tag{3.11}$$

Then, an arbitrary noise-free single-qubit unitary map may be written as

$$[R] = \sum_{R' \in \mathbb{B}_1} \alpha_{R,R'} [R'], \tag{3.12}$$

where $\alpha_{R,R'}$ are some coefficients. Accordingly, any collection $[\boldsymbol{R}]$ of single-qubit unitary maps can be written as

$$[\boldsymbol{R}] = \sum_{\boldsymbol{R}' \in \mathbb{B}} \alpha_{\boldsymbol{R},\boldsymbol{R}'} [\boldsymbol{R}]', \tag{3.13}$$

where $\mathbb{B} = \{\boldsymbol{R} = (R_1, R_2, \dots) \mid R_j \in \mathbb{B}_1\}$ and $\alpha_{\boldsymbol{R},\boldsymbol{R}'} = \prod_{j=1}^{n(N+1)} \alpha_{R_j, R'_j}$.

Now, let us assume that optimal quasi-probability distribution $\eta_{opt}(\boldsymbol{P})$ exists[1] for which the loss function vanishes, i.e.

$$\text{Loss} = \sum_{\boldsymbol{R} \in \mathbb{T}} \text{Error}(\boldsymbol{R})^2 = 0. \tag{3.14}$$

[1]For noise with low severity this is a valid assumption, as shown in [27]. However, very strong noise channels, like the maximally depolarising channel, fully corrupts quantum computation and cannot be probabilistically cancelled.

Since the loss function is only defined over the training space, it is always possible to find $\eta_{opt}(\boldsymbol{P})$ during the training process.

Finally, one says that the training set $\mathbb{T}$ is informationally complete and sufficient if

$$\text{Error}(\boldsymbol{R}) = \left| E^{\text{em}}(\boldsymbol{R}, \boldsymbol{I}) - E^{\text{ef}}(\boldsymbol{R}, \boldsymbol{I}) \right| = 0 \tag{3.15}$$

for any $\boldsymbol{R} \in \mathbb{U}$ whenever using $\eta_{opt}(\boldsymbol{P})$. Consider the subset $\mathbb{B}$ as our training set $\mathbb{T}$. Then the optimal quasi-probability distribution, which is found during the training, implies that $\text{Error}(\boldsymbol{R}) = 0$ for all $\boldsymbol{R} \in \mathbb{B}$, in other words, $E^{\text{em}}(\boldsymbol{R}, \boldsymbol{I}) = E^{\text{ef}}(\boldsymbol{R}, \boldsymbol{I})$ for all $\boldsymbol{R} \in \mathbb{B}$. In turn, this implies that

$$\begin{aligned} \text{Error}(\boldsymbol{R}) &= \left| \sum_{\boldsymbol{R}' \in \mathbb{B}} \alpha_{\boldsymbol{R},\boldsymbol{R}'} \left[E^{\text{em}}(\boldsymbol{R}', \boldsymbol{I}) - E^{\text{ef}}(\boldsymbol{R}', \boldsymbol{I}) \right] \right| \\ &= 0 \end{aligned} \tag{3.16}$$

for *all* circuits $\boldsymbol{R} \in \mathbb{U}$. Here, the assumption that single-qubit gates are noise-free is used. In this case, one can replace $[\boldsymbol{R}]$ with $\sum_{\boldsymbol{R}' \in \mathbb{B}} \alpha_{\boldsymbol{R},\boldsymbol{R}'} [\boldsymbol{R}]'$ because the noise profile stays the same in both cases, and therefore, $E^{\text{em}}(\boldsymbol{R}, \boldsymbol{I}) = \sum_{\boldsymbol{R}' \in \mathbb{B}} \alpha_{\boldsymbol{R},\boldsymbol{R}'} E^{\text{em}}(\boldsymbol{R}', \boldsymbol{I})$ and similarly for $E^{\text{ef}}(\boldsymbol{R}, \boldsymbol{I})$. In conclusion, $\mathbb{T} = \mathbb{C}$ and $\mathbb{T} = \mathbb{B}$ are both informationally complete and sufficient to be used as training sets. $\square$

3.3 Practical LBQEM

As seen in the previous section, the spaces of the single-qubit unitary gates $\boldsymbol{R}$ and error-mitigating gates $\boldsymbol{P}$ increase exponentially with the circuit size. Therefore, it is impractical to compute the error for every training circuit $\boldsymbol{R} \in \mathbb{C}$ and optimise the quasi-probability distribution $\eta(\boldsymbol{P})$. In this section, I will present the significant error approach that admits a practical implementation of learning-based quantum error mitigation. First, I will describe the approach and then later demonstrate its performance with multiple numerical simulations.

3.3.1 Significant error approach

In the significant error approach, I propose to truncate the spaces of training circuits and error-mitigating gates in a way that is expected to be effective. To do so, consider the Pauli error model. General errors can be converted into Pauli errors using Pauli twirling as shown, for example, in [45]. These errors can thus be modelled as erroneous Pauli gates being randomly inserted at specific locations in the circuit with some probability. Usually, only a small subset of Pauli errors are significant, which can be corrected by the corresponding Pauli error-mitigating gates. This correction is done analogously to the correction of the depolarising noise presented above (Eq. 3.4). Recall that all depolarising and dephasing channels are Pauli channels and, therefore, are already part of the Pauli error model.

Formally, let SigE be the set of significant Pauli errors. Then one can take quasi-probabilities $\eta(\boldsymbol{\sigma})|_{\boldsymbol{\sigma}\in\mathrm{SigE}}$ as optimisation parameters and set the rest $\eta(\boldsymbol{\sigma})|_{\boldsymbol{\sigma}\notin\mathrm{SigE}}$ to zero. In other words, the number of optimisation parameters is the same as the number of significant errors. The rationale behind such a choice is the following. Since I am assuming a Pauli error model, the noise channel can be expressed as $\mathcal{N} \approx \sum_{\boldsymbol{\sigma}\in\mathrm{SigE}} p(\boldsymbol{\sigma})[\boldsymbol{\sigma}]$, where the significant errors include the trivial error (i.e. the identity operator). I take the probabilities of all other errors to be negligible. If $p(\boldsymbol{\sigma}) \ll 1$ for all non-trivial Pauli errors, then the inverse map is approximately $\mathcal{N}^{-1} \approx \sum_{\boldsymbol{\sigma}\in\mathrm{SigE}} \eta(\boldsymbol{\sigma})[\boldsymbol{\sigma}]$ for some quasi-probability distribution $\eta(\boldsymbol{\sigma})$. To see this, consider that a matrix representation of a Pauli error channel is diagonal in Pauli basis. Therefore, its inverse exists if all diagonal elements are non-zero and its structure is the same as that of the original channel. The former condition is satisfied if $p(\boldsymbol{\sigma})$ is sufficiently small for all non-trivial Pauli errors. The distribution $\eta(\boldsymbol{\sigma})$ with $\boldsymbol{\sigma} \in \mathrm{SigE}$ is therefore a rational choice for an ansatz. By choosing an appropriate construction of the set SigE, one can ensure that the size of $\eta(\boldsymbol{\sigma})$ scales polynomially with the circuit size. Similarly, only a subset $\mathbb{T} \subset \mathbb{C}$ needs to be chosen as the training set. Later, I will show some numerical evidence that the error mitigation works well for small-scale systems when the size of $\mathbb{T}$ is about three times the size of SigE.

An example construction of the set SigE, which I have used in my numerical simulations, is as follows:

1. Use cheap gate set tomography to find the naive two-qubit gate Pauli errors. Invert the error channel and, assuming the global error model for the target circuit is only composed of the combinations of these simple Pauli errors $\boldsymbol{\sigma}$, calculate the respective quasi-probabilities $\eta(\boldsymbol{\sigma})_{\text{ini}}$ for all $\boldsymbol{\sigma}$. Here by "two-qubit gate errors" I mean the error model inferred by an experimentalist purely from tomography of the two-qubit gate operating on two otherwise-isolated qubits. This task is tractable but will fail to capture the spatial (e.g. cross-talk to other qubits) and temporal correlations that will generally occur in a real noise model. The following learning procedure will then allow us to further strengthen the mitigation to encompass these more complex errors. Because the quasi-probabilities eventually used in the error mitigation are determined in the learning, a highly accurate gate set tomography for this initialisation is not required.

In my numerical simulations, I assume the gate set tomography is accurate, up to the neglected time dependence and correlations, in order to fairly compare earlier work with the proposed learning-based approach. According to this construction, the set of significant errors is $\{\boldsymbol{\sigma} | \eta(\boldsymbol{\sigma})_{\text{ini}} \neq 0\}$. This set, however, still scales exponentially with the circuit size. This first step draws parallels with the protocol introduced in the original work of probabilistic error cancellation [31, 32].

2. To restrict the set SigE to polynomial scaling, I have chosen to truncate it by leaving only errors up to a constant weight k, i.e. in any given instance $\boldsymbol{\sigma} = (P_1, P_2, ...)$ there will be non-identity errors associated with at most k of the two-qubit gates. In other words, for each $\boldsymbol{\sigma} \in \text{SigE}$, $P_i = I$ for all $i \in J(\boldsymbol{\sigma})$, where $J(\boldsymbol{\sigma})$ is some index set of two-qubit gates and $|J(\boldsymbol{\sigma})| \geq m - k$. Here, m denotes the total number of two-qubit gates.

A straightforward extension would be to encompass the initialisation and measurement error mitigation too, but for the following numerical simulations, I will purely focus on noise associated with the two-qubit gates.

So far I have shown how to construct a simple and polynomially scaling significant error set. However, one expects every experimental effort to construct a set tailored to their device's noise profile to maximise the power of the protocol. Next, I will describe the optimisation of quasi-probability distribution for the truncated SigE set.

In the significant error approach, I additionally propose to truncate the complete Clifford training set $\mathbb{C}$ used in the learning. Later I will numerically show that a randomly selected subset $\mathbb{T} \subseteq \mathbb{C}$ with a size comparable to $c|\mathrm{SigE}|$, for some overhead constant $c > 1$, is adequate to learn the optimal quasi-probability distribution $\eta_{\mathrm{opt}}(\boldsymbol{\sigma})$. After the truncations, the loss function becomes

$$\mathrm{Loss} = \frac{1}{|\mathbb{T}|} \sum_{\boldsymbol{R} \in \mathbb{T}} |E^{\mathrm{ef}}(\boldsymbol{R}, \boldsymbol{I}) - \sum_{\boldsymbol{\sigma} \in \mathrm{SigE}} \eta(\boldsymbol{\sigma}) E(\boldsymbol{R}, \boldsymbol{\sigma})|^2. \tag{3.17}$$

Since the sizes of $\mathbb{T}$ and SigE scale polynomially with the circuit size, we may efficiently evaluate $E^{\mathrm{ef}}(\boldsymbol{R}, \boldsymbol{I})$ and $E(\boldsymbol{R}, \boldsymbol{\sigma})$ $\forall \boldsymbol{R} \in \mathbb{T}$, $\forall \boldsymbol{\sigma} \in \mathrm{SigE}$ using classical and quantum hardware, respectively.

As the final task, we require to optimise the truncated quasi-probability distribution $\eta_{\mathrm{opt}}(\boldsymbol{\sigma})$ in a way that minimises the loss function. Let me expand on this by proposing the following perspective. Consider $E(\boldsymbol{R}, \boldsymbol{\sigma})$ as a matrix W with columns corresponding to distinct values of $\boldsymbol{\sigma} \in \mathrm{SigE}$ and each row corresponding to a separate training circuit $\boldsymbol{R} \in \mathbb{T}$. Similarly, $E^{\mathrm{ef}}(\boldsymbol{R}, \boldsymbol{I})$ can be taken to be a vector b where again each row corresponds to a training circuit. Then $\mathrm{Loss} = 0$ whenever we can find a vector x, describing the quasi-probability distribution $\eta(\boldsymbol{\sigma})$, such that

$$Wx = b. \tag{3.18}$$

Indeed, x is exactly the distribution $\eta_{\mathrm{opt}}(\boldsymbol{\sigma})$ that we want to find. Unfortunately, using a lot of training data implies that the matrix W usually has more rows than x does. That is, the linear system of equations is overdetermined by having more equations than unknowns, and, in turn, x will almost always not have a solution. Instead, one can use conventional mathematical tools like the method of least squares to find the value of x such that the distance $||Wx - b||$ is minimised. Here we use $||.||$ to denote the square root of the sum of squares, and this matches

Eq. 3.17 (apart from the overall square root which we have ignored). Recall that $\gamma = \sum_{\boldsymbol{\sigma}} |\eta(\boldsymbol{\sigma})|$ describes the overhead that is needed to implement error mitigation. It may be beneficial to take into account the value of γ when optimising x. For example, one could modify W by appending additional rows that work towards minimising the magnitude of individual entries of x during the overall optimisation.

Once the optimal quasi-probability distribution is obtained, error-mitigated computation with any circuit $\boldsymbol{R}$ (not necessary Clifford) is implemented using $\eta_{\text{opt}}(\boldsymbol{\sigma})$ and has the error mitigation overhead cost $\sum_{\boldsymbol{\sigma}} |\eta_{\text{opt}}(\boldsymbol{\sigma})|$.

3.3.2 Numerical results

In this subsection I present two similar but distinct demonstrations of the learning-based quantum error mitigation using the significant error approach. In both demonstrations I use exact classical simulations of quantum circuits with certain physically-motivated correlated error models. All simulations are performed using QuESTlink - a Mathematica library which integrates the framework of the Quantum Exact Simulation Toolkit (QuEST) [46, 47].

Noise-free single qubit gates

First, I present a numerical study involving spatially and temporally correlated errors under the assumption that single-qubit gates are noise free. The setup is as follows. In this study circuits are $n = 8$ qubits wide and $N = 8$ layers of two-qubit gates deep for a total of 100 gates in a pattern similar to Fig. 3.3, where all two-qubit gates are controlled-NOT gates. See Fig. 3.4(a) for an explicit circuit.

The significant-error approach is tested with two distinct correlated Pauli error models, one representing spatially and the other temporally correlated noise. All simulations of noisy quantum circuits share the same base error model - every ideal controlled-NOT gate with a control qubit i and target qubit $i + 1$ is followed by a two-qubit channel $\mathcal{D}$ acting on the same qubits i and $i + 1$. Here, depending on the simulation, $\mathcal{D}$ represents either two-qubit depolarising channel

$$\mathcal{D}_{\text{Pol}} = (1 - \epsilon)[I^{\otimes 2}] + \frac{\epsilon}{15} \sum_{\mu \in \{I,X,Y,Z\}^{\otimes 2} \setminus I^{\otimes 2}} [\mu] \tag{3.19}$$

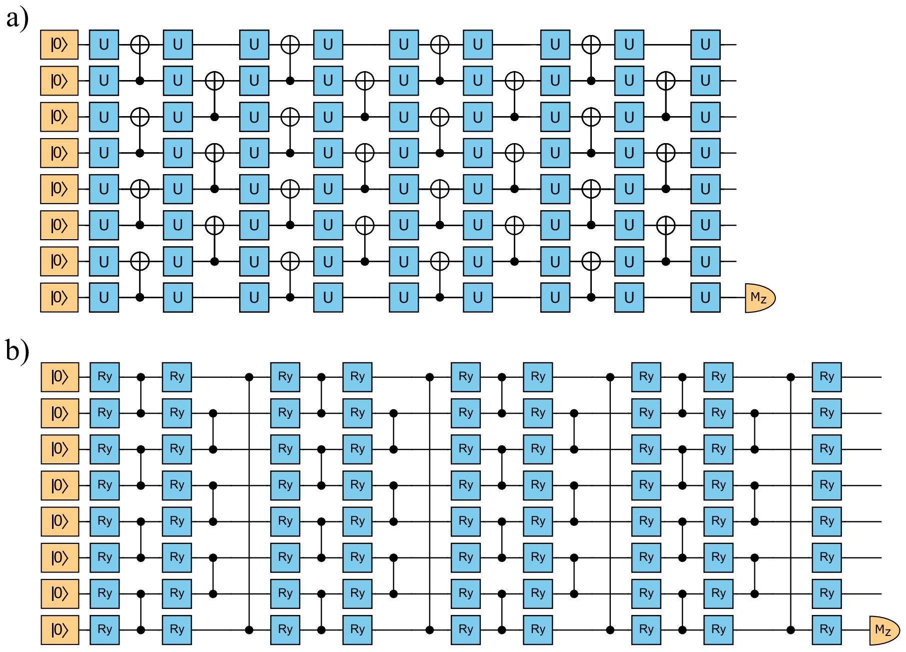


Figure 3.4: The circuits used for numerical simulations assuming (a) noise-free single-qubit unitary gates, and (b) noisy single-qubit unitary gates (represented in blue). For (a) the single-qubit gates are randomly sampled, while for (b) single-qubit gates are random rotations around the axis of Pauli-Y eigenstates. Note that in (a) two-qubit gates are controlled-NOT gates, while in (b) they are controlled-Z gates. In both circuits, the single-qubit gates are replaced by Clifford gates during the learning process.

or two-qubit dephasing channel

$$\mathcal{D}_{\text{Ph}} = (1-\epsilon)[I^{\otimes 2}] + \frac{\epsilon}{3} \sum_{\mu \in \{I,Z\}^{\otimes 2} \setminus I^{\otimes 2}} [\mu], \tag{3.20}$$

with severity $\epsilon = 0.01$. To incorporate spatially or temporally correlated errors, which are partially/fully unnoticed in the two-qubit tomography, I modify the base error model in two separate ways:

Spatial correlation. After each channel $\mathcal{D}_{\text{x}}$ on qubits i and $i+1$, I apply another *two* channels $\mathcal{D}_{\text{x}}$ with the same error rate ϵ on qubits $i+1$ and $i+2 \mod n$ and qubits $i-1 \mod n$ and i. See Fig. 3.5(a). Here x may denote the depolarising x = Pol or dephasing channel x = Ph. Note periodic boundary conditions, i.e. qubit 1 can cross-talk to qubit n. Here, the sequencing for a two-qubit gate layer is such that after each ideal gate, the three $\mathcal{D}_{\text{x}}$ noise channels are implemented before the next ideal two-qubit gate. Gates in the same layer are implemented from the top one to the bottom one.

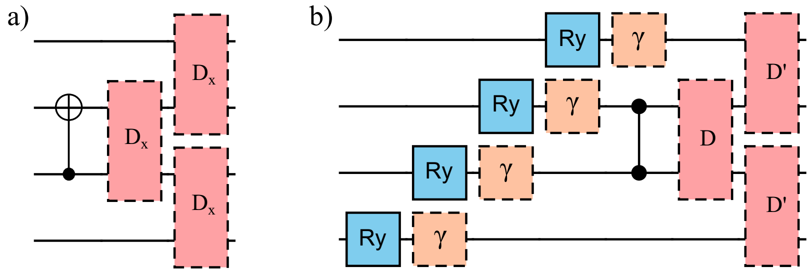


Figure 3.5: (a) Two-qubit gate error model used in the first numerical study including the spatially correlated errors. Here, $\mathcal{D}_{\mathrm{x}}$ (red) describes either a dephasing or depolarising channel. (b) The error model used in the second numerical study. Here, γ (orange) denotes the amplitude damping channel, while $\mathcal{D}$ and $\mathcal{D}'$ (red) describe a biased dephasing/depolarising channel with high and low severity respectively.

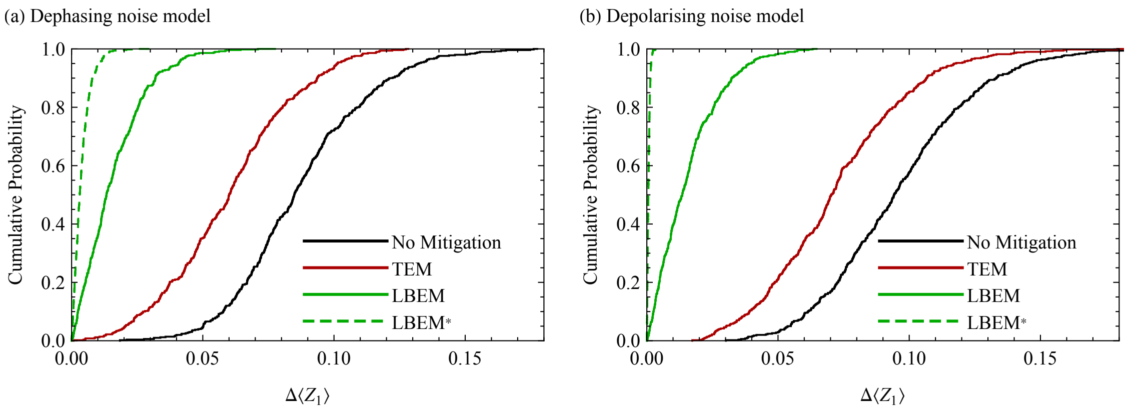


Figure 3.6: Cumulative distribution of estimated $\Delta\langle Z_1\rangle$ for 500 pseudo-random circuits with spatially correlated dephasing noise (a) and spatially correlated depolarising noise (b). Results for circuits without error mitigation (black), with tomographic error mitigation (red) and with learning-based error mitigation (green) are presented. Additionally, I include results for learning-based error mitigation when sample size $M \to \infty$ (dashed green).

Temporal correlation. Every time the circuit is run, a single qubit i, following a probability distribution Prob(i), performs much worse than other qubits. Meaning that every channel $\mathcal{D}_{\mathrm{x}}$, the qubit i is part of, has an increased error rate $\epsilon^* = g\epsilon$. In the numerical simulations I use a uniform distribution Prob(i) and set $g = 10$. Let me emphasise that this highly erroneous qubit is unknown to the experimentalist and changes every circuit run. For this part of the numerical study, only the dephasing channel $\mathcal{D}_{\mathrm{Ph}}$ is considered.

In both of these models the local noise, i.e. the noise afflicting the two qubits that are nominally involved in the gate (described by the initial error model), is assumed

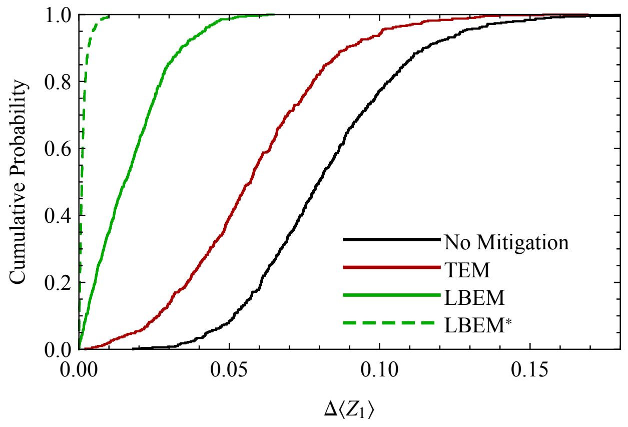


Figure 3.7: Cumulative distribution of estimated $\Delta\langle Z_1\rangle$ for 500 pseudo-random circuits with temporally correlated dephasing noise. Results for circuits without error mitigation (black), with tomographic error mitigation (red) and with learning-based error mitigation (green) are presented. Additionally, I include results for learning-based error mitigation when sample size $M \to \infty$ (dashed green).

to be fully characterised by the experimentalist - either by gate set tomography or pre-existing knowledge. No such assumption is made for the correlated part of the error model. The set of significant errors SigE is generated from the knowledge of the local noise model and truncated to the $k = 1$ error weight ($|\text{SigE}| = 85$ or 421 for the dephasing or depolarising noise model respectively). To calculate the loss in the learning process I use the deviation from the ideal expectation value of the observable Z on the first qubit. The optimal quasi-probability distribution $\eta_{\text{opt}}(\boldsymbol{\sigma})$ is found using the method of least squares as introduced in Subsection 3.3.1. The learning was performed using $|\mathbb{T}| = 3|\text{SigE}|$ filtered randomly generated Clifford circuits with the circuit layout given above. Here, I have chosen Clifford overhead constant $c = 3$ (Appendix A.1 provides some rationale for choosing the constant). What I mean by filtering is that I have discarded Clifford circuits $\boldsymbol{R}$ which do not contribute to the loss function, i.e. $E^{\text{ef}}(\boldsymbol{R}, \boldsymbol{I}) = 0$ and $E^{\text{em}}(\boldsymbol{R}, \boldsymbol{I}) = 0$, and replaced them with a different Clifford circuit. See Appendix A.2 for a further explanation.

To assess the approach, 500 pseudo-random circuits are generated that satisfy $|\langle Z_1\rangle^{\text{ef}}| > 0.3$ to represent a variety of computational tasks. The restriction with substantial $|\langle Z_1\rangle^{\text{ef}}|$ provides us with cases where noise is impactful for the following reason. For the error models and Pauli observables we have assumed, the typical effect of noise is to decrease the magnitude of expected values. Therefore, if a circuit produces an expectation value close to zero in the noise-free case, then the impact of noise will be minimal. This would result in a small performance difference between schemes that provide good mitigation and those that do not.

Each circuit is formed by drawing its single-qubit computing gates randomly from a uniform distribution of unitary square matrices of size 2. After inferring the optimal quasi-probability distribution, it is applied to all 500 circuit instances without any additional learning or re-learning. For direct comparison to earlier work, I execute each circuit $M = 10000$ times, where for each circuit I probabilistically insert $\boldsymbol{\sigma} \in$ SigE according to the distribution $|\eta_{\mathrm{opt}}(\boldsymbol{\sigma})|/\gamma$ and simply recorded a $+1$ or -1 for the observable Z_1 (inverted if the sign of $\eta(\boldsymbol{\sigma})$ is negative). In this way, I can obtain the mean value $\langle Z_1\rangle^{\mathrm{em}}$ in the same way as an experimentalist working with a real quantum computer would. Then the absolute deviation for that circuit is given as

$$\Delta\langle Z_1\rangle = |\langle Z_1\rangle^{\mathrm{em}} - \langle Z_1\rangle^{\mathrm{ef}}|.$$

The same process is repeated for alternative approaches (tomographic mitigation and no mitigation) before moving to the next of the 500 circuits. The results are displayed in Fig. 3.6, 3.7. In the figure, the label "tomography-based error mitigation" refers to the case where the experimentalist has knowledge only of the local error model and they perform error mitigation according to $q(\boldsymbol{\sigma})_{\mathrm{ini}}\ \forall \boldsymbol{\sigma} \in$ SigE truncated to error weight $k = 2$ and using the same sample size M. One may notice that I have used $k = 1$ in the learning-based error mitigation and instead $k = 2$ in the tomography-based error mitigation. Having a higher value of k truncates the significant errors at a much higher level and makes the learning process quickly intractable; while in the latter case, no such learning is performed. In fact, the usual implementation of tomography-based error mitigation takes into account all, exponentially many, errors and they are sampled probabilistically. However, since only errors of weight ≥ 3 are neglected when constructing SigE set with $k = 2$, the deviation between our output and the output from canonical tomography-based error mitigation is very small.

Noisy single-qubit gates

To further test the protocol, I apply it to a hardware-efficient variational circuit presented in Fig. 3.4(b). The circuit has 8 qubits and consists of 8 layers of two-qubit controlled-Z gates. The single qubit gates are random rotations around the axis of Pauli-Y matrix eigenstates, and just like in the last study I wish to extract the mean value of observable Z on the first qubit, $\langle Z_1 \rangle$. Qubits are assumed to be laid out in a cycle graph pattern such that a local two-qubit gate may be applied between qubits i and $i+1 \mod n$ or $i-1 \mod n$.

To this circuit I introduce an error model which more closely mimics the errors of current NISQ devices compared to the error model in the previous study. Single-qubit gates are considered to be error free, but are followed by a small probability of qubit relaxation γ. The two-qubit gates are followed by an error channel

$$\mathcal{D}(\epsilon) = (1-\epsilon)[I^{\otimes 2}] + \epsilon(\frac{\chi}{\chi+1}\mathcal{D}^*_{\text{Ph}} + \frac{1}{\chi+1}\mathcal{D}^*_{\text{Pol}}), \tag{3.21}$$

where

$$\mathcal{D}^*_{\text{Ph}} = \frac{1}{3}\sum_{\mu\in\{I,Z\}^{\otimes 2}\setminus I^{\otimes 2}} [\mu],$$

$$\mathcal{D}^*_{\text{Pol}} = \frac{1}{15}\sum_{\mu\in\{I,X,Y,Z\}^{\otimes 2}\setminus I^{\otimes 2}} [\mu].$$

Here χ describes the noise bias between the reduced dephasing $\mathcal{D}^*_{\text{Ph}}$ and depolarising channels $\mathcal{D}^*_{\text{Pol}}$ with $\eta = 0$ describing a fully depolarising channel and $\chi = \infty$ describing a fully dephasing channel. In our simulations we use $\chi = 10$, $\epsilon = 0.01$ and $\gamma = 0.001$.

Our proposed learning-based protocol for error mitigation is particularly powerful when dealing with correlated noise. To that extent, similarly to the previous numerical study, I introduce additional cross-talk errors that are often unnoticed in local tomographic noise characterisation processes. These cross-talk errors are modelled by a much weaker noise channel $\mathcal{D}' = \mathcal{D}(\frac{\epsilon}{10})$ happening after the two-qubit noise channel $\mathcal{D}$. Like in the previous demonstration of spatial correlations, channel $\mathcal{D}'$ is applied between each qubit that is nominally involved in the two-qubit gate and a nearby qubit not involved in the gate. See Fig. 3.5(b) to see the full error

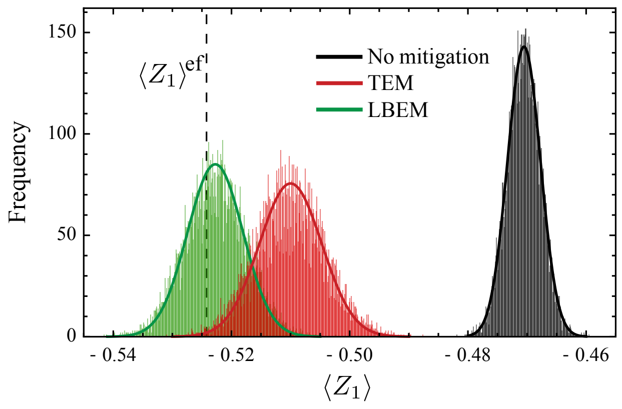


Figure 3.8: Expectation values of $\langle Z_1\rangle$ obtained from a single experiment repeated 10000 times with no error mitigation (black), tomography-based error mitigation (red) and learning-based error mitigation (green). Dashed line indicates the error-free expectation value $\langle Z_1\rangle^{\text{ef}}$. Solid lines describe the analytically derived probability distributions for each approach. $M = 10^6$ samples were taken for each experiment.

model after each controlled-Z gate. For completeness, I also show the relaxation errors that are introduced after every single qubit gate.

With this error model, I generate a *single* 8 qubit noisy circuit that satisfies $|\langle Z_1\rangle^{\text{ef}}| > 0.5$ to better observe the effect of the proposed error mitigation protocol. The optimal quasi-probability distribution $\eta(\boldsymbol{\sigma})_{\text{opt}}$ is learned by considering the significant error set SigE truncated to $k = 1$ error weight and using Clifford overhead constant $c = 3$. This set only consists of two-qubit Pauli errors following each controlled-Z gate. Using the learned optimal distribution, I compare 10000 error mitigated expectation values to the ideal expectation value $\langle Z_1\rangle^{\text{ef}}$, see Fig. 3.8. Each expectation value is obtained by sampling $M = 10^6$ shots.

For comparison, I include non-mitigated expectation values and expectation values obtained using tomography-based error mitigation. As in the previous numerical study, tomography-based error mitigation refers to the case where the experimentalist has knowledge only of the local error model and they sample according to $q(\boldsymbol{\sigma})_{\text{ini}}\ \forall\boldsymbol{\sigma} \in \text{SigE}$, where SigE is truncated to $k = 2$ error weight. Due to the learning set being truncated and due to the error model involving non-Pauli error processes, like the relaxation gates, the learning-based protocol does not perfectly mitigate the error but has substantial improvement compared to the tomography-based error mitigation. Also, notice that the variance of the learning-based approach is less than the variance of the tomography-based approach while achieving expectation values closer to the ideal value. The explanation of this is as follows. During the learning process, the proposed algorithm finds which gate errors do and do not affect the computational result. In our circuits, for example, some of the later two-qubit gate errors do not affect the expectation values of the first qubit. In turn, these errors are not corrected to reduce the total error mitigation overhead and, hence, the variance. This is another feature of learning-based quantum error mitigation protocol.

3.4 Discussion

In this chapter, I have presented a novel and intuitive way of mitigating errors in quantum computation. It is based on the probabilistic error cancellation technique to which we introduce a new learning component. Given this component and some additional well-motivated assumptions, there is no need to reconstruct the noise model in the experiment. The learning process exploits the efficient simulatability of Clifford circuits to learn the optimal quasi-probability distribution that is required for probabilistic error cancellation. In turn, this distribution allows us to mitigate noise not only for Clifford circuits but for any circuit constructed in a specific but universal way.

Under various numerical studies of correlated noise, I have shown that the learning-based quantum error mitigation can be practically implemented on circuit sizes that are comparable to the circuits currently run on quantum platforms. All of these studies indicate that, in the presence of correlated noise, our protocol greatly outperforms the canonical tomography-based approach in which the learning part is omitted.

As proposed and briefly explored in [27], the learning subroutine might be carried out using different tactics than the ones I have introduced in this chapter. If one wants to use quantum algorithms where multiple observables need to be evaluated, then the learning process needs to reflect this. For example, finding the optimal quasi-probability distribution that maximises the overall output fidelity can be advantageous as it will work for any observable. Finally, on platforms where it may be difficult to alter the circuit every run, mean values of the most likely sampled circuits can be estimated and used for the computational result.

Possible extensions to this work include modifying the learning component of the protocol in cases where some information about the noise model is given or easily accessible, for example, in a scenario where the whole circuit undergoes an unknown global phase shift. Another extension would be to sample Clifford circuits respective to the unitary circuits they replace. While this is considered as a circuit-specific error mitigation, one would expect that the learning part is less expensive in this case.

All of the error-mitigating strategies that I have explored in this chapter can be combined, adjusted and tailored to the quantum system at hand. A comprehensive recent summary of quantum error mitigation techniques can be found in [48]. Even when QEM has unfavourable scaling with the system size, it may find its uses in large-scale fault-tolerant quantum computation during particular subroutines. For example, in [49] the authors propose to use probabilistic error cancellation to mitigate the noise in encoded T gates. This opens a new paradigm to use quantum error mitigation and error correction in tandem, which I believe might be an effective strategy before fault-tolerance has been fully reached.

On that note, the rest of the chapters in this thesis will describe my contributions towards practical quantum error correction. While error mitigation might find its uses in near-term computation or niche routines, it is error correction and fault-tolerance that will make quantum computing truly scalable.

The following chapter is based on two articles authored by me. The first article, to which most of the chapter is dedicated, is titled "*Quantum computing is scalable on a planar array of qubits with fabrication defects*" [50]. I was the main author of this work with co-authors Simon C. Benjamin and Benjamin J. Brown. The contributions are as follows. I and BJB identified the problem by me surveying the literature. The *shell* methodology was devised by both of us which I further adapted for a later proof. The main part of the paper - the threshold theorem - was fully proven by myself. All three of us participated in writing the paper and devising minor ideas that are found within. SCB and BJB supervised the project throughout.

The second article is presented in the last few pages of the chapter and provides evidence of the practicality of the methods found in the first article. It is titled "*Adaptive surface code for quantum error correction in the presence of temporary or permanent defects*" [51]. The main author of this work was Adam Siegel with co-authors Simon C. Benjamin, Thomas Flatters and myself. The contributions are as follows. AS carried out the numerics (with input from TF) and devised a new decoder to implement the shell approach introduced in the previous theory paper. AS, SCB and I devised the formalism for comparing different methodologies. The paper was mostly written by AS with edits from SCB and me. Both I and SCB supervised the project throughout.

Chapter 4

Quantum computing on a defective lattice

In the last chapter, I gave a short introduction to quantum error mitigation and had a closer look at some of its variants. They constituted an algorithmic way to reduce the effect of noise in quantum computation. However, given their cost scales exponentially with the system size, they may be only applicable in small to medium-scale computations.

In this and the following chapter, I will shift the focus to quantum error correction (QEC) - a much more mathematically rigorous yet powerful way of algorithmically dealing with noise and general errors in quantum computation. As explained in Chapter 2, a QEC code uses a larger, redundant space of qubits to detect and correct errors on the encoded information. The exact overhead to perform QEC depends on the chosen code, however, even simple approaches scale only logarithmically with the system size. Therefore, quantum error correction paves a path towards truly scalable and fault-tolerant quantum computation.

In this chapter, I will first introduce a prominent and practical CSS stabiliser code called the surface code. Then, as per the theme of practical quantum computation, I will provide rigorous evidence that this code does not fail whenever some parts of the quantum processor are deemed inoperable or faulty. The evidence consists of both analytical proofs and numerical studies.

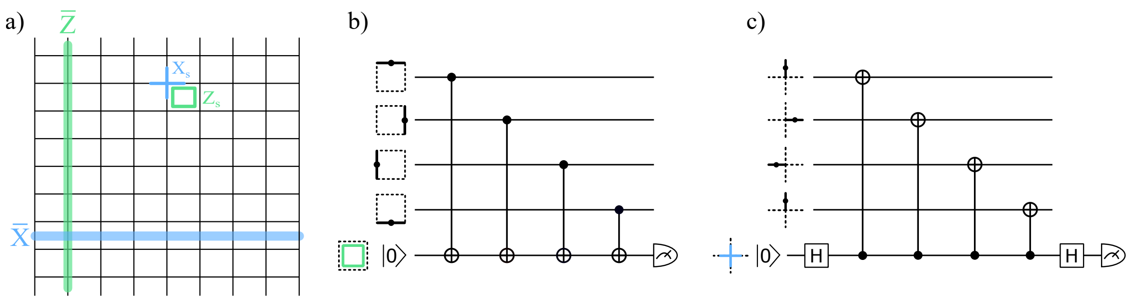


Figure 4.1: (a) A lattice representing the surface code. The data qubits are placed on edges and X(Z) stabilisers are defined on vertices(faces). As an example, I highlight a single X and Z stabiliser and their associated data qubits. Physical errors are detected by measuring the eigenvalues of stabiliser generators. This is commonly done with an ancilla qubit (bottom) using a circuit (b) for Z stabilisers and a circuit (c) for X stabilisers.

4.1 Introduction to surface codes

An approach towards fault-tolerant quantum computation adopted by multiple research teams across various platforms is to fabricate a two-dimensional device with a large array of interacting qubits realised on its surface [13, 52, 53]. A natural code that fits such an architecture is the surface code which has a planar layout, and therefore, does not need the generation of long-range entanglement. Such entanglement is often hard to reliably produce, has a high error rate or requires additional costs [54, 55]. Moreover, the surface code has one of the highest physical error rate thresholds, and it is clear how to efficiently perform universal quantum logic using this code [26, 56]. I will explain what I mean by error rate threshold in Subsection 4.1.2. One could say that the surface code is one of the most experimentally friendly solutions to dealing with errors in quantum computation. Indeed, there has been intensive experimental development and testing of small instances of the surface code, for some examples see [3, 22, 57–59].

4.1.1 Basics

Let me start by briefly introducing the basics of surface codes. For a more thorough introduction, I guide the reader to A. Fowler's review on surface code quantum computation [26].

The easiest way to analyse the surface code is by visualising it as a two-dimensional square lattice, Fig. 4.1(a). The lattice has rough boundaries on the top and bottom and smooth boundaries on the left and right. The data qubits are placed on every edge and the stabiliser group is generated by the stabiliser generators that are defined on every individual vertex (X stabilisers) and every individual face (Z stabilisers) of the lattice. They are formally written as $X_v = \prod_{\partial e \ni v} X_e$ for each vertex v and $Z_w = \prod_{e \in \partial w} Z_e$ for each face w, where X_e and Z_e denote the standard Pauli matrices acting on the data qubit on edge e, ∂e denote the set of vertices v on the boundary of edge e and ∂w denote the set of edges e on the boundary of face w. As one can see, both types of operators have four qubits in their support unless the face/vertex is part of a lattice boundary, in which case the support consists of three qubits only. The requirement that stabilisers need to commute is satisfied as any pair of X and Z operators share an even parity of data qubits. Note that in the following I will often use the term “stabiliser” as a shortened version of “stabiliser generator”.

Recall, that the number of encoded logical qubits is $k = n - m$, where m is the number of independent stabilisers. For the surface code of side length L, there are $n = 2(L^2 - L) + 1$ data qubits and $m = 2(L^2 - L)$ independent stabilisers, which are simply the vertex/face generators defined above. Hence, the surface code encodes $k = 1$ logical qubit. The representative logical operators of this code can be seen in Fig. 4.1(a) as blue ($\overline{X}$ operator) or green ($\overline{Z}$ operator) lines spanning the lattice. The data qubits in the non-trivial support, i.e. the weight, of these operators are edges under these lines, and it is easy to verify that the weight of both representatives is L. Any other representative logical operator will have equal or higher weight. Recall that the least weight of any logical operator sets the code distance $d = L$ and therefore, the surface code can reliably correct up to $\left\lfloor \frac{L-1}{2} \right\rfloor$ errors.

On most quantum platforms, both X and Z stabiliser generator values are extracted by measuring an ancilla qubit using the circuits given in Fig. 4.1(b) and (c) respectively. Each ancilla qubit is linked to at most four data qubits using CNOT gates, after which, it is measured out in the computational basis. Such

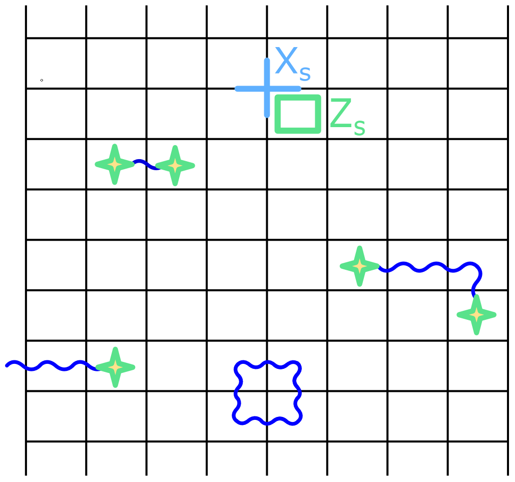


Figure 4.2: An example spatial error configuration during a single measurement round. In this representation, the data qubits, whose corresponding edges cross the error strings (blue), have undergone a bit-flip (X) error. Local Z stabilisers (faces) produce detection events (stars) if an odd number of qubits in their support have an X error. Therefore, only the ends of errors strings are detected.

measurements are repeated over time, and therefore, form a (2+1)-dimensional space-time lattice where the time axis runs in a direction orthogonal to the planar qubit array that supports the surface code. Two chapters ago, I defined error detection as observing a negative eigenvalue outcome of any Pauli stabiliser measurement. Here, I will generalise this notion and say that an error is detected when the outcome of any given stabiliser differs over two consecutive measurements. Therefore, the locations of the error detection events (events in short) are defined in both space and time. The configuration of all the detection events on the space-time lattice is called the error syndrome.

Errors on the surface code can be regarded as strings in the space-time lattice that produce detection events at their endpoints. Bit-flip (X) and phase-flip (Z) errors that occur on the data qubits produce error strings that lie parallel to the spatial plane of space-time lattice, for example, see Fig. 4.2. While stabiliser measurement errors, i.e. errors that cause a stabiliser measurement to return the incorrect outcome, are error strings that point in the temporal direction of the lattice. In general, any type of individual errors can compound to make longer strings. To reliably correct both data and measurement errors, the stabiliser measurements are usually repeated $\approx d$ times forming a size $d \times d \times d$ space-time lattice. Using a decoder, error detection events of the collected data are matched together to look for an error correction that, together with the original error, acts trivially on the encoded information.

As one can imagine, running the circuits that are used to extract the syndrome data might cause additional errors compared to the case where no error correction is applied. This higher error rate contributes to logical errors, and for any code to be useful, the logical error rate should be at least smaller than the physical error rate. Otherwise, one should not encode their physical qubits at all. Furthermore, one may want to suppress the logical error rate to an arbitrary level. This requires using codes with greater distances and, therefore, higher protection against errors. Considering stabiliser codes from the same family, e.g. the surface codes family, a larger code instance not only corrects but also introduces a higher number of errors - simply because there are more qubits and more syndrome extraction circuits. The turning point at which it becomes beneficial to use the larger code is described within the framework of the threshold theorem.

4.1.2 Threshold theorem

One may assume that if operating the code, for example, running the syndrome extraction, introduces more errors on average than the number of errors the code can reliably correct, then the code is overall useless. However, that is not necessarily the case. Given some error model, many configurations of physical errors may not contribute to the logical failure rate, for example, a loop of errors at the bottom of Fig. 4.2 acts trivially on any codeword. Such errors are equivalent to an action of a stabiliser. Keeping that in mind, the logical error rate is expressed as

$$\overline{P} = \sum_{E \in \mathcal{F}} p(E), \tag{4.1}$$

where $\mathcal{F}$ denotes the set of physical errors that lead to a logical failure, and $p(E)$ denotes the probability that the error E is drawn from some error model. Then, for a given code family, the *threshold theorem* states that $\overline{P}$ exponentially vanishes as the code distance grows for a suitably low but constant error rate associated with each physical qubit and its operations. The maximal value of the error rate below which $\overline{P}$ is suppressed is called the *threshold*. If such a non-zero value does not exist, then we say that the threshold does not exist. Ultimately, the nonexistence of the threshold limits the scalability of the proposed code family for fault-tolerant quantum computation.

Finding whether the threshold exists is not trivial, however, some indications are useful. If a code family has stabiliser supports that grow with the code size, then in most cases the threshold does not exist. In this case, larger syndrome extraction circuits are required for larger codes and the additional errors introduced accumulate faster than one is able to correct them. As an example, Bacon-Shor code [60] does not have a threshold, while the aforementioned surface code does.

In the rest of this subsection, I will summarise a threshold existence proof for the surface code which was first shown by Dennis *et al.* [61]. In this proof, the main part is to upper-bound the logical failure probability $\overline{P}$ by characterising the set of physical errors $\mathcal{F}$. The proof presumes a phenomenological error model which, as I described before, simplifies the more rigorous circuit-based error model to two kinds of errors - X/Z data qubit errors which both happen with probability ϵ every measurement round and ancilla measurement errors which flip the measured stabiliser value with probability q. For simplicity, I will set $q = \epsilon$ and call ϵ the physical error rate. Finally, the proof also requires a decoder that can find the least weight correction. This means that the decoder infers the shortest error configuration that matches the syndrome as the most likely error. If there are multiple shortest configurations, the decoder can output any of them. For example, the minimum-weight perfect-matching algorithm [19, 26] is an efficient decoder that satisfies the criteria.

The decoder fails (and the computation results in a logical error) whenever its correction error chain E_{min}, together with the actual error chain E, traverse some non-trivial path $\boldsymbol{\ell}$ on the space-time lattice such that $E + E_{min}$ is a logical operator. Since both $\overline{X}$ and $\overline{Z}$ operators have weight at least L, the path has some minimal length $\ell \geq L$. It is argued in [61] the weight of the error $|E|$ (which in our case denotes the number of data qubit or measurement errors that have happened in the space-time lattice) must be at least $\ell/2$, where I use ℓ to denote the weight of a path $\boldsymbol{\ell}$. Otherwise, the decoder will either successfully correct the error or create a new error which acts trivially on the codespace. With this characterisation, we can regard errors $E \in \mathcal{F}$ as error configurations lying along non-trivial paths $\boldsymbol{\ell}$. This enables us to upper bound $\overline{P}$ as

$$\overline{P} \leq \sum_{\ell \geq L} N(\ell)\Pi_\ell, \tag{4.2}$$

where we sum over all non-trivial paths of length at least L and $N(\ell)$ denotes the number of error configurations that lead the decoder to give rise to a non-trivial path of length ℓ. The associated weighting Π_ℓ is bounded by the worst case probability that an error of weight at least $\ell/2$ occurs

$$\Pi_\ell \leq \epsilon^{\ell/2}. \tag{4.3}$$

If the error weight is $< \ell/2$, then $E + E_{min}$ does not contain any path of length ℓ due to our assumption that the decoder infers the correction E_{min} as the shortest error configuration that matches the syndrome. That is, the inferred error cannot be larger than the actual error.

In the original work of Dennis *et al.* [61], the authors upper bound $N(\ell)$ by recognising that the number of non-trivial paths $\boldsymbol{\ell}$ is upper bounded by the number of self-avoiding random walks with the same length $K(\ell)$. Furthermore, we can bound $K(\ell) \leq V\nu^\ell$, where $\nu + 1$ is the valency of the lattice and V is the number of starting points for the walk (see Fig. 4.3 for a visual explanation). The number V is upper bounded by the volume of the space-time lattice which is a polynomial of the code distance d. Therefore, the error configurations $N(\ell)$ can be bounded as $N(\ell) \leq 2^\ell V \nu^\ell$ where 2^ℓ upper bounds the number of ways to distribute $\ell/2$ or more errors along a path of length ℓ. Finally, the minimum length of any non-trivial path is determined by the code distance $\ell \geq L = d$. Putting everything together we obtain

$$\overline{P} \leq V \sum_{\ell \geq d} 2^\ell \nu^\ell \epsilon^{\ell/2} = V \sum_{\ell \geq d} (2\nu\epsilon^{1/2})^\ell. \tag{4.4}$$

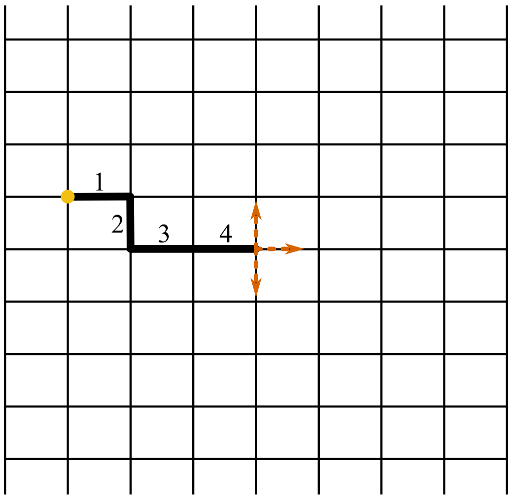


Figure 4.3: A schematic of the path counting argument in [61]. The number of non-trivial paths are upper bounded by the number of walks of length $\geq L$. Each walk has a starting point and takes a step in one out of ν directions. The valency for this lattice is four, however, one can obtain a better estimate of self-avoiding walks by discounting those walks that backtrack previous step. Therefore, we take $\nu = 3$ for all steps apart from the initial step. Here, I show an example walk that so far consists of four steps. The yellow point indicates the starting point of the walk, and the fifth step can be made in the three directions highlighted in orange.

If the physical error rate ϵ is low enough such that the argument $2\nu\epsilon^{1/2} \leq 1$, then increasing the code distance d exponentially suppresses the upper bound of the logical error rate $\overline{P}$, and hence, the threshold exists! However, the turning point ϵ_0 at which $2\nu\epsilon_0^{1/2} = 1$ is not necessary the exact threshold value and only provides its lower bound. Using $\nu = 5$ one obtains $\epsilon_0 \approx 0.7\%$, while the actual threshold for the phenomenological error model is about 3% [62]. The exact value is greatly underestimated due to multiple pessimistic simplifications that were made during the proof, for example, not every long walk of errors necessarily causes a logical error. Nevertheless, following this proof technique, one can ascertain the existence of threshold in general.

4.2 Fabrication defects

Having a threshold is one of the requirements for the code family to be used as a scalable approach for arbitrarily large and deep quantum computations. To successfully implement a higher amount of logical operations, a code with greater distance is employed. For example, a version of Shor's algorithm requires around 10^9 logical Toffoli and T gates to factor a 2048 bit integer [1]. The authors of [1] estimate that for a physical gate error rate of 10^{-3} a surface code with distance $d = 27$ is needed to perform this computation with 27% success chance. Using the canonical surface code layout, the code requires 1405 operational data qubits per logical qubit. Research efforts across various platforms plan to fabricate a two-dimensional chip that can host many if not all such logical qubits. Given the sheer amount of components that need to be fabricated, they are likely to find that some non-zero fraction of them will be highly imperfect, or will not function at all. Moreover, depending on the platform, it may be difficult to keep a large unbroken array of surface code patches over large scales. Instead, it may be necessary to introduce "pathways" or "holes" where classical electronics, cooling infrastructure or other operational elements are passing through. In the rest of the chapter, I will refer to all of the above as fabrication defects.

Even the ability to identify the locations of defects offline, which I will assume for most of the chapter, is not enough to negate the harmful effects caused by them. Effectively, fabrication defects give rise to punctures on the lattice where the standard stabiliser generators of the ideal code cannot be measured. Nevertheless, it is still important to identify the endpoints of the canonical string-like errors at the locations of these punctures. One can imagine that if an ancilla qubit in the middle of the surface code lattice were found to be a defect, there would be undetectable error strings of length approximately $L/2$. These would constitute an unwanted logical operation which implies that the code distance has been halved. One, therefore, needs to use alternative strategies to determine the locations of detection events in areas of the lattice where fabrication defects exist. In this section, I will describe two methods on how to tackle fabrication defects using a surface code architecture. While they may seem similar at first, the latter protocol will allow me to rigorously prove that the surface code does indeed preserve the physical error rate threshold under a finite rate of fabrication defects and, therefore, is a scalable solution to fabrication defects.

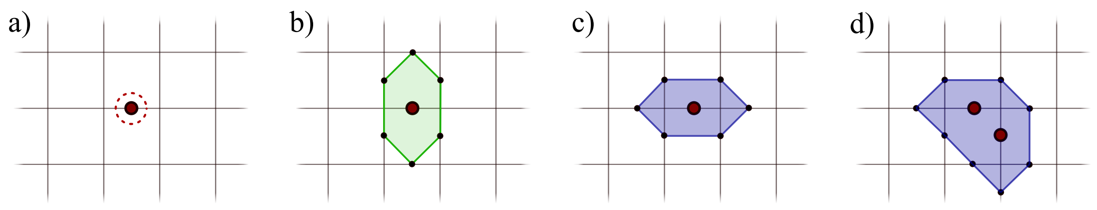


Figure 4.4: (a) A defective data qubit is identified in red. To isolate it from the rest of the code, the defective qubit is removed from the support of all stabilisers. (b) Around the puncture, a Z super-stabiliser is formed using the intact elements of the array. It has an increased weight (in this case 6), and it commutes with all other stabilisers as required. (c) Similarly, an X super-stabiliser is formed. (d) If multiple defects are close to each other, then bigger super-stabilisers are needed. Here, only the X super-stabiliser is shown.

4.2.1 Stace's method

The first protocol makes use of super-stabilisers that enclose the fabrication defects to negate their harmful effect. The idea to quarantine defects in such a manner was first proposed by Stace *et al.* [63] and later improved in [64, 65]. Hence, let me call this approach Stace's method, whose details I now summarise.

Once a fabrication defect is identified within the qubit array, a puncture P of the code is created that fully encloses it by turning off the respective elements of the array. This may manifest as disabling nearby data or ancilla qubits and the respective CNOT links that connect them. Around each puncture P, two high-weight stabiliser operators are created - one Z super-stabiliser and one X super-stabiliser. Fig. 4.4 shows an example of this process. While directly measuring the super-stabilisers would be enough to infer the ends of error strings in the proximity of fabrication defects, this task is difficult due to their syndrome extraction requiring circuitry that was not initially fabricated. Instead, [63–65] propose a different method to measure super-stabilisers. They show that these complicated measurements can be decomposed into several low-weight non-commuting measurements to read out the stabilisers in a practical way.

The value of the X super-stabiliser $\mathcal{A}_P$ around a puncture P is inferred by measuring the set of vertex operators X_w for $w \in \partial P$ and taking their overall product since $\mathcal{A}_P = \prod_{w \in \partial P} X_w$. Similarly, the Z super-stabiliser can be inferred by taking the product of the respective face operators around the puncture. See Fig. 4.5.

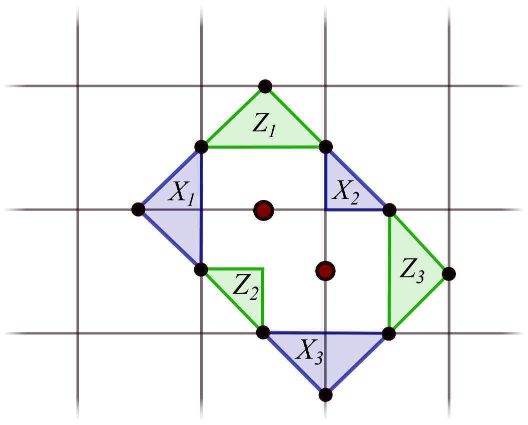


Figure 4.5: Each super-stabiliser is broken down into weight-3 and weight-2 gauge operators around the puncture. In this example, I consider the defect configuration displayed in Fig. 4.4(d). Taking the product of all respective gauge operator values gives the value of the super-stabiliser.

Now, the low-weight face/vertex gauge operators do not commute and, therefore, are not part of the persistent stabiliser group and cannot be measured simultaneously. Instead, both subsets of operators are gauge operators and must be measured in an alternating fashion. Note that, all of these gauge measurements commute with the rest of the stabilisers on the lattice and, therefore, either subset can be measured simultaneously with other stabilisers of the code. Likewise, super-stabilisers $\mathcal{A}_P$ and $\mathcal{B}_P$ both commute with all of the gauge operators. Therefore, measuring one subset of gauge operators followed by the other, and so on, allows us to measure the value of one super-stabiliser without disturbing the value of the other. This sequence of measurements acts like a code deformation where the boundary of a puncture is transformed from rough to smooth, and vice versa.

Following the above procedure of measuring the gauge operators in an alternating fashion and finding the value of super-stabilisers works well for small code sizes. In fact, numerical results have shown that the logical failure rate can be suppressed exponentially by increasing the code distance [63–65]. However, it remained to be shown whether these solutions *scale* to give an arbitrarily low logical failure rate in the presence of a finite density of fabrication defects. It is commented that some of these results may only demonstrate a pseudo-threshold [64] - a turning point in the physical error rate below which a specific size code outperforms a smaller code from the same family (or no encoding at all). Unlike a code threshold, the pseudo-threshold may tend to zero in the asymptotic limit. The observation that only pseudo-threshold exists is consistent with recent work that argues that a two-dimensional code with a high density of static punctures, arranged in a fractal pattern, will not demonstrate a finite error rate threshold [66].

The reason why Stace's method may only exhibit a pseudo-threshold is as follows. As one increases the code size, it is highly likely that some of the fabrication defects by chance occur near one another, forming a cluster and requiring a larger puncture. Respectively, large super-stabilisers are created that consist of many individual gauge operators. While each of the gauge operators has a finite and small chance of being erroneous, the chance that the product of them gives us the wrong value increases with the number of operators in the product. Therefore, for large enough codes (which will have large fabrication defect clusters), the probability that either of the super-stabilisers $\mathcal{A}_P$ or $\mathcal{B}_P$ around a large puncture contains a measurement error approaches 50%. At that point, one may not reliably infer whether an error string has terminated at the puncture and, therefore, any error string between two such punctures is undetected. An undetected error string is equivalent to a logical operator and, hence, the separation of two large clusters limits the code distance that one can achieve by scaling the system size.

The above argues (but doesn't prove) that the threshold does not exist for Stace's method of dealing with fabrication defects on a surface code. In the rest of this section, I will introduce a slightly different dynamical protocol of "shells" that allows one to reliably learn the values of high-weight stabilisers presented here. Later in the chapter, I will rigorously prove that the shell method recovers a non-zero threshold for a finite density of fabrication defects.

4.2.2 Shells

Before introducing the protocol of shells, let me first describe some simplifications that I will use in the rest of the chapter. Consider a planar array of qubits

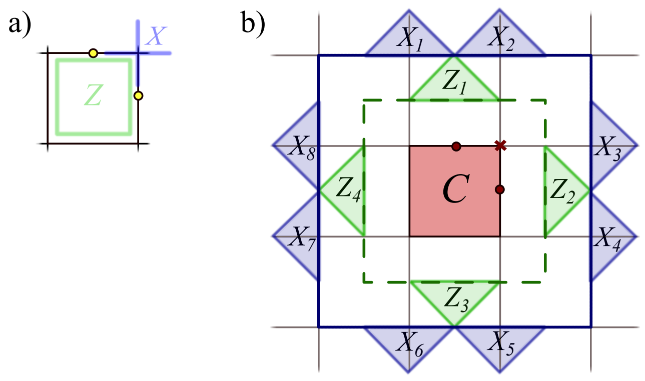


Figure 4.6: (a) A unit cell consists of a Z stabiliser in the centre, an X stabiliser on the top right, and two qubits located above and on the right of the cell. This ensures that the probability of one unit cell being faulty is independent of any other unit cell being faulty. (b) Code deformations around a faulty unit cell C during the Z-type gauge measurements (green) and X-type gauge measurements (blue) resulting in a square super-stabiliser. Here, only weight-three gauge operators are displayed, however, to ensure commutativity, the Z super-stabiliser also consists of the corner weight-four stabilisers.

constituting a surface code as described in the previous section. One can course-grain the array in terms of unit cells, see Fig. 4.6(a). The unit cell consists of four elements; two data qubits, one vertex operator, and one face operator. This enables us to define a simplified and homogeneous model of fabrication defects - we call a unit cell a fabrication defect unless all of its components are intact. That is, its qubits, the circuitry and ancilla qubits used to measure stabilisers all function correctly. However if even one of the elements is not intact then, for simplicity and clarity, we will assume that the entire unit cell is faulty. Similar to Stace's method, we also assume that we can use the circuitry of intact unit cells to measure super-stabilisers around the fabrication defect. Specifically, we will make weight-three and weight-four measurements around each fabrication defect forming square super-stabilisers (see Fig. 4.6(b) where weight-three gauge operators are highlighted).

Stace's method describes two sets of operations that effectively deform a boundary around a puncture from rough to smooth and vice versa. This respectively corresponds to measuring a Z super-stabiliser (rough boundary) or an X super-stabiliser (smooth boundary). As I discussed, a finite rate of gauge measurement errors can lead to high-error rates of unavoidably large super-stabilisers. Therefore,

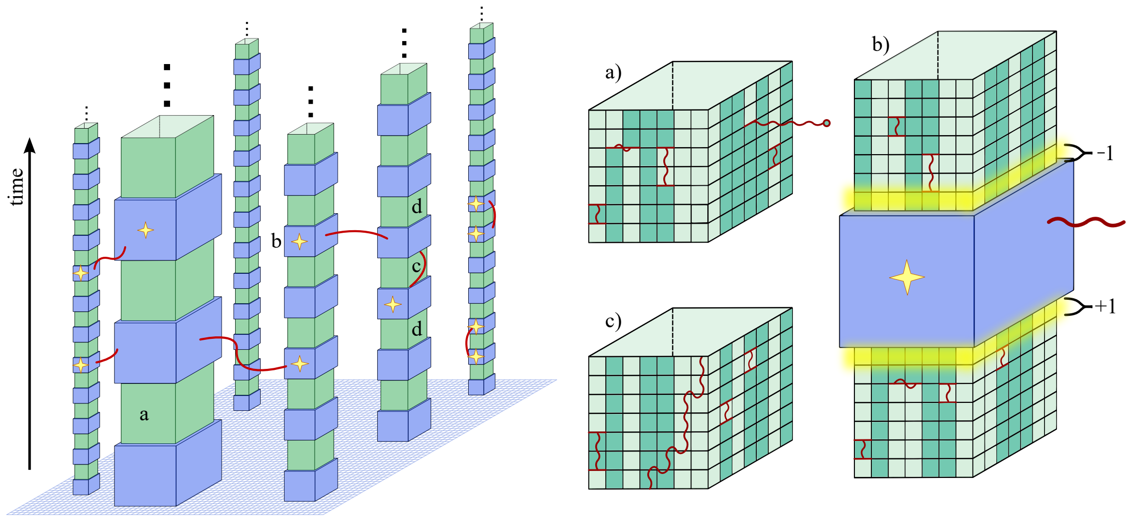


Figure 4.7: Space-time lattice with super-stabilisers forming shells around the fabrication defect clusters. The super-stabilisers are repeatedly measured around each puncture P forming columns that align in the temporal direction. The measurements of super-stabilisers $\mathcal{A}_P$ and $\mathcal{B}_P$ are depicted in blue and green colours respectively. The collection of consecutive measurements of the same super-stabiliser forms a shell. The height of the shell is proportional to the size of the puncture that fully encloses the fabrication defect cluster. The red strings show a specific error configuration, with stars indicating the shells that detect an event at the end of some error string. The right panels describe common circumstances; (a) a typical shell with a few gauge-operator measurement errors and a bit-flip error string (with light green and dark green squares representing $+1$ and -1 gauge-operator outcomes respectively), (b) the value of super-stabilisers before and after an X error string has terminated at the smooth shell between them and (c) a collection of errors that results in a temporal error string across the shell; these errors are detected by comparing the values of the previous and next shell of the same boundary ((d) in the left panel).

a new strategy is required to learn the values of super-stabilisers and maintain the surface code threshold.

To reliably infer the super-stabiliser values I propose to temporarily *fix the gauge* by repeating the measurements of one set of gauge operators for multiple rounds before switching to the other set and doing the same. Specifically, given a puncture of linear size r, I propose to measure the gauge operators of a given type over $O(r)$ rounds before transforming the boundary into its opposite type and measuring the other type of operators. This method gives rise to objects in the space-time lattice that we call shells. To visualise this, consider a space-time picture of the lattice given in Fig. 4.7, where the punctures of the qubit array are projected along columns. In this figure, the columns are shown in alternating colours, where the green blocks indicate periods where one measures the Z super-stabiliser (rough boundary) and blue blocks, X super-stabiliser (smooth boundary). A single block is called a *shell*. The height of each shell is proportional to its width since we have chosen to measure the gauge operators of one type for a number of rounds proportional to the width of the puncture before switching to the other type.

Let me now discuss how errors are detected using the protocol of shells. Strings of X errors can terminate at smooth boundaries (blue shells in Fig. 4.7) without immediate detection. To find out whether an X error string has terminated at a blue shell around puncture P, we compare the product of the next Z gauge operator measurements $v \in \partial P$ to the product of Z gauge operator measurements before initialising the smooth boundary; see Fig. 4.7(b). This is equivalent to comparing super-stabiliser values over time, and the value of the product should not change if no error string has terminated at the shell. However, the parity of these measurements will change if any X error string terminates at the given blue shell, or if any of the face measurements experience a measurement error. Due to this possibility of measurement errors of individual gauge operators, it is crucial to repeat the measurements of the same gauge operators. The repetition fixes the gauge and provides an indication of where such measurement errors might have happened. In the case that we find odd parity, we mark a detection event at the shell that can be paired with another event. Likewise, Z error strings can terminate on rough boundaries, which correspond to green shells, and the detection of Z error strings is analogous to X error strings. In Fig. 4.7, I show an error configuration (red strings) of X errors on data qubits (horizontal direction) and Z gauge operator measurement errors (vertical direction) that create multiple detection events on blue shells.

One may observe that we cannot necessarily measure the value of a shell when the system is initialised. For instance, the first time we deform a smooth boundary to a rough boundary, we cannot compare the value of $\mathcal{B}_P$ to an earlier measurement of the same operator, because we have not initialised the system into an eigenstate of $\mathcal{B}_P$ yet. At these locations, the shell is regarded as an extension of the temporal boundary. Similarly, if the fabrication defects are too close to one of the physical boundaries of the qubit array, one might not have enough space to fully enclose them in a shell. In this case, the collection of defects is regarded as the extension of the spatial boundary by walling them off with the respective boundary. This strategy for handling fabrication errors near boundaries was proposed in [64], where its details can be found.

The repetition of gauge measurements of the same type is the key feature of the protocol; it enables us to reliably identify the locations of all types of errors near fabrication defects irrespective of how large the defects become. Importantly, assuming that large punctures on the qubit array are well separated, then the repetition means that two shells of the same type are also well separated in the temporal direction of the space-time lattice. Compared to Stace's method, we have effectively homogenised space and time from the perspective of super-stabilisers. In the next section, I will show that the method of shells allows one to obtain, for the first time, a threshold theorem for a two-dimensional quantum error correcting code suffering a finite rate of fabrication defects.

4.3 Threshold theorem with fabrication defects

In this section, I will formally prove that a non-zero threshold exists for a surface code with a finite rate of fabrication defects, which is the main takeaway of this work. The section is composed of three subsections. First, I will describe the fabrication defect and error models that will be assumed throughout the proof. Second, I will rigorously prove the threshold theorem for the quantum memory case - that is when no other logical gate apart from the identity is being implemented. This allows me to develop tools that are used in the final part of the section. There, I will generalise the proof to the case of universal fault-tolerant computation by considering the effects of our method on a universal logical gate set.

4.3.1 Defect and error models

Let me start by describing the key features of the distribution of fabrication defects and defining some of the terminology that will be used later. I will draw on the work [67] to characterise the independent and identically distributed configuration of fabrication defects, see Proposition 7 in Appendix B of [67] for details. Let me briefly summarise their results. Specifically, it is shown that we can decompose a configuration of defects into connected components of unit cells, that are called *clusters*. Clusters are characterised by different length scales, or *levels*, that are indexed by integers $0 \leq j \leq m$ such that a level j component has a linear length of at most Q^j, and is separated from all other clusters of level $k \geq j$ by distance at least $Q^{j+1}/3$ where $Q \geq 6$ is an integer constant that we can choose. I will show that a surface code of size $d \times d$ demonstrates a threshold if it is based on a qubit array that is operable in the following sense

Definition 4.1 (Operable qubit array)**.** *The qubit array is operable if its highest cluster level m satisfies $Q^m \leq d/5 - 3$,*

for some positive integer Q that I will bound later. The choice of integers in this definition will become clear later.

Indeed, given a low enough fabrication defect rate, the authors of [67] show the probability of finding a cluster of linear size larger than $d/5 - 3$ is exponentially suppressed. Specifically, given a qubit array of volume $V \sim d^2$, the probability of the occurrence of a level m cluster with $Q^m > d/5 - 3$ is upper bounded by $p_{fail} = \exp(-\Omega(d^\sigma))$, where $\sigma \approx 1/\log Q$, as long as the fabrication defect rate f on any given unit cell is smaller than $f_0 = (3Q)^{-4}$. The proof can be found in [67]. As we have assumed that we can test a qubit array for the locations of fabrication defects offline, we can discard any devices that have clusters of fabrication defects larger than $d/5 - 3$, but as shown in [67] , given a sufficiently small fabrication defect rate, the likelihood of discarding any given device is vanishingly small.

In practical terms, this means that smaller defects that are close to each other are clustered together. This ensures that defect clusters of specific sizes are well separated from each other, otherwise, they would have formed a new, bigger cluster. Following this procedure implies that a cluster may also contain some well-functioning elements of the qubit array. It would be wise to use these elements

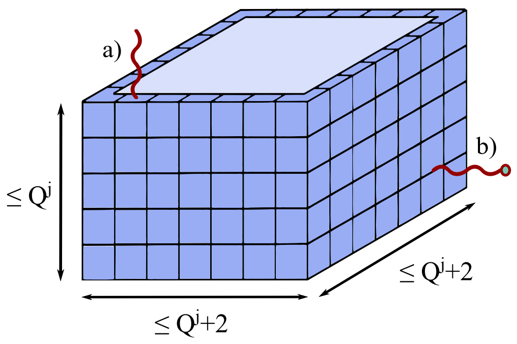


Figure 4.8: A smooth shell of level j. (a) An example temporal error that terminates at the shell due to an erroneous Z type gauge measurement. (b) An example bit-flip spatial error string that terminates at the shell.

practically, but for the simplicity of the following proof, we will treat everything within each cluster as a defect. Therefore, to deal with a level j cluster of fabrication defects, we quarantine all of the qubits within the smallest rectangle that contains the cluster. Meaning that this rectangle of inactive qubits has a linear size at most Q^j. The intact unit cells at the boundary of the square are then used to measure the gauge operators around the perimeter of the cell; these give rise to the shells introduced in Subsection 4.2.2. Due to the additional space required to construct the gauge operators, the shells have a linear size at most $Q^j + 2$ (c.f. Fig. 4.6(b)).

As we have proposed, we alternate between the two types of gauge operators with a frequency that is inversely proportional to the size of the fabrication defect cluster. For simplicity, I choose to alternate the type of gauge every Q^j rounds for a cluster of level j. This results in shells of level j with a spatial side length at most $Q^j + 2$ (given as the linear size of the super-stabiliser) and the temporal side length of Q^j (See Fig. 4.8). I define the diagonal width of the shell to be the Manhattan distance between two corners of the shell with the largest separation. Therefore, the diagonal width of a level j shell is at most $2(Q^j + 2) + Q^j = 3Q^j + 4$.

In the rest of the subsection, I shall use these results to prove the following bounds. First, I will bound the number of error string termination points for individual shells. Second, I will bound the distance between the same type shells. Both temporal and spatial directions need to be considered in each case. These results will allow me to upper bound the number of non-trivial error configurations and lower bound the weight of error strings that span shells across the space-time lattice.

Lemma 4.1. *For a smooth or rough shell of level j, there are at most $40Q^{2j}$ points where the respective error strings may terminate.*

Proof. It is enough to examine the largest size smooth shell of level j as any rough shell of the same level is strictly smaller by construction and, hence, has fewer termination points. The spatial and temporal length of a smooth shell of level j is bounded by $Q^j + 2$ and Q^j respectively, therefore the four sides of the shell have at most $4(Q^j + 2)(Q^j + 1) = 4Q^{2j} + 12Q^j + 8$ points where bit-flip errors may terminate without signature. Finally, one can account for temporal errors due to the erroneous initial and final measurements of the other gauge. There are at most $4(Q^j + 1)$ such termination points at both - the top and bottom of the shell. See Fig. 4.8 for a visual guide. Adding all of the contributions together we upper bound the number of termination points of any shell of level j as $4Q^{2j} + 20Q^j + 16$. For simplicity of later calculations I further upper bound this number by $40Q^{2j}$, where I have used that $j \geq 0$ and Q is a strictly positive integer. □

In addition, the choice of gauge alternation frequency allows me to prove the following Lemma about the shell separation on the space-time lattice:

Lemma 4.2. *Let $Q \geq 9$, then any shell S_j of level j is separated from any other shell of level k of the same type, S_k, as*

$$D(S_j, S_k) \geq Q^{\min(k,j)}, \tag{4.5}$$

where distance D is defined on the space-time lattice of unit cells and corresponds to the infinity norm.

Proof. To prove this Lemma I bound the spatial and temporal separations of the shells independently. First, I bound the spatial separation of shells that belong to different clusters. Then, I bound the temporal separation of shells that belong to the same cluster. Combining these bounds one arrives at the final result.

Let us start by considering the spatial separation D_s of shells belonging to different clusters. Recall that a cluster of level j with linear size at most Q^j is separated from any other cluster of level $k \geq j$ by spatial distance at least $Q^{j+1}/3$. Since the shells enclosing clusters have a spatial size at most $Q^j + 2$, any shell S_j is spatially separated from any other shell S_k with $k \geq j$ and belonging to a different cluster by $D_s(S_j, S_k) \geq Q^{j+1}/3 - 2$. Using that $Q \geq 9$ and $j \geq 0$ one can find $D_s(S_j, S_k) \geq 9Q^j/3 - 2 = 3Q^j - 2 \geq Q^j$. The same calculation applies for the case when $j \geq k$ but with j interchanged with k. Therefore, $D_s(S_j, S_k) \geq Q^{\min(k,j)}$ irrespective of the relative sizes of k and j.

Now, consider the temporal separation D_t of shells belonging to the same cluster. Recall, that we alternate between smooth and rough type shells every Q^j time steps around a cluster of level j. Therefore, any two same-type shells S_j and S'_j around the same cluster (hence the same level j) have a temporal separation $D_t(S_j, S'_j) \geq Q^j$.

Given that the infinity norm is used, the total separation $D(S_j, S_k)$ is the maximum between spatial and temporal separations. For the shells belonging to different clusters, I lower bound $D(S_j, S_k) \geq D_s(S_j, S_k)$ while for the shells belonging to the same cluster, I use $D(S_j, S_{k=j}) = D_t(S_j, S'_j)$. By combining these results I have proven the Lemma.

□

While in general this is a loose lower bound, it allows one to consider separations across space and time in a homogeneous way.

With this characterisation of the defect error model, it remains to determine if the functional qubits of the qubit array can successfully deal with generic random errors, namely bit-flip, phase-flip, and stabiliser measurement errors. Like before, we assume that the intact cells are subject to a standard phenomenological noise model where their data qubits suffer independent and identically distributed X or Z errors with probability ϵ, and a stabiliser or a gauge operator returns an incorrect measurement outcome with probability $q = \epsilon$. In the following part of the proof, I concentrate on a subset of errors, namely, bit-flip (X) errors and measurement errors of Z-type measurements. This is acceptable since one can deal with X and

Z data qubit errors (together with respective Z and X type measurement errors) separately with the surface code [61]. Likewise, I only consider shells with smooth boundaries, as these shells detect X error strings. An equivalent proof will hold for the conjugate category of errors, with only small changes to the details, e.g. the size of the same level cluster is a single unit smaller (see Fig. 4.7). The generalisation of the following results to the circuit-based noise model is straightforward with only small changes to the constant factors in the proof, see e.g. [26, 68].

4.3.2 Threshold theorem for phenomenological noise

In this subsection, I will prove that the construction of shells on a planar surface code architecture gives rise to a scalable quantum *memory* under the phenomenological noise model as long as the qubit array is operable. In the next subsection, I will generalise these results to a fault-tolerant universal quantum computation.

Theorem 4.1. *Suppose a space-time lattice that is generated from an operable qubit array of linear size d and code deformations as presented above. Then, there exists a phenomenological error rate threshold ϵ_0 such that for independent and random errors with rate $\epsilon < \epsilon_0$ the probability of the logical failure is at most*

$$\overline{P} = \exp(-\Omega(d^\eta)) \tag{4.6}$$

for some constant $\eta > 0$.

To prove this theorem, I will start by drawing parallel arguments from the Subsection 4.1.2. Later, I will insert the necessary details to complete the proof. In Subsection 4.1.2 we expressed the logical failure rate is expressed as

$$\overline{P} = \sum_{E \in \mathcal{F}} p(E), \tag{4.7}$$

where $\mathcal{F}$ denotes the set of errors that lead to a logical failure, and $p(E)$ denotes the probability that the error E is drawn from the error model. I will prove the threshold theorem by showing that $\overline{P}$ rapidly vanishes as the code distance diverges for a suitably low but constant physical error rate.

Recall, that the crucial part of the threshold theorem proof, as shown by Dennis *et al.* [61], is to characterise the set of errors $\mathcal{F}$. They propose to do so by assuming a minimum-weight perfect-matching decoder that finds the least-weight correction and considering the minimal conditions that cause it to fail. In this proof, I will also assume that for a space-time lattice of shells, one can use the minimal-weight perfect-matching decoder to find the least-correction. Let me argue why. Having shells on the lattice still cause errors to create syndrome events in pairs (c.f. Fig. 4.7), however, finding the shortest path between two events is not as simple as in the defect-free case. There are pathways that lead through a shell, and therefore, a reduced number of qubit errors are necessary to form these paths compared to the equivalent paths in the defect-free case. Nevertheless, classical graph algorithms such as Dijkstra's algorithm [69] can find the shortest path efficiently and, indeed, the original work of Stace *et al.* [63] already shows how to change the matching graph in the presence of super-stabilisers. This matching graph can then be used for the minimal-weight perfect-matching (MWPM) decoder. The extension of their ideas to the space-time lattice with shells is straightforward.

Just as before, the decoder fails whenever its proposed correction error chain E_{min}, together with the original error E traverse some non-trivial path $\boldsymbol{\ell}$ such that $E + E_{min}$ is a logical operator. For an L by L surface code without defects, such a chain has a minimal length $\ell \geq d = L$. In our case, as discussed above, shells compromise the distance of the code, hence, we must obtain a new lower bound on the length L of non-trivial paths over the lattice.

Moreover, recall that Dennis *et al.* [61] upper bounds $N(\ell)$, the number of error configurations that lead the decoder to give rise to a non-trivial path of length ℓ, by counting the number of walks on the space-time lattice that constitute error strings. The number of starting points of the walk V is bounded by the volume of the space-time lattice which is polynomial in the code distance d. This matches our case. Finally, each step of the walk can go in ν directions, where ν is the valency of the lattice. This part greatly differs for the case of having a space-time lattice with shells. Let me briefly explain how. Consider forming a walk by adding one step at a time. Away from any shells, any new step can be taken in $\nu \approx 6$ directions as it was in the defect-free case. Here, the 6 directions correspond to a step in one of 4 spatial directions or 2 temporal directions. Once the walk, modelling an error string, has approached the boundary of the respective shell, it may terminate and continue at any other location on the boundary of the same shell. Such a walk should still be considered in our calculations since the shell will contain an even parity of errors and therefore will not result in a syndrome event. Therefore, a new upper bound for $N(\ell)$ needs to be derived.

Once we have obtained new bounds for L and $N(\ell)$, we can upper bound $\overline{P}$ just as before

$$\overline{P} \leq \sum_{\ell \geq L} N(\ell)\Pi_{\ell}, , \tag{4.8}$$

where the associated weighting is unchanged, i.e. $\Pi_{\ell} \leq \epsilon^{\ell/2}$, due to us having the ability to find the least-weight correction using the MWPM decoder.

4.3.3 Proof of Theorem 4.1

In the remainder of this section, I prove Theorem 4.1 by upper bounding $N(\ell)$ and lower bounding L for the case of a space-time lattice with shells. Note that one needs to consider non-trivial paths travelling in both spatial and temporal directions, because, in the generality of fault-tolerant logical gates, logical errors may also occur due to paths traversing the temporal direction. Additionally, note that the formulation of my proof is agnostic to such details, i.e. the space and time-like directions are isotropic (c.f. Eq. 4.5). This is exactly what I meant earlier by writing that the shell picture homogenises space and time.

Let me first formalise the definitions and introduce the rest of the terminology required. The errors in $\mathcal{F}$ are characterised in terms of walks of a given length that contain some requisite number of errors in their support. Let ℓ denote the length of a walk $\boldsymbol{\ell}$, where the walk can make progress with steps along the spatial or temporal directions of the space-time lattice. A spatial step of the walk is to move from one cell to any other cell that shares an edge with it (a qubit error) at some time t, while a temporal step of the walk is to stay on the same cell from time t until time $t+1$ (a measurement error, either a stabiliser or a gauge operator). One says that a walk is *non-trivial* if it starts at one boundary of the space-time lattice and ends at the respective other boundary. These boundaries should not be confused with the boundaries of shells.

A walk *encounters* a shell if the error string along the walk terminates at that shell. For example, a walk of bit-flip errors may encounter a shell with smooth boundaries. As noted before, after encountering a shell, the walk can continue from any other point of the same shell. As in [61], only *self-avoiding* walks are explicitly considered for which every site, including shells, is visited at most once. This is justified because walks that cross any given site, or shell, more than once have an equivalent action on the code space as an error configuration that contains a self-avoiding walk. Moreover, walks that are not self-avoiding are implicitly accounted for by the weighting function Π_ℓ.

With these definitions and the fabrication defect distribution from Subsection 4.3.1 the following lemma can be proven. This lemma will be helpful later in obtaining bounds on $N(\ell)$ and L.

Lemma 4.3. *Let $Q \geq 9$ and assume an operable qubit array. Then on the space-time lattice any self-avoiding non-trivial walk $\boldsymbol{\ell}$ may encounter at most*

$$n_j \leq 2 \times \frac{15^j \ell}{Q^j} \tag{4.9}$$

shells with smooth boundaries of level j. Moreover, as a consequence, the length of an augmented walk, i.e, a walk that contains shells of all sizes up to k, stretches over a length

$$\ell'_k \leq 15^{k+1}\ell. \tag{4.10}$$

Proof. Consider a long walk in our space-time lattice. It may encounter multiple shells, and the diagonal width of these shells will contribute to the effective walk length.

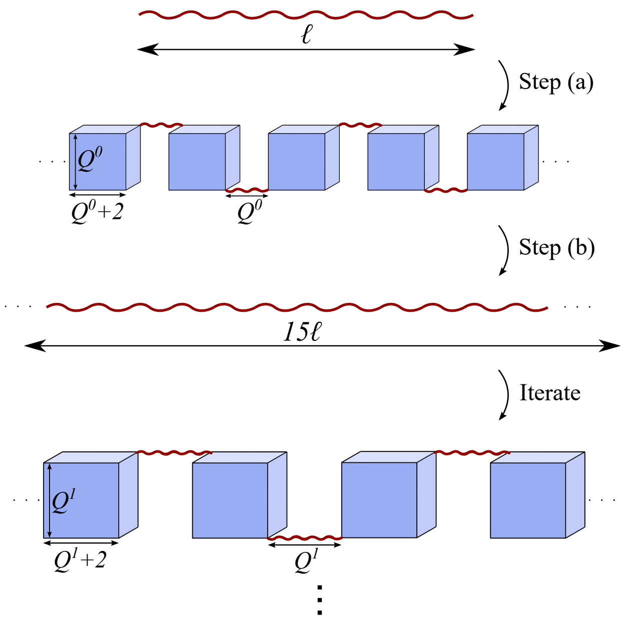


Figure 4.9: Diagram showing the first steps of the proof of Lemma 4.3. Let us start with a self-avoiding walk of length ℓ at the top. Step (a) inserts the maximum number of level $j = 0$ shells that a walk can encounter. Step (b) then replaces this augmented walk with a shell-free walk of the same length. The same steps are repeated with increasing j until the largest level m shells have been accounted for.

We wish to bound the number of shells that the walk may encounter. To do so, start from a shell-free walk, and insert shells systematically at the highest possible density. Then a considerable simplification can be made - regard our augmented walk as a shell-free walk of a new, greater length. Then repeat the same shell-insertion process now with shells one level greater in size. This will lead to an over-counting of the number of shells encountered, but it will permit one to reach an analytic expression. The approach is illustrated in Fig. 4.9.

The Lemma is proved by induction. Given that $Q \geq 9$, Lemma 4.2 shows that at most $n_0 = \ell/Q^0 + 1$ shells of level $j = 0$ can be encountered by a walk since any two shells are at least Q^0 separated. The $+1$ term arises from considering shells at both ends of the walk. The analysis can be further simplified by upper bounding $n_0 \leq 2\ell/Q^0$, where I have used the fact that any non-trivial walk has a length $\ell \geq Q^0$ on an operable qubit array. Moreover, the length of the walk ℓ together with the diagonal width of all level 0 shells, $3Q^0 + 4$, have a combined effective length at most $\ell'_0 = n_0(3Q^0 + 4) + \ell \leq n_0 7Q^0 + \ell \leq 14\ell + \ell = 15\ell$. This concludes the base case for $j = 0$.

Next, assume the Lemma holds for $j = k$. We are left to show that it holds for $j = k + 1 \leq m$. Consider a new shell-free walk with length ℓ'_k instead, this allows us to over-count the number of $k + 1$ shells encountered by our original walk of length ℓ. There can be at most $n_{k+1} = \ell'_k / Q^{k+1} + 1 \leq 2 \times 15^{k+1}\ell/Q^{k+1}$ shells of level $k + 1$ that can be encountered by such a walk since any two shells of this level are at least Q^{k+1} separated. Here again I have simplified analysis by converting the $+1$ term to a multiplicative factor of 2. This simplification is justified due to the fact that every non-trivial walk on the space-time lattice will be long enough such that their respective augmented walk must have length $\ell'_k \geq Q^{k+1}$ for $k \leq m - 1$ given that the operability condition is satisfied. A more general proof can be found in Subsection 4.3.4. Now the walk ℓ'_k, which already accounts for all shells up to level k, together with all level $k + 1$ shells has a combined effective length at most $\ell'_{k+1} = n_{k+1}(3Q^{k+1} + 4) + 15^{k+1}\ell \leq n_{k+1}7Q^{k+1} + 15^{k+1}\ell \leq 15^{k+2}\ell$.

The inductive step holds until the largest shells of level m have been accounted for. □

With Lemma 4.3 in hand, one can find an upper bound for $N(\ell)$, the number of error configurations lying on a path of length ℓ, and a lower bound for the length of the shortest non-trivial path, L.

Recall that the key idea to finding an upper bound for $N(\ell)$ is to count the set of self-avoiding non-trivial walks of length ℓ. Let us denote this number by $V \times W(\ell)$, where as before V denotes the starting locations for a walk and $W(\ell)$ takes into account all possible steps that the walk can take. This upper-bound for $N(\ell)$ is justified because the set of random walks of length ℓ must contain all of the non-trivial paths over the space-time lattice of the same length. Finally, one needs to multiply $V \times W(\ell)$ by the number of error configurations of at least $\ell/2$ errors that can lie on a path of length ℓ. Let us denote this number $C(\ell)$ and therefore $N(\ell) \leq V \times W(\ell) \times C(\ell)$. Now, let us continue by upper bounding each of the term in this expression. It is straightforward to check that $C(\ell) < 2^\ell$. Moreover, the volume of the whole space-time lattice is $V = d^3$ and that gives us an upper bound on the possible start locations of a walk. As such, I proceed by obtaining the number of random walks from a fixed starting point, $W(\ell)$ where the walks may proceed through some number of shells that has been upper bounded using Lemma 4.3.

The first step of the walk can be made in $\nu = 6$ directions in the space-time lattice with ν the valency of the lattice, any subsequent step can then proceed in at most in $\nu - 1 = 5$ directions in order for the walk to be self-avoiding. One performs $\ell - 1$ of such steps. This bounds the number of walks with different step configurations as $(6/5) \times 5^{\ell}$. However, to properly upper bound $W(\ell)$ one must account for the shells that the random walk may encounter, since after encountering a respective shell, the walk may continue in many directions corresponding to all of the terminations points of the shell. Lemma 4.3 provides us an upper bound on the number of such encounters, and hence, we have that

$$W(\ell) \leq 6 \times 5^{\ell-1} \prod_{j=0}^{m} (40Q^{2j})^{2\times 15^j \ell/Q^j}, \tag{4.11}$$

where $40Q^{2j}$ is an upper bound of termination points of a level-j shell (Lemma 4.1) that accounts for the number of distinct steps along which a walk can recommence after an encounter, and the exponent $n_j < 2 \times 15^j \ell/Q^j$ is the upper bound on the number of times a walk of length ℓ will encounter distinct level-j shells. The product is taken over shells of all levels that appear on an operable qubit array $j \leq m$.

Let me rearrange Eq. 4.11 to obtain a clearer expression for the logical failure rate. First, I will express the product in Eq. 4.11 as a summation in the exponent. Then each summation is taken up to infinity to easily upper bound its value. Such a rearrangement gives

$$\prod_{j=0}^{m} (40Q^{2j})^{2\times 15^j \ell/Q^j} \leq 40^{2\ell \sum_{j=0}^{\infty} 15^j/Q^j} \times Q^{4\ell \sum_{j=0}^{\infty} j 15^j/Q^j}. \tag{4.12}$$

Finally, using standard geometric series expressions with $Q > 15$ such that $x = 15/Q < 1$, namely

$$\sum_{n=0}^{\infty} x^n = \frac{1}{(1-x)}, \quad \text{and} \quad \sum_{n=0}^{\infty} n x^n = \frac{x}{(1-x)^2}, \tag{4.13}$$

one obtains

$$W(\ell) \leq 6 \times 5^{\ell-1} \times 40^{2\ell Q/(Q-15)} \times Q^{60\ell Q/(Q-15)^2}. \tag{4.14}$$

Therefore, I can upper bound $N(\ell)$ as

$$N(\ell) < VW(\ell)C(\ell) \leq cd^3\rho^\ell \tag{4.15}$$

where $c = 6/5$ and $\rho = 2 \times 5 \times 40^{2Q/(Q-15)}Q^{60Q/(Q-15)^2}$ are constants.

The lower bound L is also readily obtained from Lemma 4.3. Let d be the linear size of the space-time lattice, then any non-trivial augmented walk must have an effective length $\ell'_m \geq d$. Using Eq. 4.10 we can see that the maximum effective length that a walk of length ℓ can achieve is $\ell'_m \leq 15^{m+1}\ell$ when $m \leq \log_Q(d/2-1)$ in the case of an operable qubit array. Therefore, any non-trivial walk must be at least of length

$$L = 15^{-(m+1)}\ell'_m = 15^{-(m+1)}d. \tag{4.16}$$

Finally, we can combine the obtained equations to complete the proof of Theorem 4.1. Substituting the expression of $N(\ell)$ given in Eq. 4.15 into Eq. 4.8 together with Eq. 4.3 enables us to bound the logical failure rate as follows

$$\overline{P} \leq cd^3 \sum_{\ell \geq L} \rho^\ell \epsilon^{\ell/2} = kd^3 \left(\rho\epsilon^{1/2}\right)^L = kd^3 \exp(-\Theta(L)), \tag{4.17}$$

where I have taken $\rho\epsilon^{1/2} < 1$ and where $k = c/(1-\rho\epsilon^{1/2})$ is a positive constant term.

Substituting Eqn. 4.16 into Eqn. 4.17 reveals the exponential (technically sub-exponential) decay in the logical failure rate. That is,

$$\overline{P} \leq kd^3 \exp\Big(-\Theta(15^{-(m+1)}d)\Big) \leq kd^3 \exp(-\Theta(d^\eta)), \tag{4.18}$$

where $\eta = 1 - \log_Q(15)$. By further observation an exponential decay in the logical error rate is achieved as long as $Q > 15$ and $\rho\epsilon^{1/2} < 1$. A lower-bound for the threshold of the phenomenological model error rate is therefore $\epsilon_0 \equiv 1/\rho^2$. This completes the proof of the main theorem.

4.3.4 Universal quantum computation

In this subsection, I will show that fault-tolerant quantum computation is possible on a planar array of qubits with fabrication defects. The computational techniques I have developed in the previous section are easily adapted to the computational case in which a universal set of fault-tolerant gates is implemented using deformations of the surface code [26, 61, 70–73]. The following proof considers deformations that implement a minimal set of universal gates, however, the arguments presented in this subsection can be tailored to an arbitrary choice of them [73, 74]. The universal gate set I consider is constructed as follows. I will take a Hadamard gate as presented in [61] and consider performing entangling operations with lattice surgery [72]. Given these fault-tolerant gates, one can complete the set by distilling magic states [25] and Pauli-Y eigenstates [26, 70].

Hadamard gate

The authors of [61] propose implementing a logical Hadamard gate by rotating the corners of the surface code around its boundary [73] before transversally performing a Hadamard gate on each data qubit followed by a swap operation. See also [26]. The latter unitary operation only maps between the two different types of error strings through a plane in the space-time lattice. Therefore, let me concentrate on the physical transformations of the lattice boundaries. Fig. 4.10 shows this operation on a space-time lattice with shells.

The transformation of the surface code boundaries must be done in a way that preserves the code distance. As one can see in Fig. 4.10(b), during the switch of the boundaries, one may have purely diagonal strings of errors that lead to a logical error compared to the static quantum memory case. For the threshold theorem to hold, only the proof of Lemma 4.3 needs to be reconsidered by taking into account such diagonal paths. Lemma 4.3 already captures the diagonal extent of the shells, however, for it to hold, one needs to show that every non-trivial walk will be still long enough such that their respective augmented walk has length $\ell'_k \geq Q^{k+1}$ for $k \leq m-1$. I will prove it formally.

Proof. Consider a single shell of the largest level m along a non-trivial walk. Then given an operable qubit array ($Q^m \leq d/5-3$), the diagonal width of such a shell is at most $3Q^m+4 \leq 3/5d-5$, where d is the linear size of the code. Since the Manhattan distance between the lattice boundaries of the same type is preserved during their transformation, the respective augmented walk of a non-trivial walk with all shells of level $\leq m-1$ has to be at least $\ell'_{m-1} \geq d-(3/5d-5) = 2/5d+5 \geq d/5-3 \geq Q^m$, where I have subtracted the maximum diagonal width of a single level m shell. If there is more than one shell of level m along the non-trivial walk, then by Lemma 4.2, their mutual separation is at least Q^m and the result holds trivially.

Moreover, having this result hold for the biggest shells of level m implies that it holds for all shells of level $\leq m$. Let me show it by induction, where I have already proven that $\ell'_j \geq Q^{j+1}$ for the base case $j = m-1$. I am left to show that if the result holds for the $j = k$ case then it also holds for $j = k-1$ case. Consider a single shell of level k along a non-trivial walk. It has a diagonal width of $3Q^k+4$. Given that $\ell'_k \geq Q^{k+1}$ holds from the $j = k$ case, the respective augmented walk of a non-trivial walk with all shells of level $\leq k-1$ has to be at least $\ell'_{k-1} \geq Q^{k+1} - (3Q^k+4) \geq Q^k$. Similar to the previous argument, if there is more than one shell of level k along the non-trivial walk, then by Lemma 4.2, their separation to any shell of level k or greater is at least Q^k and the result holds trivially.

One may repeat the inductive steps until they have proven that $\ell'_k \geq Q^{k+1}$ for any $0 \leq k \leq m-1$. □

This slight extension of the proof ensures that Lemma 4.3 holds. The rest of the proof of the threshold theorem follows from the memory case.

Lattice surgery

Lattice surgery was introduced to perform entangling operations between surface codes by logical parity measurements [72]. In its simplest form, two adjacent surface code patches are merged using a routing space which consists of new low-weight stabilisers. Then the logical parity measurement is made by measuring the parity of the new stabilisers of the merged code. This makes sense since the logical parity

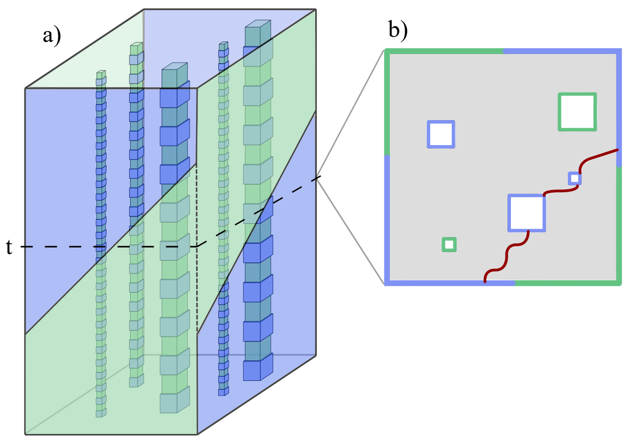


Figure 4.10: (a) Implementation of a Hadamard gate in the space-time picture. (b) A possible path of errors that lead to a logical failure during the switch of the boundaries at time t.

measurement is a member of the stabiliser group of the merged code. Once one has repeated these measurements over a longer time, i.e. $\approx d$ rounds to maintain the distance and reliably infer the outcomes, the codes are separated by collapsing the routing space with single-qubit measurements on all newly introduced data qubits.

In our case, one needs to make sure that the value of the logical parity measurement can still be inferred even when the routing space contains fabrication defects. Naturally, our shell construction can be used to reliably evaluate the necessary super-stabilisers that are near or on the routing space. Moreover, it is easy to check that the logical failure rate of lattice-surgery operations is greatly suppressed using a qubit array even with fabrication defects. Broadly speaking, compared to the memory proof, the only difference is that the lattice has become bigger (during the merger of the two codes).

Let me restate this argument in a more formal sense. Unlike the case of a quantum memory, logical errors during lattice surgery operations can also occur if error strings extend along the temporal direction of the routing space while the codes are merged. However, Lemma 4.3 already considers paths that traverse the temporal direction. As such, the only quantitative change from the memory case proof is the factor V denoting the starting points of a walk. Considering that the volume of an efficient lattice surgery operation, which includes the volume of the two codes as well as that of the routing space, is also a polynomial of code size, $\sim 2d^3$,

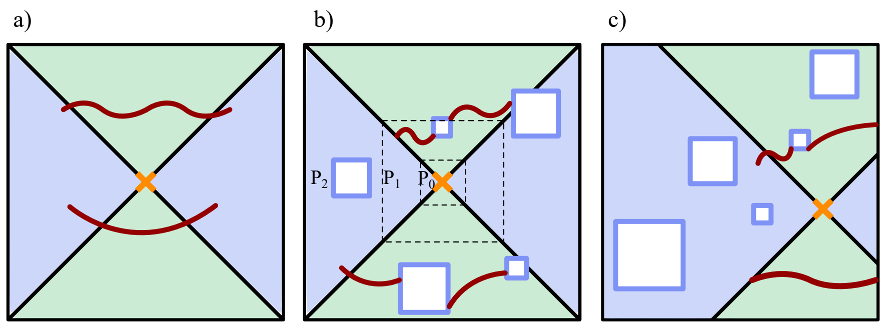


Figure 4.11: (a) State injection scheme, where the injection qubit is denoted with an orange cross and blue(green) qubits are initialised in the $|+\rangle(|0\rangle)$ basis. Error strings in space-time (red) may form from one boundary to the respective other boundary and cause a logical failure. (b) Fabrication defects increase the noise acting on the logical input state. Panels P_j are defined in the main text whereby no level-j shell is supported on panel P_j. Here the orange cross marks a *good injection point.* (c) In general, a good injection point may be found off-centre.

the number of logical string starting points is of order $V = O(d^3)$. The exact number depends on the size of the routing space. Therefore, one can rewrite the Eqn. 4.18 as

$$\overline{P} \leq kO(d^3)\exp(-\Theta(d^\eta)), \tag{4.19}$$

where the exponential suppression of the logical error probability with respect to the code size is apparent.

The above assumes an operable qubit array, hence, let me briefly comment on that. Consider a qubit array that is large enough to perform a lattice surgery operation. It is exponentially likely, with respect to the code size, that the array is operable, assuming the fabrication error rate is sufficiently low but constant. This is shown by applying the same arguments from [67] (c.f. Subsection 4.3.1) since the width of the qubit array needs only to be of order $O(d)$.

S and T gates

Given the aforementioned logical gates, one can complete a universal gate set with distillation and teleportation protocols. Here, I have chosen to complete the (non-universal) Clifford gate set with a phase gate S implemented through a gate teleportation protocol shown in [75]. The procedure requires one to distil an eigenstate of the Pauli-Y operator [26, 70]. This is done using noisy state injection [76, 77], together with the Hadamard gates and lattice surgery operations that I have already shown to be resistant to fabrication defects. Then, given the ability to perform all of the Clifford gates, in addition to noisy state injection, one can use magic state distillation and teleportation circuits to implement a T gate for universal quantum computation [25].

Therefore, it remains to discuss and show that noisy state injection can be reliably done on a two-dimensional qubit array with a finite rate of fabrication defects. In the ideal case, we initialise noisy ancilla states by preparing the qubit array with some qubits in the $|+\rangle$ state and others in the $|0\rangle$ state, together with a single qubit prepared in the desired input state $|\psi\rangle$. Then the stabilisers of the surface code are measured to encode the input state into the whole code. A suitable input configuration is shown in Fig. 4.11(a), where blue (green) qubits are initialised in the $|+\rangle$ ($|0\rangle$) basis, respectively, and the central qubit is prepared in the state $|\psi\rangle$ (orange). For our purposes, $|\psi\rangle$ is either $|Y\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle)$ or $|A\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}\,|1\rangle)$ depending on whether one wants to perform an S or T gate respectively.

It is important to argue that the shell construction does not limit one's ability to prepare an input state with a bounded logical error rate at *any* scale. Provided that is the case, one can prepare a noisy logical state of a necessary fidelity (usually set by the state distillation scheme) given that the physical error rate is appropriately low but constant. Formally, I will show that one can initialise a state with a finite logical error rate that does not diverge with the size of the code below some threshold rate of the phenomenological error model. My argument will first assume that one can find a good injection point, as I define it below. Using this assumption, I will prove that one can perform state injection with bounded probability. Finally, I will argue that the good injection point exists in some region of a qubit array with exponentially high probability provided that the fabrication defect rate is below some threshold, therefore, confirming the validity of the aforementioned assumption.

It is possible to initialise an ancilla state given a *good injection point*, i.e., a region of the lattice that is distant from fabrication defects. Let us regard any unit cell v as a good injection point if

$$D(v, u_j) \geq 2Q^j + 2 \tag{4.20}$$

for all unit cells u_j contained within a level-j shell. This means that a good injection point is further away from larger clusters of fabrication defects compared to smaller ones. See Fig. 4.11(b) for an example.

The goal is to show that the logical error rate of an input ancilla state is bounded given a good injection point. Here, I will use similar arguments presented in the previous section and will obtain an expression for the logical failure rate by counting the number of ways in which a logical error can occur. Again, this value is found in terms of random walks. In this case, the upper bound on the number of non-trivial walks will depend on the number and sizes of shells in the vicinity of a good injection point. Similarly, a new lower bound on the length of the walk that can result in a logical error needs to be considered. Specifically, this lower bound will depend on the starting location of the walk.

To begin, assuming a good injection point, we can use the result from Lemma 4.3 to upper bound the number of shells that a non-trivial walk may encounter. It is justified to use Lemma 4.3 because any non-trivial self-avoiding walk that encounters a level j shell has $\ell'_{j-1} \geq Q^j$ where ℓ'_{j-1} is the length of the augmented walk excluding level j shells or larger. This justification is depicted geometrically in Fig. 4.11(b) and follows from proof in Subsection 4.3.4. By letting $Q \geq 9$, one can therefore say that any non-trivial self-avoiding walk of length ℓ from one boundary to the respective other boundary encounters at most $2 \times \frac{15^j \ell}{Q^j}$ shells of level j. Note that this bound is exactly the same as the one we had for the memory case.

Next, I count the number of starting points from which a random walk can begin. To simplify the following calculation I will discretise the qubit array in disjoint panels around the injection point as shown in Fig. 4.11(b). It is done in a way such that panel P_j has edges that are a distance of $2Q^j + 2$ away from the good injection point. This enables me to upper bound the number of starting points s_j of any walk originating from within panel P_j as $s_j \leq (4Q^j + 4)^2$.

Given the same arguments as in the previous section, one can upper bound the number of error configurations that give rise to a non-trivial path of length ℓ originating from panel P_j as

$$N_j(\ell) \leq c(4Q^j + 4)^2 \rho^\ell, \tag{4.21}$$

where $c = 6/5$ and $\rho = 2 \times 5 \times 40^{2Q/(Q-15)} Q^{60Q/(Q-15)^2}$ are the same constants as before. One can see that in this expression only the number of starting locations have changed compared to the memory case. The plurality of them is now dependent on the panel that the walk is starting from.

The final ingredient to bound the logical error rate is to find a lower bound on the minimal length of the walk. Any non-trivial walk starting from within panel P_j must have an effective length $\ell' \geq 2Q^{j-1}$. Due to Eq. 4.20 describing a good injection point, no shell of level j may be found inside panel P_j. This implies that the minimum length of any non-trivial walk originating from panel P_j must satisfy $L_j \geq 15^{-j}\ell' \geq 15^{-j}(2Q^{j-1})$. Note that this expression accounts for the case where the injection point is found close to the boundary of the surface code, see Fig. 4.11(c).

Taking the expressions for $N_j(\ell)$ and L_j for each panel P_j and recalling Eq. 4.8 and Eq. 4.3 we can bound the logical failure rate as

$$\overline{P}_{\text{in}} \leq \sum_{j=0}^{n} \sum_{\ell \geq L_j} N_j(\ell)\epsilon^{\ell/2} \leq c\sum_{j=0}^{n} s_j \sum_{\ell \geq L_j} \rho^\ell \epsilon^{\ell/2} = k\sum_{j=0}^{n} s_j (\rho\epsilon^{1/2})^{L_j} \leq O(1) \tag{4.22}$$

as long as $Q > 15$ and $\rho\epsilon^{1/2} < 1$, which is satisfied for $\epsilon < 1/\rho^2$. Here, $k = c/(1-\rho\epsilon^{1/2})$ is again a positive constant and $n \geq m+1$ denotes the total number of panels. Consequently, there must exist a phenomenological error rate ϵ, independent of the system size, below which $\overline{P}_{\text{in}}$ is low enough for the encoded qubit to be used in some respective distillation protocol.

Finally, I will show that a good injection point can be found with an exponentially high probability for a fabrication defect rate below some threshold. I argue that the point exists as long as $Q \geq 33$ and the biggest cluster of level m has size $Q^m \leq d/5 - 3$, that is, the qubit array is operable as defined in Definition 4.1. As can be seen in [67] this condition is satisfied with exponentially large probability with respect to the qubit array size below the fabrication defect rate $f_0 = (3Q)^{-4}$.

From the properties of cluster decomposition (c.f. Subsection 4.3.1) any cluster of level j is separated from all other clusters of level $k \geq j$ by distance $Q^{j+1}/3$. Hence, for $Q \geq 33$ any two shells of the same level $j \geq 1$ are separated by distance at least $3(2Q^j + 2)$. This allows us to find a region of linear size $2Q^j + 2$ between these shells such that the unit cells inside this region are separated by distance at least $2Q^j + 2$ from all shells of level j. Now, this region may contain a shell of level $j - 1$. One can use the same argument to find an even smaller region in which all cells are separated from all level $j - 1$ shells by distance at least $2Q^{j-1} + 2$. The same argument is iteratively applied to smaller and smaller regions until level 0. The region found at level 0 corresponds to the set of good injection points. Note that one requires $Q^m \leq d/5 - 3$ to ensure that for all m the region of good injection points is found within the boundaries of the surface code.

I have shown that given a good injection point one may encode a qubit in an arbitrary state with a finite rate of logical failure. Furthermore, I have shown that such a good injection point can be found with exponentially high probability with respect to the system size below some fabrication defect rate. Therefore, together with fault-tolerant implementation of lattice surgery and Hadamard gates, one can perform state distillation protocols for both S and T logical gates fault-tolerantly. This concludes the proof that universal quantum computation is scalable on a two-dimensional array with a finite rate of fabrication defects.

4.4 Cosmic rays and time-correlated errors

Several defect-like events can compromise a two-dimensional qubit architecture over a long time. The above analytic threshold result assumed permanent fabrication defects whose locations could be determined offline before the quantum computation. However, there exist time-correlated errors that might arise during the computation. In this section, I will discuss how the shell protocol can be applied to time-correlated errors and how one might be able to detect them.

For example, consider the errors introduced by cosmic rays [78, 79], where the qubit array absorbs a large amount of energy that significantly increases the error rate of the qubits in some region. It may be expected that the qubits return to their operational state as the energy dissipates. This can take a long time, and it could be advantageous to regard the qubits as faulty and isolate them from the rest of the code using the shell protocol while the energy dissipates. If one decides to continue using the qubits that are significantly more noisy than others, then the logical performance of the whole code will be compromised mainly due to the same difficulties of reliably estimating the parity measurements of local stabilisers. Therefore, the generalisation of the shell protocol to general time-correlated errors might find its use in many quantum computing platforms. Let me now introduce this generalisation, where the main ideas are summarised in Fig. 4.12.

Assume that a time-correlated error occurs at time t_1. It might take some time to infer the occurrence of such an error, and therefore, one begins forming shells around such errors at time t_2. They are constructed in the same way as in the permanent defect case (Section 4.2) to quarantine the faulty parts of the qubit array from the rest of the surface code. Additionally, at time t_2, all data qubits that fall in the quarantine zone are measured in $|+\rangle / |-\rangle$ basis to infer the value of $\mathcal{A}_P$, where P denotes the corresponding puncture enclosing the defect. After that, one can continue to perform the shell protocol as normal, where, as usual, the types of shells are alternated with a frequency inversely proportional to the size of the puncture.

Assuming the code was initialised much earlier than t_1, one can obtain the value of both types of super-stabilisers reliably even at some time $t_0 < t_1$, that is, before the time-correlated error began. Indeed, all super-stabiliser operators considered in this chapter are members of the stabiliser group at all times irrespective of whether a puncture is created or not. The values of such super-stabilisers are calculated by taking the product of the individual X and Z stabilisers within the region that the super-stabiliser encompasses, see Fig. 4.12. Therefore, to identify the error strings that have entered the faulty region during the cosmic ray discovery time $\Delta = t_2 - t_1$

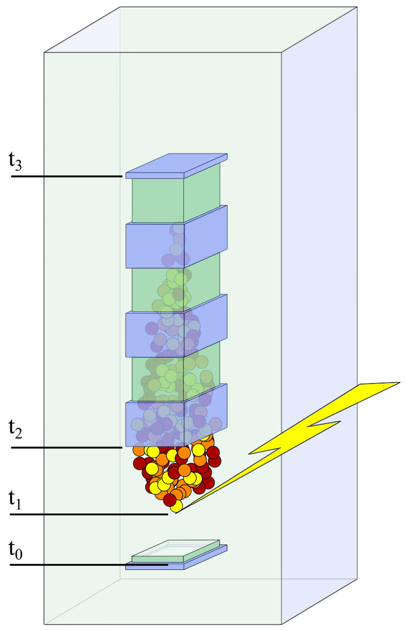


Figure 4.12: A cosmic ray event. The ray hits the qubit array at time t_1. The event is being detected between times t_1 and t_2 after which the affected qubits are isolated with shells. Stabiliser measurements before the cosmic ray hits (e.g. at t_0) can be used to infer the base value of the shells. One waits until the energy due to the cosmic ray dissipates before initialising all affected qubits back into the error-correcting code at time t_3.

one can compare the super-stabiliser value of a time before the expected cosmic ray event took place to the value of the first super-stabiliser of the shell.

It is likely that a time-correlated error will only persist for some time. For example, qubits that have been impacted by, say, a cosmic ray, will become functional again once the energy from the radiation dissipates. This time frame may be much shorter than the total length of the computation. Therefore, one should reinitialise the quarantined qubits back to the surface code by measuring stabilisers of the code once again in the region P. This happens at time t_3 in Fig. 4.12. These measurements give a final reading of the super-stabilisers to complete the measurement of the last shells that are measured around a given puncture. Simply, the value of the new super-stabilisers is compared to the last measurements of $\mathcal{A}_P$ and $\mathcal{B}_P$ that were inferred by the shell protocol. Irreparable qubits can remain isolated until the end of a computation.

I have so far explained how one can prepare shells around a time-correlated error that appears during quantum computation. However, one still needs the ability to recognize these errors as they cannot be spotted before the computation like most permanent defects. A simple argument for monitoring the frequency of stabiliser events (i.e. locations when subsequent measurement outcomes of a stabiliser differ) is sufficient. In a practical setting where the physical error rate is at least a factor of ten below the threshold error rate, one may be worried about time-correlated errors which increase the physical error rate above the threshold. Shells can be used to isolate such significant errors and leave others for the base code to deal with. In this case, to leading order, the frequency of stabiliser events close to the affected area will also increase by a factor close to ten. Therefore, this change should be witnessed within a very short time and the user can start isolating the damaged qubits immediately after detection. Indeed, experimental results [78] have shown that cosmic rays can be identified very rapidly, i.e., on the timescale of a single round of stabiliser measurements. It is therefore reasonable to assume that $\Delta = t_2 - t_1$ can be very small.

More generally, one can identify drifts in noise parameters by measuring the frequency of the stabiliser events. In turn, this helps to improve the performance of a quantum error-correcting code by updating the prior information given to the decoding algorithm [80–82], or experimental control parameters [83]. It may even be advantageous to use the shell protocol to isolate regions of the lattice where the error rate increases dramatically, even if the error rate of the affected qubits remains below the threshold. The spatial locations of time-correlated errors may drift over time as well. This more general case was considered in [84]. These kinds of drifts can be monitored and identified in a similar manner by measuring the frequency of stabiliser events near the puncture. Given that information, one can change the shape and size of the shells on the fly to keep a mobile time-correlated error isolated from the qubit array.

Another category of time-correlated errors are those where a stabiliser measurement device becomes unresponsive and always reports the no-event outcome, independent of the state of the qubit array. For example, the measurement of some Z stabiliser always yields a value of 0. Authors of [85] call this a “silent-stabiliser error”.

A straightforward method is to identify such errors and treat them as time-correlated errors as discussed above. Indeed, one can continually test the responsiveness of the stabiliser readout circuits by manually applying Pauli rotations to the qubits in known locations such that, assuming no other errors occur, the state of a stabiliser degree of freedom is changed with probability one. Let me elaborate on this using an idea that was proposed by Benjamin Brown in [50]. Here, one needs to make sure that they do not lose the ability to identify the standard physical errors the code is designed to detect. For this, one can find and use a Pauli operator that rotates *every* stabiliser degree of freedom in between each round of stabiliser measurements. Now, the stabilisers are regularly measured as normal with this additional step in the periodic readout circuit. However, instead of looking for stabiliser events by looking for odd parity between two subsequent stabiliser measurements, one looks for even parity between two measurements to detect an event. This circuit will identify silent stabiliser errors very quickly, as they will produce detection events at every single measurement round! Using this protocol, one can therefore quickly identify an unresponsive stabiliser and again isolate it from the rest of the code with shells.

In the next section, I will introduce some of my collaborator's work in which cosmic ray events are considered and their effect is numerically studied under the shell formalism.

4.5 Numerical results

So far, I have shown that in the asymptotic limit, there exists a non-zero threshold for the surface code with a finite rate of fabrication defects. Additionally, we have hinted that the same methodology is applicable to temporal defects, such as cosmic rays, which only persist for a finite amount of time and need to be detected during the computation. In this section, I will introduce some of the results from my co-author A. Siegel [51] who compared the novel shell methodology to Stace's and estimated a numerical value of the threshold for this approach. For the latter, Siegel introduced a variant of a decoder that detects time-correlated errors on the fly, but I will not discuss its implementation here as it is not crucial to the shell methodology

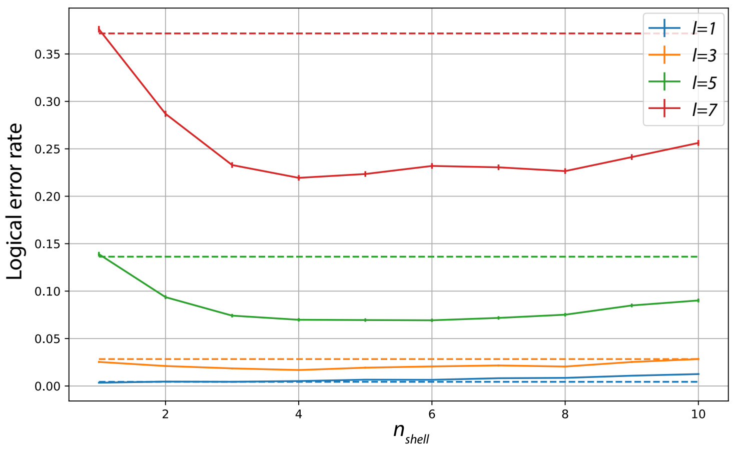


Figure 4.13: The performance comparison of the shell (solid lines) and Stace's (dashed lines) methods. They are compared over a wide range of measurements rounds n_{shell} and for various defect sizes l. The figure was produced by my coauthor A. Siegel in [51].

presented in this chapter. All of these studies are practical extensions of the work presented above, and they confirm the usefulness of the proposals found therein.

4.5.1 Stace's method vs the shell method

The first straightforward idea is to numerically compare the performance of both presented methodologies - Stace's and the shell one. Recall, that the shell method requires a repeated measurement of the same type of gauge operators (forming a super-stabiliser) proportional to the defect width, while Stace's method always alternates between super-stabilisers every measurement round. Therefore, both of these methods coincide in the limit of very small defects and, to see the discrepancy, larger defects need to be considered.

With the above in mind, the first numerical study set up by A. Siegel in [51] is the following. Take the rotated variant of the surface code with a side length $L = 15$. Note that in the defect-free case, the distance of the code matches this side length, i.e. $d = L$. To compare the methodologies, we introduce a square defect with a linear length l in the middle of the code and a parameter n_{shell} denoting the number of measurement rounds that specific type of gauge operators are measured before changing its type. The performance of the surface code is numerically simulated

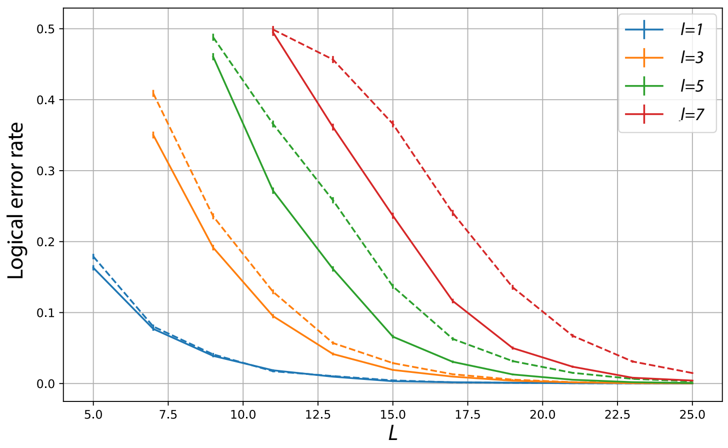


Figure 4.14: The performance comparison of the shell (solid lines) and Stace's (dashed lines) methods. They are compared over a wide range of surface code sizes L and for various defect sizes l. For each defect size the optimal value of measurement rounds n_{shell} was used in the shell method. The figure was produced by my coauthor A. Siegel in [51].

to obtain the logical error rates for various defect sizes and values of n_{shell}, see Fig. 4.13. Here and after, all simulations assume the phenomenological error model with equal single-qubit and measurement error rate set at 1.5%.

As one can expect, the results for both methodologies exactly match the case when $n_{shell} = 1$. For larger defect sizes, a larger value of n_{shell} results in lower error rates and, therefore, the shell method is favourable. Notice that for each defect width, there is an optimal value of n_{shell} at which the logical error rate is minimised. Going beyond this value implies that the gauge measurement types are alternated with a rate that is too low and, hence, the temporal locations of error string endpoints are poorly resolved.

The next study compares the logical error rate versus the size of the surface code for various defect sizes, see Fig. 4.14. In this demonstration, the optimal values of n_{shell} are used that were found in the previous study. Once such a value is evaluated, we can observe that, for every size of the surface code and every size of the defect, the shell method outperforms Stace's method. It is not at all surprising, since the shell method is an extension of the latter, however, it is still crucial to observe that in both studies the performance difference increases with the defect size. This is consistent with the theoretical guess that Stace's method breaks down at larger defect sizes.

4.5.2 Threshold estimate

In the analytic threshold existence proof for the formalism of shells, I made multiple worst-case assumptions. They were necessary to establish the result rigorously and to make sure that a threshold persists in the asymptotic limit. If one were to use realistic numbers for the constant ρ (c.f. Eq. 4.15) and calculate the analytically proven lower bound of the error rate threshold, that number could easily come out to be 10^{-50}. Of course, this is not a realistic error rate in any quantum platform. For that reason, A. Siegel [51] performed another numerical study to estimate the actual value of the threshold one may expect using the formalism of shells.

Generally, it is very difficult to numerically compare different sizes of the surface codes in the presence of fabrication defects. One could have a model where each element of the surface code is faulty with some given probability. However, assuming that the fabrication error rate is at least as low as 10^{-4}, it is very unlikely that many fabrication defects will cluster near each other for surface code sizes below $L = 25$. Any classical simulations of codes larger than that would require too many resources to be practical, e.g. just decoding a single run might take hours instead of seconds. Therefore, as a way to investigate the performance of the shell method as a solution against clusters of defects, we proposed to use a model in which defects of larger size are manually introduced in the space-time lattice that persist for some time. Such a model is equivalent to a quantum chip being subject to cosmic ray events I discussed in the previous section. Furthermore, since the defect locations are not determined before the start of the computation and instead need to be determined on the fly, our team proposed a new decoding scheme in which the decoder detects a cosmic ray defect from the surge of errors that it causes. After the detection, the region is quarantined from the rest of the code using shells.

The setup for the numerical study is the following. Assuming that the cosmic rays occur uniformly in space and time, we can define a constant σ to represent the probability of a cosmic ray event per qubit per stabiliser round. Very pessimistically

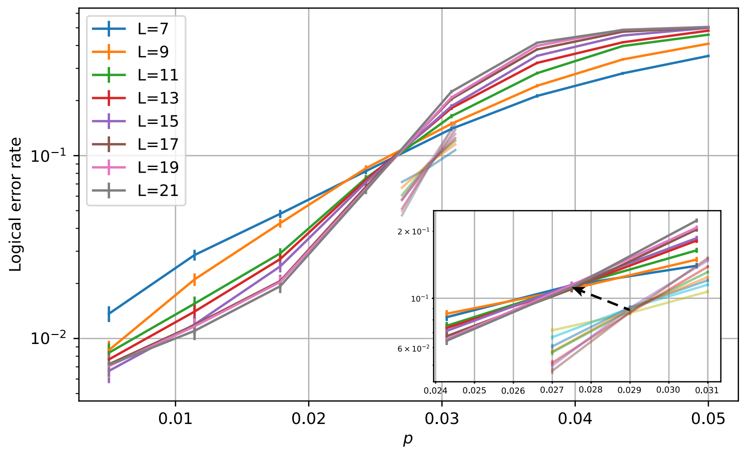


Figure 4.15: Opaque lines: threshold plot for the surface code with defect probability $\sigma = 10^{-5}$ and survival time $T = 100$. Here, the method of shells is used. Faded lines: threshold plot for the non-defective surface code (i.e.$\sigma = 0$). Inset: zoom around the threshold region. The dashed arrow represents the evolution of the threshold when the rate of defects is increased. The figure was produced by my coauthor A. Siegel in [51].

we have assumed $\sigma = 10^{-5}$; for a surface code with $L = 25$, this corresponds to around one defect appearing every 110 measurement rounds. The cosmic ray can strike any qubit of the lattice, and whenever it does it affects a square area of side length 3, i.e. the nearby qubits in a 3×3 area all have their error rate increased to 50%. We set the defect survival time $T = 100$, which means that the affected region is dysfunctional for 100 measurement rounds before we can integrate the qubits back into the code.

Given this setup, one can model the logical error rate using Poisson distribution as

$$\overline{P} = \sum_{k\geq 0} \langle t_k \rangle \times P(\text{logical error}|n_{\text{def}} = k) \tag{4.23}$$

$$\langle t_k \rangle = \frac{\mathrm{e}^{-2L^2\sigma T}(2L^2\sigma T)^k}{k!}, \tag{4.24}$$

where n_{def} denotes the number of simulated defects in the lattice and $\langle t_k \rangle$ denotes the proportion of time that the lattice is affected by k defects. Here, the model assumes that defects are mutually independent and that they can overlap. In [51], A. Siegel numerically estimated $P(\text{logical error}|n_{\text{def}} = k)$ for $k = 0, 1$ and various physical qubit error rates p. Logical error rates for larger k were estimated from other numerical data. Using the variable values estimated above, the logical error rate $\overline{P}$ was plotted as a function of the physical error rate p and a threshold was observed, see Fig. 4.15.

As one can notice, the threshold of the lattice undergoing cosmic ray events only marginally deviates from the ideal case with $\sigma = 0$. Specifically, the threshold for the defect-free case is around 2.9%, while the threshold of the defective case with $\sigma = 10^5$ is around 2.7%. This confirms that the shell method is capable of dealing with catastrophic cosmic ray events at very little cost to the threshold itself. Of course, the scaling of the logical error rate below the threshold will be affected since the code distance is lower during the presence of a time-correlated defect. Observe that at very low p, the logical failure rate is not exponentially vanishing with increasing L. This is due to a logical error floor that is set by the probability that multiple cosmic rays will happen at once and affect the whole lattice. Fortunately, we have used pessimistic assumptions about the prevalence of cosmic rays, and therefore, one can expect in practice such a floor will be at a much smaller value than indicated in Fig. 4.15.

For exact details of the proposed decoder and the numerical study, I guide the reader to the original paper by A. Siegel and three others including myself [51]. Note that in all of the above numerical studies the rotated variant of the surface code is used. While the above proof of the threshold assumed the non-rotated version, the same theoretical arguments fully apply to the rotated case.

4.6 Discussion

In this chapter, I have explored multiple methodologies on how one may be able to combat the harmful effects of fabrication or time-correlated defects on a planar qubit array implementing the surface code. More precisely, I have briefly introduced the theory behind the surface code, discussed Stace's method of super-stabilisers and introduced a new, more general, method of shells - a new syndrome readout protocol for two-dimensional architectures that are now under experimental development. Using the latter method, as the main result, I have shown that one can perform an arbitrarily large quantum computation on a defective lattice with a rapidly vanishing logical failure rate. This represents a substantial improvement upon existing proposals to deal with defects or general time-correlated errors that make use of single-shot error correction [86, 87]. Such codes require three-dimensional architectures that are generally more difficult to manufacture. Therefore the work described in this chapter represents the first provably scalable protocol for dealing with fabrication or other defects on two-dimensional architectures.

Given that in the analytic proof of the threshold I used a number of conservative simplifications and worst-case assumptions, it may be worthwhile to convince oneself that indeed the proposed methodology is useful in practice. For that reason, I have introduced some collaborative work in which the ideas found in this chapter are put into practice. Unsurprisingly, our proposed shell protocol already shows significant performance improvements for small-scale systems with realistic defect rates [51]. Similar results were reproduced in a different numerical study a short time after, which also suggest a highly encouraging performance of the shell formalism [88]. Furthermore, we can expect variations of our protocol or variations of previously suggested methods [64] to perform better in practice when we take the system's architecture into consideration.

To employ the shell protocol it is necessary to establish the locations of fabrication defects or other anomalies such as cosmic ray events. This can be achieved either offline or during the execution of computations by studying the frequency with which the qubit array produces stabiliser events. In contrast, single-shot codes remain functional even if they operate unaware of information about the locations of time-correlated errors [84]. In addition to the detection algorithm already proposed in Section 4.4, it will be interesting to find other, possibly better ways of inferring the locations of time-correlated errors in two-dimensional systems [80–82, 89–91].

Another interesting perspective is to consider the challenges that a finite rate of fabrication defects presents for the task of circuit compilation [74, 92]. At the compilation level, one typically assumes that all logical qubits are performing equally well. This might not be the case for architectures based on two-dimensional quantum arrays that suffer fabrication defects. In practice, there will be regions of the qubit array which have a larger density of fabrication defects. Any logical qubit that is built over such a region will have a higher chance of experiencing a logical error compared to other qubits. One can treat these regions as lower-grade and could allocate a larger portion of the array to form a logical qubit, e.g. increase the distance of all surface code patches in the lower-grade regions. This, however, will lead to a mismatch in code sizes and might create difficulties in performing some of the logical operations required for quantum computation, such as lattice surgery, or create overall tessellation problems. Another solution could be using a circuit compiler that is *hardware-aware* and takes into account the logical qubits with lower fault tolerance. The compiler might allocate such qubits to less error-sensitive tasks or fewer computations in general. Indeed, the compilation of topological circuits with an inhomogeneous array of codes remains an unexplored area of study.

A further route for future research would be to modify the shell proposal for other topological codes. The shell protocol adapts readily to other variations of the surface code with an equivalent boundary theory [93–96]. However, one may also find that generalisations of the shell protocol aimed towards general topological codes [56, 97–99] that have richer boundary theories [100–105], such as recently introduced quantum floquet codes, demonstrate a better performance than the version presented here. This and other future work will determine how best to combat the harmful effects of fabrication defects in modern qubit architectures.

In the next chapter, I will remain within the realm of quantum error correction. However, instead of building practical protocols upon the existing error-correcting schemes, I propose a novel way to construct quantum codes tailored to architectures that are currently being explored. With this, I continue the theme of working towards practical quantum error-correcting systems.

The following chapter is based on an article titled "*Quantum Low-Density Parity-Check Codes for Modular Architectures*" [106]. I was the main author of this work with co-author Lucas Berent. I conceived the main idea behind constructing codes for modular architectures and devised and proved both theorems. I described and generated all example codes that are found within this chapter and the article. Both I and LB generalised the main idea in the penultimate Section 5.6. All authors contributed to the writing of the paper. There was no direct supervision of this project.

Chapter 5

Quantum LDPC Codes for Modular Architectures

In the previous two chapters, I introduced practical advancements of existing QEM and QEC schemes. For the former, I explored a new variant to efficiently learn and mitigate errors, while for the latter, I showed a new methodology to reliably quarantine fabrication defects on a surface code. The following chapter will show a different approach. Instead of further developing existing schemes, I will use the newer ideas in QEC literature to design practical error-correcting codes. Specifically, I will show how one can construct quantum low-density parity-check (LDPC) codes tailored for modular architectures. Such computing designs are envisioned by many current quantum efforts [107–111], and therefore, the topic is very timely.

The chapter is structured as follows. I will first introduce LDPC codes and talk about their construction. Then, I will define what I mean by modular architectures before turning to the main topic of code design. In the later sections, I will generalise some of the main ideas by considering the most recent developments in constructing asymptotically good LDPC codes.

5.1 LDPC codes

Classical LDPC codes were first developed by Robert Gallager in the 1960s [112], albeit largely forgotten until the late 1990s. Nowadays, LDPC codes are used in many technologies such as 5G and Wi-Fi, due to them approaching the Shannon limit

- the maximum rate at which information can be sent through a noisy channel [113]. Formally, low-density parity-check codes are defined as follows.

Definition 5.1. *A code family is called LDPC if for all codes in the family, every parity check has a support on at most k bits and every data bit is involved in at most j checks, where both k and j are constants.*

Note that, just as in the case of code thresholds, the LDPC criterion applies to code families. Nevertheless, we often refer to codes from LDPC families as LDPC codes. As an example, consider the repetition code family from Chapter 2. These codes have parity check matrices of the form

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & \dots \\ 0 & 1 & 1 & 0 & \dots \\ 0 & 0 & 1 & 1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \tag{5.1}$$

where the column and row weight is upper bounded by 2 for any code in the family. Since parity checks correspond to rows and data bits to columns, the statement of bounded column and row weight is equivalent to Def. 5.1, and therefore, the code family is LDPC.

Quantum LDPC codes are defined in the same way but replacing bits with qubits and equating parity-checks with stabiliser generators. Of course, one has the choice to choose the stabiliser generating set, however, if any set satisfies the quantum analogue of Def. 5.1, then we say the code family is LDPC. One may expect the good properties of classical LDPC codes to translate well to the quantum realm. Indeed, as I discussed in Chapter 4, the growing weight of stabiliser checks may lead to a thresholdless code family, while the LDPC criterion ensures us that the stabiliser weights are at most constant. Moreover, it was shown by Daniel Gottesman in 2013 [114] that some quantum LDPC codes can provide fault-tolerant quantum computation with only a constant qubit overhead! See [115] for a good review of quantum LDPC codes.

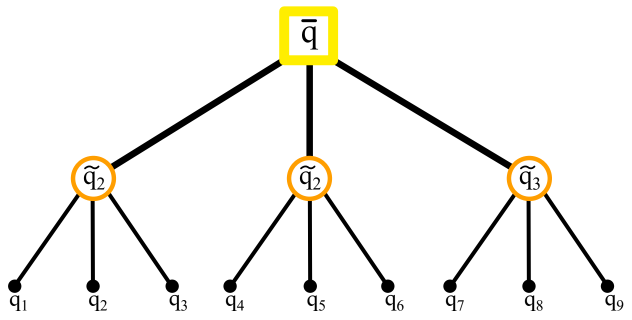


Figure 5.1: 9 qubit Shor code is obtained by concatenating two classical repetition codes. The first layer (lower) encodes each logical qubit $\widetilde{q}_i$ (orange) into a bit-flip repetition code with data qubits q_i (black). The second layer (upper) encodes a single qubit $\bar{q}$ (yellow) into a phase-flip repetition code with data qubits $\widetilde{q}_i$. In turn, $\bar{q}$ is protected from both a single X or Z error. Note, this concatenation graph should not be confused with the Tanner graph.

5.1.1 Quantum code construction

The global theme of this chapter is about devising practical quantum LDPC codes, therefore, let me now introduce some of the relevant ideas behind quantum code design. Most quantum codes are constructed using one of the following methods - tessellating a manifold, concatenating classical/quantum codes or taking their product. Let me briefly comment on all of them in order.

A partial example of the first method was already explored in the last chapter. The surface code is a close relative to the toric code - a topological code that is constructed by square tessellating a torus. The surface code is obtained by appropriately cutting the torus open and defining the boundary stabilisers. As I discussed in Chapter 2, to construct a quantum code, the elements of a tessellation, i.e. faces, edges and vertices, are identified with X/Z stabilisers and data qubits in some order. Other examples include hyperbolic surface codes that are obtained by tessellating hyperbolic manifolds [116] and 2D colour codes constructed from trivalent lattices whose faces are tricolourable [117]. Unfortunately, code families based on low dimensional manifolds have code parameters whose scaling is severely bounded, and therefore, they do not offer the best overheads for scalable QEC [118, 119].

Code concatenation is based on a simple principle where logical qubits of one code are encoded as data qubits of another code. Each concatenation layer decreases the probability of logical failure given the physical error rate is below some threshold. While code concatenation is mainly done with smaller quantum codes, interestingly, in some instances, one can concatenate two classical codes to design a quantum code. An example of this is a 9 qubit Shor code [17], where three $d = 3$ repetition codes are concatenated with a single $d = 3$ repetition code. See Fig. 5.1. The code is designed in a way such that the first layer provides protection against bit-flip (X) errors, while the second layer against phase-flip (Z) errors. In total, the code can correct an arbitrary single error and encodes $k = 1$ logical qubit. The main drawback of constructing quantum codes in this way is the increased weight of stabiliser supports after each concatenation. For example, the stabilisers in the first layer of the Shor code have supports of weight 2, while those in the second layer have weight 6. The reason is simple, the higher level stabilisers have to check the parity of the lower-level logical qubits that already consist of multiple data qubits. Therefore, the concatenated code family is generally not LDPC if it contains arbitrary long concatenations. Moreover, large codes constructed in this way often require “long-range” qubit connectivity that is hard to achieve in practice.

The final code construction method, which recently has gained a lot of attention, involves taking products of *seed* codes. First introduced by Jean-Pierre Tillich and Gilles Zémor in [120], the simplest case is to construct a quantum CSS code as a product of two classical codes. The resulting code from the construction given in [120] is called a hypergraph product code to emphasise that the product is taken over hypergraphs representing binary linear codes. Such a representation of classical codes is analogue to the tessellation representation of quantum codes. Here, vertices correspond to parity-checks and hyperedges (which can also be simple edges) to bits. Fig. 5.2 shows hypergraphs representing an $n = 3$ repetition code and its transposed code, and Fig. 5.3 shows how a surface code is constructed as a product of these two hypergraphs. Note that in the rest of the chapter, most examples will use hypergraphs that are also graphs for ease of visualisation.

The general idea of hypergraph products is similar to the specific concatenated method where a combination of two classical codes is used to provide protection

a) $H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$ $H^T = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$

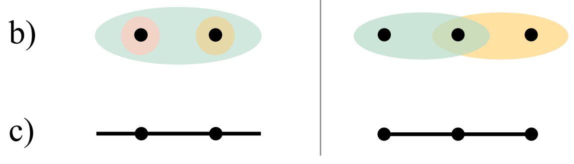


Figure 5.2: Hypergraph representations of two classical codes - a three-bit repetition code (left) and its transposed code (right). Their parity check matrices are given in (a), hypergraphs in (b) and a more conventional way of drawing hypergraphs in (c).

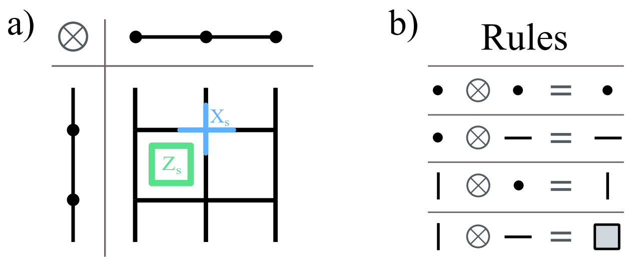


Figure 5.3: A distance $d = 3$ surface code can be constructed as a hypergraph product of a three-bit repetition code and its transpose. The product table is given in (a) and its rules in (b).

against both, X and Z errors while ensuring that stabilisers commute. Importantly, if the seed codes are LDPC, then the resulting quantum code is also LDPC. Later, I will expand on these comments and formalise the construction with the language of homological algebra and chain complexes.

5.1.2 Asymptotically good quantum codes

Unfortunately, similarly to the case of topological codes, the hypergraph product codes have a bounded code parameter scaling [120]. Specifically, given two classical $[n, k, d]$ seed codes with $[m, \widetilde{k}, \widetilde{d}]$ code as their transpose, the hypergraph product code has parameters $[\![n^2 + m^2, k^2 + \widetilde{k}^2, \min(d, \widetilde{d})]\!]$. This means that a quantum code family constructed as a hypergraph product of any two classical code families cannot be *good* in the following sense.

Definition 5.2. *A code family is called "good", if the parameters* $[\![n, k, d]\!]$ *of the codes in the family scale as* $[\![n, \Theta(n), \Theta(n)]\!]$.

The codes in these good families are sometimes called asymptotically good quantum codes. It is clear that hypergraph product code family cannot be good, for example, if one constructs it from two families of good classical codes then its parameters will scale as $[\![n, k = O(n), d = O(\sqrt{n})]\!]$. In terms of scaling, good classical codes are optimal, and therefore, this example sets the upper bound on the parameters of any hypergraph product code.

Recently, novel code constructions that generalise the hypergraph product were shown to obtain better parameters. The first work to break the $\sqrt{n}$ distance barrier (up to a polylog) was by M. Hastings, J. Haah and R. O'Donnell in 2020 [121] in which they introduced fiber bundle codes. Soon after, works introducing lifted product [122, 123], balanced product [124] and quantum tanner codes [125] appeared that each provided a different perspective on how to construct quantum codes from classical codes. In some limits, the different formalisms coincide. All of this development culminated in a groundbreaking work by P. Panteleev and G. Kalachev in 2021 [123] in which they proved that asymptotically good quantum LDPC codes exist. While the authors used the lifted product code construction in their proof, the same results apply to any of the novel schemes.

One may assume quantum LDPC codes will lower the necessary qubit overhead for fault-tolerant quantum computation. However, it is known that well-performing codes require long-range qubit connectivity to embed them in some finite-dimensional Euclidean space [126]. Therefore, even with better parameters, one should ask whether LDPC codes are practical enough to favour them over the surface code - the current gold standard for many platforms. In this chapter, I take a step towards answering this question by providing a novel way to perceive and construct quantum LDPC codes tailored to modular architectures.

5.2 Modular architectures

In classical computing it has become standard to design architectures that divide the necessary processing power into smaller components instead of only increasing the power of a single system [127, 128]. A similar trend is observed in recent proposals around scaling quantum computation. Many quantum computing platforms have natural limitations, for example, the number of qubits that may be contained within a single ion trap or a superconducting chip, where each instance of such a unit I refer to as *a module* [107–111]. Since scaling up existing systems is crucial for large-scale computation, *modular architectures* that consist of many similar modules will likely be necessary [108, 129–133].

When considering the setting of modular architectures, one may expect that some degree of long-range qubit interactions that scale with the code size is feasible [134–137]. These can either be long-range interactions within each module or between the modules themselves, and one may construct quantum LDPC codes that make use of them. Even then, it is not clear whether quantum LDPC codes provide the best solution towards practical quantum computation. All aspects of fault-tolerant quantum computation need to be considered, such as the implementation of fault-tolerant logic, decoding, and the code performance, while most of these topics are still in the infancy for quantum LDPC codes. Previous works have mentioned the compatibility of quantum LDPC codes and modular architectures, but without providing the exact details for the code construction or the partition of the qubits into modules [138]. Alternatively, past works have in detail described the way to use a surface code for modular architectures [139], but do not consider general quantum LDPC codes.

In this chapter, I will further explore LDPC code construction for modular architectures. In a formal and general way, I will show how quantum LDPC codes can be *tailored* to modular architecture connectivity constraints. Specifically, this work will expose a correspondence between product constructions of quantum LDPC codes and modular architectures by viewing the intra- and inter-modular connectivities as Tanner graphs of classical or quantum LDPC codes. Before proceeding to the main results, let me first formalise the definitions concerning modular architectures and introduce the language of chain complexes.

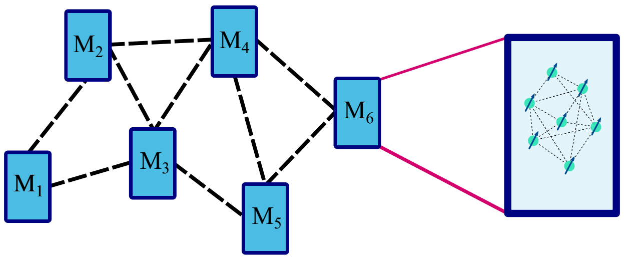


Figure 5.4: In this chapter, I consider quantum computing architectures where modules $\{M_k\}$ have a defined sparse inter-modular connectivity. Each module contains a finite and equal number of qubits with the same connectivity constraints.

5.2.1 Formalism

In this chapter, I will consider various qubit connectivity constraints that may arise in a quantum computer based on a modular architecture. Taking the constraints into account, I will provide a novel recipe on how to construct quantum LDPC codes that respect a certain modular architecture.

Let me first concretely define what I mean by a modular architecture. Consider a quantum computer with a (finite) collection of qubits $\{q_N\}$. In a modular architecture, each qubit is assigned to one and only one module, where the (finite) collection of such modules is given as $\{M_k\}$, see Fig. 5.4. I assume that the modules are equivalent copies of each other. Therefore, one can partition the collection of qubits into disjoint sets $\{q_i\}_k$ such that $\{q_N\} = \bigcup_k \{q_i\}_k$, and I require that each module contains a finite and equal number of qubits n, i.e. $|\{q_i\}_k| = n$ for all k. To simplify the notation, I will use the same canonically ordered index set for each module and hence drop the subscript k altogether. With this in mind, the intra-modular qubit connectivity is defined in the usual sense as follows

Definition 5.3. *A qubit $q_i \in M_k$ is connected to a qubit $q_j \in M_k$ if the architecture allows us to directly implement two-qubit entangling operations between these qubits for all k.*

For most part, I consider entangling gates such as controlled-not (CNOT) which are required for most syndrome circuits. However, entangling operations using measurements, for example, in using photonic links [107], or otherwise are just as valid. The only requirement is that these gates allow one to construct a syndrome extraction circuit required to perform quantum error correction.

In a modular architecture, each module may be connected to some number of other modules in a specific way. Let me define the inter-modular connectivity as follows

Definition 5.4. *A module M_k is connected to a module M_j if the architecture allows us to directly implement two-qubit entangling operations between a qubit $q_i \in M_k$ and its respective qubit $q_i \in M_j$ for all i.*

In Section 5.6.1, this requirement will be slightly generalised to allow for *twists* of the connections between modules to construct better quantum codes.

Both inter-modular and intra-modular connectivity graphs will be shown to have a one-to-one correspondence to the Tanner graphs of potentially different quantum or classical codes. Finally, one says that the code *respects* the connectivity constraints if each physical qubit of the system can be associated to a parity check or a data qubit of the code such that parity checks qubits are connected to the data qubits in their support.

In the following, I will describe a way to create new codes that respect the overall architectural connectivity constraints as defined above. To keep this framework as general as possible, I employ the language of chain complexes.

5.3 Chain complex representation of codes

In this section, I will discuss on how to view quantum codes through an ($\mathbb{F}_2$) homological perspective [115]. Before expressing codes in terms of chain complexes (which are algebraic structures in homological algebra and elsewhere), let me start by introducing the binary representation of quantum CSS codes.

5.3.1 Binary representation of quantum codes

Since quantum LDPC codes are often constructed using classical codes, it is worthwhile to reformulate some of the background and definitions of quantum codes with the equivalent binary language.

Each n qubit Pauli operator can be written as a binary vector. More specifically, the elements of the Pauli group, i.e., n-fold tensor products of I, X, Y and Z Pauli matrices, have a one-to-one mapping to binary vectors $(x|z)^T \in \mathbb{F}_2^{2n}$ up to some irrelevant phase. The correspondence is as follows. The bit with index $i \in [0, n]$ in the binary string $x \in \mathbb{F}_2^n$ has value 1 iff the Pauli operator $P \in \mathcal{P}_n$ has a Pauli matrix X or Y in the ith position of its n-fold product. The $z \in \mathbb{F}_2^n$ case is defined in the same way but with Z instead of X. Therefore, $P, Q \in \mathcal{P}_n$ commute iff for their binary representations $P \cong (x|z)^T, Q \cong (x'|z')^T$ it holds that

$$\langle x, z' \rangle + \langle z, x' \rangle = 0. \tag{5.2}$$

Given a stabiliser group based on Pauli operators, this representation naturally applies to all m stabiliser generators $S_1, \dots S_m$ of some quantum code $\mathcal{C}$. In turn, one can define a quantum parity check matrix $H = (H_X \mid H_Z)$ with dimensions $m \times 2n$, where ith row of H corresponds to the binary representation of a stabiliser generator S_i.

This representation allows one to work with CSS codes in a simpler way. Recall that they are defined as codes where the non-identity operators for each stabiliser generator are either all X or all Z single-qubit Pauli operators. Therefore, the non-trivial support of each stabiliser generator either belongs to H_X or H_Z, that is, the matrices H_X and H_Z define the X and Z stabilisers (parity checks) respectively. Moreover, the commutativity relation necessary for stabiliser codes can be rewritten as

$$H_X H_Z^T = 0. \tag{5.3}$$

Now I am ready to introduce an alternative perspective on codes that was essential in the recent theoretical developments and has become standard.

5.3.2 Chain complexes

In general, a chain complex is an algebraic structure of modules (the algebraic kind) with particular homomorphisms between them. For the purposes of this chapter, I will only consider modules that are vector spaces. In this case, a chain complex is a collection of vector spaces $\{C_i\}$ together with linear maps (homomorphisms) called boundary maps

$$\partial_i : C_i \to C_{i-1}, \tag{5.4}$$

that satisfy the following condition

$$\partial_i \partial_{i+1} = 0. \tag{5.5}$$

This is equivalent to requiring that im $\partial_{i+1} \subseteq$ ker ∂_i. Elements in C_i are called *i-chains* and

$$\begin{aligned} Z_i(C) &= \ker \partial_i \subset C_i \\ B_i(C) &= \operatorname{im} \partial_{i+1} \subset C_i \\ H_i(C) &= Z_i(C)/B_i(C) \end{aligned}$$

are the *i-cycles*, *i-boundaries* and the *i*th *homology* of the chain complex C respectively. For instance, when using the chain complex formalism to describe a tessellation of some two-dimensional surface, 2-chains correspond to a linear combination of faces over $\mathbb{F}_2$, 1-chains to a combination of edges and 0-chains to a combination of vertices. In this example, 1-cycles are a linear combination of closed walks, while 1-boundaries are those cycles that are boundaries of a set of faces.

A classical binary linear code $\mathcal{C}$ can be represented as a two-term chain complex

$$C = C_1 \xrightarrow{\partial_1} C_0, \tag{5.6}$$

where $C_1 = \mathbb{F}_2^n$ and $\partial_1 = H$ is the parity check matrix. Then the codespace $\mathcal{C}$ is the space of 1-cycles

$$\mathcal{C} = Z_1(C) = \ker \partial_1, \tag{5.7}$$

and the space C_0 is the space of code syndrome, where intuitively, one can associate the basis elements of C_0 to parity checks. Note that the homology $H_0(C) = 0$ if all parity-checks of H are linearly independent. For classical codes, the chain complex representation does not provide any new insights and hence is rarely used. However, a quantum CSS code necessitates commutation relations between the X and Z parity check matrices H_X and H_Z, Eq. 5.3. Therefore, it has become common to represent a quantum CSS code with a three-term chain complex

$$C = C_{i+1} \xrightarrow{\partial_{i+1}} C_i \xrightarrow{\partial_i} C_{i-1}, \tag{5.8}$$

where $\partial_{i+1} = H_Z^T$ and $\partial_i = H_X$. Here, the boundary condition given in Eq. 5.5 ensures that $H_X H_Z^T = 0$ as needed. Qubits are associated with 1-chains and X and Z checks with 0-and 2-chains, respectively. A prototypical example is a toric code, where C_2, C_1 and C_0 are vector spaces of *faces*, *edges* and *vertices*, obtained from a square tessellation of a torus. Conversely, given an arbitrary chain complex

$$... \to C_{i+1} \xrightarrow{\partial_{i+1}} C_i \xrightarrow{\partial_i} C_{i-1} \to ... \tag{5.9}$$

one can pick a vector space of dimension i to be the space of qubits and view the corresponding three-term chain complex as a CSS code. In this formalism, the code parameters are $n = \dim C_i$, $k = \dim H_i$ and d is the minimum weight of a non-trivial representative in H_i or the respective ith cohomology H^i. The ith cohomology is defined in a similar way to the ith homology but using a chain complex with “arrows” pointing the other direction. Therefore, the non-trivial elements of H_i correspond to logical Z operators, while the non-trivial elements of H^i to logical X operators.

Furthermore, to a three-term chain complex one can associate a quantum Tanner graph by considering a simple mapping. A quantum Tanner graph G is a tripartite graph with a vertex set $V = P_Z \sqcup Q \sqcup P_X$, where the vertices Q are associated with qubits and vertices $P_{X(Z)}$ are associated with $X(Z)$ parity checks, see Subsection 2.2.2. Then, for an arbitrary three-term chain complex C, I define a one-to-one mapping where the basis elements of the vector space C_{i+1} are mapped to vertices in P_Z, the basis elements of C_i to Q, and the basis elements of C_{i-1} to

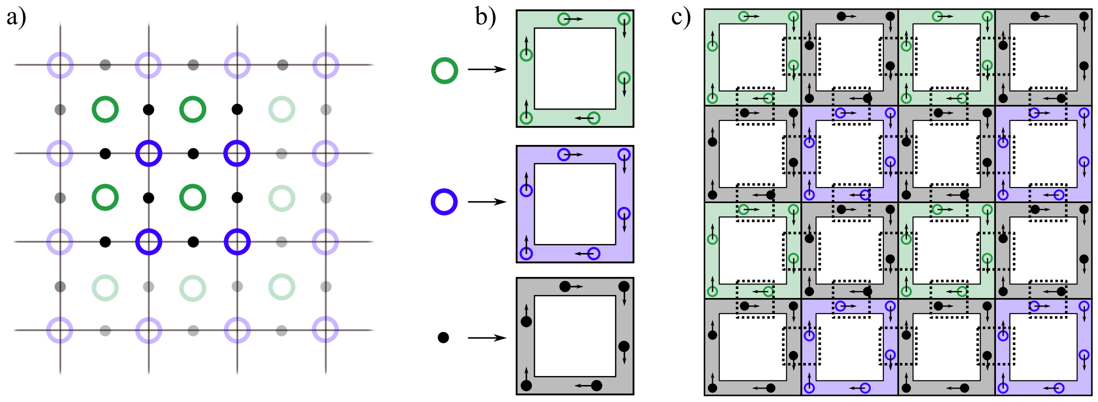


Figure 5.5: A stack of 2D surface codes can be generated by replacing every data qubit (black), Z parity check qubit (green) and X parity check qubit (blue) of the regular surface code (a) by a loop of corresponding qubits (b). The qubits communicate (entangle) with each other once they enter the dashed regions depicted in (c).

P_X. Finally, the edges between vertex partitions P_Z and Q are given by ∂_{i+1} such that an edge exist between vertices $p \in P_Z$ and $q \in Q$ if and only if $\overline{q} \in C_i$, which is the corresponding basis element of q, is in the span of $\partial_{i+1}\overline{p}$, where $\overline{p} \in C_{i+1}$ and corresponds to the vertex p. Edges between P_X and Q are defined in an analogous manner using ∂_i. Note that there are no edges between any two vertices within the same partition or between partitions P_X and P_Z. Graphically, if the chain complex is drawn as an object with faces, edges and vertices, then each individual face and edge is replaced by a vertex. The partition of vertices is naturally induced from this mapping and edges exist only between those pairs of vertices that correspond to adjacent faces and edges (or vertices and edges) in the original chain complex.

Conceptually the same mapping to a graph can be applied to an arbitrary-length chain complex. For example, a two-term chain complex would be mapped to a classical Tanner graph. Because of this bijection, I will refer to chain complexes, their boundaries or their corresponding Tanner graphs (sometimes called connectivity graphs) interchangeably for the rest of this chapter.

5.4 Looped pipeline architecture

To form a better intuition about the construction of codes for modular architectures that I am about to present, it is beneficial to recap a recent result about performing

QEC on a looped pipeline architecture [140]. This will constitute the starting point from which I will build the framework to design high-performing quantum codes.

The work by Z. Cai and others considers a surface code layout (Fig. 5.5(a)) of a quantum chip where every data and parity check qubit is replaced by a rectangular loop of fixed number of qubits as shown in Fig. 5.5(b). All types of qubits are moved along the loop in the clockwise direction at the same frequency. Once the qubits approach another qubit from a different loop, they interact to become entangled in a way that corresponds to the syndrome extraction circuit as illustrated in Fig. 5.5(c). After the parity check qubits have gone around the full loop, they are measured to read out the syndrome. The authors showed that this forms a stack of 2D surface codes, where the stack size is given by the number of qubits within each loop.

In their work, the authors and independently M. Fogarty [141] identified that the stack of 2D surface codes may be used to generate a 3D surface code, but did not provide an explicit construction. In the rest of this section, I will show how the looped pipeline architecture can be extended to implement a single 3D surface code by assuming an additional connectivity within each qubit loop. Finally, I will formalise and generalise this construction to the setting of modular architectures. This will allow one to construct quantum LDPC codes for more general connectivity constraints.

5.4.1 3D surface code

Consider a single data qubit loop in the looped pipeline architecture. Here, each qubit in the loop is part of a separate 2D surface code, however, if one extends the connectivity between the nearby qubits then the stack of surface codes is linked together into a single 3D block. Note that this construction does not immediately allow one to produce a 3D surface code, for example, the data qubits are linked to other data qubits and that is not necessary for any code.

Instead, to produce a valid 3D surface code one additionally needs to re-identify the qubits within each loop. In this regard, three different types of loops are formed - face, edge and vertex loops - where the naming draws parallels to the tessellation

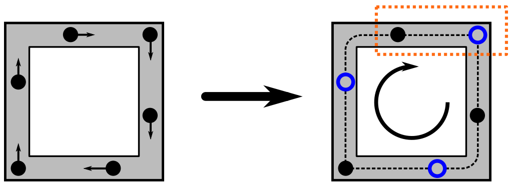


Figure 5.6: All loops are replaced by a loop that has both parity check (blue) and data (black) qubits. Additionally, one may allow for nearby qubits in the loop to communicate, for example, the nearby qubits entangle whenever both of them enter the orange dashed box.

Loop label	Odd qubits	Even qubits
Vertex	X-stab (6)	Data
Edge	Data	Z-stab (4)
Face	Z-stab (4)	—

Table 5.1: Qubit assignments of different loops to generate a 3D surface code. Even/Odd assignment of qubits indicate their position in the qubit chain within the loop. Numbers in the parenthesis indicate the weight of the stabiliser.

representing a surface code. That is, the loops are laid out in a pattern such that face loops replace the Z parity check loops, edge loops the data qubit loops and vertex loops the X parity check loops. Each of these loops may contain both data and parity check qubits of the 3D surface code, hence, they all must contain measurement devices as described in [140] to extract the code syndrome Fig. 5.6. Let me assume that the total number of qubits per loop is n which is even. Then, depending on the type of the loop and position within the loop each physical qubit can be given an assignment. These assignments are listed in Table 5.1. The even/odd parity of the qubit corresponds to its index $i \in [1, n]$ within the loop. Here, indexing is such that a qubit with index i interacts with qubits $i+1 \mod n$ and $i-1 \mod n$ within the same loop and the same index qubits in the neighbouring loops. Note that one can exclude the even qubits in the face loop as they have no assignment.

By viewing the modular architecture in terms of chain complexes that describe codes, I will prove that such a qubit assignment indeed produces a 3D surface code. To do so, I need to introduce further background in the chain complex formalism.

Remark. The presented construction of a 3D surface code by shuttling the qubits around in loops will not have a threshold in general. Since the entangling and measurement operations are done on a one-by-one basis, the time it takes to extract syndrome is proportional to the number of qubits within each loop. A slightly altered scheme was proposed by [140], where the qubits in adjacent loops circulate in opposite directions and all qubits on the same side of the loop are being entangled at the same time with their respective qubits in other loops. If this was supplied with a number of measurement devices that scale in proportion with the number of qubits in the loop, the whole syndrome extraction would require $O(1)$ operations (e.g., one operation constitutes moving qubits to a new side of the loop). However, from a physical perspective, one could argue that, in general, as the number of qubits per loop increases, so does the size of the loop that is needed. Therefore, the physical time it takes for a qubit to be shuttled at a finite speed to a new side of the loop (to perform a single operation) scales with the number of qubits per loop and, hence, the syndrome extraction time scales with the total number of qubits. In turn, errors accumulate faster than they can be corrected and the threshold for the code does not exist. This might be a general consequence for any non-concatenated quantum code for which the shuttling is used to do many long-range entangling gates locally (the same conclusion was reached by numerical analysis for a conceptually similar setup in [142]). Some ideas, like the ones presented in recent work on hierarchical memories [143], may be used to overcome this challenge. Note that the aforementioned drawback only applies to this specific scheme based on shuttling. The main ideas in the rest of the paper do not necessarily share the same consequences.

5.4.2 Tensor product of chain complexes

Recall that in Subsection 5.1.1 I described a method of constructing quantum codes by taking products of other (not necessarily classical) codes [120, 144]. This construction can be succinctly expressed using the formalism of chain complexes. For that, let me introduce the tensor product of chain complexes as follows.

Given two chain complexes C and D, one can define a *double complex* $C \boxtimes D$ as

$$(C \boxtimes D)_{p,q} = C_p \otimes D_q. \tag{5.10}$$

$$\begin{tikzcd}
C_1 \otimes D_1 \arrow[r, "id^C \otimes \partial_1^D"] \arrow[d, "\partial_1^C \otimes id^D"] & C_1 \otimes D_0 \arrow[d, "\partial_1^C \otimes id^D"] \\
C_0 \otimes D_1 \arrow[r, "id^C \otimes \partial_1^D"] & C_0 \otimes D_0
\end{tikzcd}$$

Figure 5.7: A commuting diagram representing a double chain complex of two 2-term chain complexes C and D.

with vertical boundary maps $\partial_i^v = \partial_i^C \otimes \mathrm{id}^D$ and horizontal boundary maps $\partial_i^h = \mathrm{id}^C \otimes \partial_i^D$ such that $\partial_i^v \partial_{i+1}^v = 0$, $\partial_i^h \partial_{i+1}^h = 0$, and $\partial_i^v \partial_j^h = \partial_j^h \partial_i^v$. Here, ∂_i^A denotes the usual boundary map, while id^A denotes an identity map acting on chain complex A. An example double complex where C, D are both 2-term chain complexes is shown in Fig. 5.7.

Furthermore, a *total complex* is constructed by merging the vector spaces of equal dimensions. That is, one "sums over the anti-diagonals" in the double complex as follows

$$\mathrm{Tot}(C \boxtimes D)_n = \bigoplus_{p+q=n} C_p \otimes D_q = E_n \tag{5.11}$$

where the boundary maps are $\partial^E = \partial^v + \partial^h$. Finally, the tensor product of two chain complexes is defined as the total complex of their double complex, i.e., $C \otimes D = \mathrm{Tot}(C \boxtimes D)$. For example, the tensor product complex of Fig. 5.7 is

$$C_1 \otimes D_1 \xrightarrow{\partial_2} C_0 \otimes D_1 \oplus C_1 \otimes D_0 \xrightarrow{\partial_1} C_0 \otimes D_0, \tag{5.12}$$

where

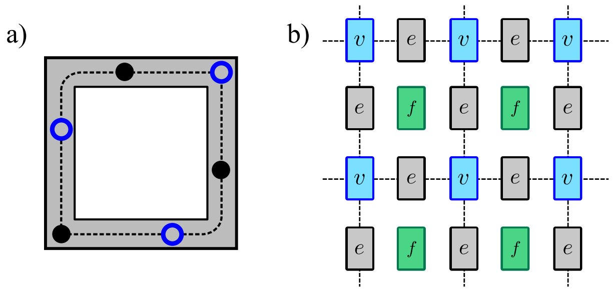
$$\partial_2 = \begin{pmatrix} \partial_1^C \otimes id^D \\ id^C \otimes \partial_1^D \end{pmatrix}, \tag{5.13}$$

$$\partial_1 = \left(id^C \otimes \partial_1^D \mid \partial_1^C \otimes id^D \right). \tag{5.14}$$

Since the homology of a chain complex is related to the parameters of the code it describes, the Künneth formula for calculating the homology of tensor product complexes is crucial. The formula is given as

$$H_n(C \otimes D) \cong \bigoplus_{p+q=n} H_p(C) \otimes H_q(D), \tag{5.15}$$

where the homologies of individual complexes in the product are used.



$$C = C_1 \xrightarrow{\partial_1} C_0 \qquad D = D_2 \xrightarrow{\partial_2} D_1 \xrightarrow{\partial_1} D_0$$

Figure 5.8: A single loop corresponds to a two-term chain complex and a layout of loops to a three-term chain complex representing a repetition code (a) and a surface code (b) respectively.

5.4.3 3D surface code in the chain complex formalism

While the previous construction of a 3D surface code may seem arbitrary at first, we can naturally describe it in the language of chain complexes. This description allows one to further generalise the construction of codes for modular architectures and gives a strong intuition for which codes may or may not be constructed given some connectivity constraints.

First, consider a single loop of qubits with nearest neighbour connectivity between them as described in Subsection 5.4.1. It is natural to view the loop as a classical repetition code where half of the qubits are assigned to be data qubits and the other half parity check qubits as depicted in Fig. 5.6. As mentioned in Section 5.3, a two-term chain complex $C = C_1 \xrightarrow{H_X} C_0$ can be used to describe the repetition code, where the data and parity check qubits are elements (chains) of C_1 and C_0 respectively and H_X is the parity check matrix of the code, see Fig. 5.8(a).

Moreover, the two-dimensional layout of loops can be considered as a two-term chain complex $D = D_2 \xrightarrow{H_Z^T} D_1 \xrightarrow{H_X} D_0$ representing a 2D surface code. In this language, each loop is labeled as an element of either D_2, D_1 or D_0, which represent Z parity checks (faces), data qubits (edges) or X parity checks (vertices) respectively, see 5.8(b).

With this perspective, it is easier to prove that the layout of loops and qubit assignments given in Subsection 5.4.1 describe a 3D surface code $\mathcal{E}$. To show this,

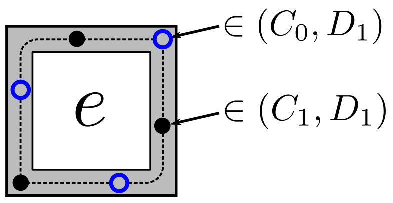


Figure 5.9: With the presented perspective, one can readily identify which vector spaces every qubit is associated with. The parity check qubits of the repetition code are elements (chains) of C_0, while the data qubits are chains of C_1. The loop itself has an assignment as an edge loop, therefore, every qubit in it belongs to D_1.

I want to argue that $\mathcal{E}$ is represented by a chain complex

$$E = C \otimes D, \tag{5.16}$$

where C corresponds to the code within the loop (the repetition code) and D corresponds to the code describing the layout of the loops (the surface code).

For this, consider each qubit (data or parity check) in our system to be associated with two vector spaces - one from the chain complex C and one from D. The set of spaces C_i describes whether the qubit in the loop is a parity check $i = 0$ or data $i = 1$ qubit, while the set D_j describes whether the qubit belongs to the face $j = 2$, edge $j = 1$ or vertex $j = 0$ loop. See Fig. 5.9 for a schematic explanation of qubits in the edge loop. It is easy to verify that the boundaries of the chain complexes are consistent with the qubit interactions described in Subsection 5.4.1. That is, if any qubit corresponding to an i-chain is in the boundary ∂_{i+1} of another qubit corresponding to an $i + 1$-chain, then both of these qubits can interact. Simply stated, the boundaries of C and D define the qubit interactions. Finally, we assign each qubit to a vector space E_k, with $k = i + j$, which collectively form the sequence of vector spaces for a new four-term chain complex

$$E = E_3 \xrightarrow{\partial_3} E_2 \xrightarrow{\partial_2} E_1 \xrightarrow{\partial_1} E_0, \tag{5.17}$$

where

$$\begin{aligned}
E_3 &= C_1 \otimes D_2, \\
E_2 &= C_0 \otimes D_2 \oplus C_1 \otimes D_1, \\
E_1 &= C_0 \otimes D_1 \oplus C_1 \otimes D_0, \\
E_0 &= C_0 \otimes D_0
\end{aligned}$$

and where the boundary maps are inherited from the chain complexes C and D as given in Subsection 5.4.2. Therefore, the resulting chain complex E is a tensor product of chain complexes representing a repetition code and a surface code which is known to represent a 3D surface code [144, 145].

In this construction, one could identify the data qubits with vector space E_1 and Z (X) stabilisers with spaces E_2 (E_0) respectively. Then, parity check matrices are given as boundary operators $\partial_2 = H_Z^T$ and $\partial_1 = H_X$. Note that, since the chain complex C describes a periodic repetition code, one of the boundaries of the 3D surface code is periodic. I want to remark that one is free to identify data qubits with either the vector space E_2 or E_1, which corresponds to the choice of having the logical X (Z) operators to be planar (string)-like on the 3D surface or the other way around.

As an example, consider an $L \times L$ surface code as the layout of the loops. It has parameters $[\![2(L^2 - L) + 1, 1, L]\!]$. The respective chain complex D has homology $H_1(D) \cong \mathbb{Z}_2$ and $H_0(D)$ is trivial. Similarly, consider a length L classical repetition code inside the loop with parameters $[L, 1, L]$. The respective chain complex $\mathcal{C}$ has homology $H_1(C) \cong \mathbb{Z}_2$ and $H_0(C) \cong \mathbb{Z}_2$ since one of the checks is linearly dependent. Then, by identifying the basis elements of E_1 as data qubits, one constructs a 3D surface code $E = C \otimes D$ with the number of data qubits $n = \mathrm{dim}E_1 = \dim(C_1 \otimes D_0 \oplus C_0 \otimes D_1) = L \cdot (L^2 - L) + L \cdot (2(L^2 - L) + 1)$, where D_0 and C_0 are vector spaces associated with X parity checks. The number of encoded qubits is given by the dimension of the first homology $H_1(E)$. Recall, that one can compute it using the Künneth formula (Eq. 5.15) as

$$k = \mathrm{dim}H_1(E) = \dim(H_0(C) \otimes H_1(D) \oplus H_1(C) \otimes H_0(D)) = 1. \tag{5.18}$$

The distance of the code is still L, hence, the resulting 3D surface code has parameters $[\![3L(L^2-L)+L,1,L]\!]$. Using $L=20$ the code parameters are $[\![22820,1,20]\!]$.

Note that in this construction qubits that were previously data qubits may be reassigned to parity check qubits and vice versa. More importantly, the intra- and inter-modular connectivity requirements of $\mathcal{E}$ correspond to the connectivity requirements given by codes $\mathcal{C}$ and $\mathcal{D}$. Let me prove this statement for general tensor products of chain complexes representing codes.

Theorem 5.1. *Let C and D be a two- or three-term chain complexes representing classical or quantum codes $\mathcal{C}$, $\mathcal{D}$ respectively. Let their boundaries ∂^C and ∂^D define the intra- and inter-modular connectivity respectively. Then a quantum code $\mathcal{E}$ corresponding to the chain complex $E = C \otimes D$ respects the connectivity constraints of the architecture.*

Proof. The chain complex E (corresponding to $\mathcal{E}$) is at least a three-term chain complex given that both C and D are at least two-term chain complexes. Therefore, one can identify chains of E_i with data qubits, E_{i+1} with Z parity checks and E_{i-1} with X parity checks. The required qubit connectivity of $\mathcal{E}$ is defined by its parity check matrices, which are given by the boundary operators

$$\partial^E = \partial^h + \partial^v = \mathrm{id}^C \otimes \partial^D + \partial^C \otimes \mathrm{id}^D. \tag{5.19}$$

Note that basis i-chains of C label the qubit c within each module and that basis i-chains of D label each module d. We can ensure that $\mathcal{E}$ respects the connectivity constraints of the architecture if both terms of Eq. 5.19 match the given qubit connectivity. Let me consider both of the terms separately.

The basis elements of E are pairs of chains $(c,d) \in C \times D$ on which the first term acts as $\partial^h : (c,d) \to (\mathrm{id}^C(c), \partial^D(d))\ \forall c,d$. By linear extension, this defines a map on $C \otimes D$. It is equally stated that *each* qubit c in a module d is connected to its *respective* qubits c in the adjacent modules given by $\partial^D\ \forall d$. This matches Def. 5.4 describing the inter-modular connectivity. Similarly, the second term in Eq. 5.19 defines a map that acts on the basis elements as $\partial^v : (c,d) \to (\partial^C(c), \mathrm{id}^D(d))\ \forall c,d$.

This is equally stated that in *each* module d a qubit c is connected to qubits $\partial^C(c)$ $\forall c$. This matches Def. 5.3 describing intra-modular connectivity. Therefore, the required qubit connectivity of $\mathcal{E}$ is given by some additive combination of terms which define the intra- and inter-modular connectivity respectively. $\square$

Furthermore, it is clear that the Theorem 5.1 still applies whenever boundaries ∂^C and ∂^D define any subgraphs of intra- and inter-modular connectivity respectively. If the subgraphs are proper then some connectivity that is allowed by the architecture is not required.

The correspondence between modular architectures and quantum codes obtained from product constructions is very natural and gives an intuitive way of designing codes that obey architectural connectivity constraints. In the following section, I will gradually generalise the 3D surface code construction to more exotic codes by considering richer and less local connectivity between qubits.

5.5 Hypergraph product codes

The proposed formalism of the looped pipeline architecture allows one to construct a 3D surface code as a tensor product of two codes - a repetition code describing the qubit connectivity within each loop and a surface code describing the connectivity between the loops themselves. As shown in Theorem 5.1, the same perspective can be applied to the construction of codes for architectures in which long-range qubit connectivity is possible. In this section, I will extend the 3D surface code construction by considering modular architectures with arbitrary intra-modular and inter-modular connectivity.

5.5.1 Generalised intra-modular connectivity

The first generalisation of the above architecture is to replace the loops of qubits with modules admitting more sophisticated intra-modular connectivity. Similarly to viewing the loops of qubits as repetition codes, one may perceive this connectivity as a chain complex corresponding to some code $\mathcal{C}$. Note that for better codes in general, a higher degree of intra-modular connectivity is needed.

Formally, consider a tensor product $E = C \otimes D$ of a chain complex C corresponding to some not necessarily simple classical or quantum code and a three-term chain complex D corresponding to a surface code which, as before, describes the overall layout of modules. Therefore, E is either a four- or five-term chain complex, and by fixing a three-term subcomplex, one can describe a code $\mathcal{E}$. The elements of unused vector spaces of E can either be ignored or associated with syndrome "meta-checks". The respective Tanner graph of the code $\mathcal{E}$ looks like a 2D square grid of modules with each module containing the code $\mathcal{C}$.

Let me produce an explicit example. Here, I consider intra-modular connectivity that corresponds to a pseudo-random classical LDPC code with a higher degree and less geometrically local qubit connectivity than the repetition code before. This code is obtained by generating a random sparse parity check matrix with dimensions 51×60 (see repository [146] for the full matrix). Through exhaustive search over all codewords, one can establish that the code has parameters $[60, 9, 20]$. Its maximum row or column weight is 8, but on average each check has a support of ≈ 5 bits. The respective chain complex C has homology dimension $\dim(H_1(C)) = 9$ since it encodes 9 bits and $H_0(C)$ is trivial as all parity checks are linearly independent. The homological properties of the chain complex D associated with the surface code describing the inter-modular layout are given in the previous example. Note that $H_2(D)$ is trivial, and for this example, I choose a surface code with length $L = 20$. Then, by identifying the elements of E_2 as data qubits, the resulting code $\mathcal{E}$ has $n = \dim E_2 = \dim(C_1 \otimes D_1 \oplus C_0 \otimes D_2) = 65040$ data qubits, where D_2 and C_0 are vector spaces associated with Z and X parity checks respectively. The number of encoded qubits is given by the dimension of the second homology $H_2(E)$, which one can compute using the Künneth formula (Eq. 5.15) as

$$k = \dim H_2(E) = \dim(H_1(C) \otimes H_1(D) \oplus H_0(C) \otimes H_2(D)) = 9. \tag{5.20}$$

Generally, finding the distance of a code is not trivial, and therefore, Monte-Carlo or similar approaches are usually employed. However, for a special case of a hypergraph product code where one of the seed codes is a classical code (two-term chain complex), Zeng and Pryadko [144] proposed a method to compute the distance exactly. Specifically, they proved the following theorem

Theorem 5.2. *(Reproduced from [144]) Let A be an m-term chain complex with distances d_i for $0 \leq i \leq m-1$ and let B be a two-term chain complex. Then,*

$$d_i(A \otimes B) = min(d_{i-1}(A)d_1(B), d_i(A)d_0(B)).$$

Here, the homological distance d_i is the minimum Hamming weight of a non-trivial representative in the ith homology group. That is, if one associates data qubits with the ith vector space in the chain complex, then the code has distance $d = \min(d_i, d^i)$, where d^i is the ith cohomological distance.

Applying Theorem 5.2 to the example above, the Z distance of the code is

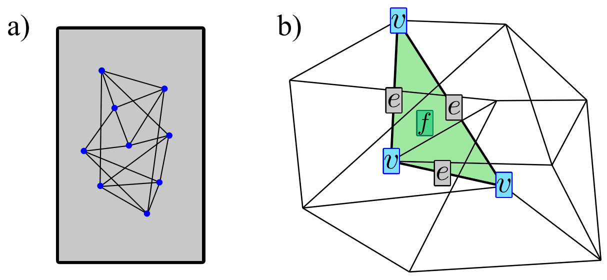
$$d_2(E) = \min(d_1(D)d_1(C), d_2(D)d_0(C)) = 400, \tag{5.21}$$

where by convention $d_i(A) = \infty$ if $H_i(A)$ is trivial. Similarly, the X distance of the code can be found using cohomology

$$d^2(E) = \min(d^1(D)d^1(C), d^2(D)d^0(C)) = 20. \tag{5.22}$$

Therefore, the code $E = C \otimes D$ has parameters $[\![65040, 9, 20]\!]$. In comparison to the 3D surface code (with full parameters given in Subsection 5.4.3) one can see that this example code encodes 9 times more qubits at the cost of increasing the overall number of physical qubits by a factor of about 3. One should expect such favourable trade-offs for architectures with less constrained qubit connectivity.

The same idea can be generalised to intra-modular connectivity that corresponds to a quantum code, i.e., a three-term chain complex. Note that this may imply a higher degree of connectivity and a higher number of qubits associated with each module. The latter implies that there are more connections between modules as per Def. 5.4. Due to the generality of the chain complex formalism, this construction is equivalent to the previous example, with the only difference being that the resulting product is a 5-term chain complex.



$$C = C_1 \xrightarrow{\partial_1} C_0 \qquad D = D_2 \xrightarrow{\partial_2} D_1 \xrightarrow{\partial_1} D_0$$

Figure 5.10: (a) The intra-modular qubit connectivity is represented with a chain complex C, where the physical qubits sit on all i-chains. (b) Similarly, the inter-modular connectivity is represented with a chain complex D, where modules are placed on every i-chain. Some elements of the chain complex are highlighted. The code $E = C \otimes D$ fully respects the connectivity constraints of the architecture (Theorem 5.1).

5.5.2 Beyond planar surface layouts

In a similar fashion to intra-modular connectivity, one can generalise the layout of the modules. While the 2D grid (corresponding to a surface code) has the advantage of planarity, it is known to limit the parameters of the derived hypergraph product codes. Additionally, some architectures, e.g. based on photonic links between modules, might be subject to other than nearest-neighbour connectivity constraints, and hence, have no benefit from a planar layout of modules. In such cases, the layout of the modules can be described with less geometrically local codes.

For example, a given inter-modular connectivity may correspond to a three-term chain complex

$$D = D_2 \xrightarrow{\partial_2} D_1 \xrightarrow{\partial_1} D_0.$$

That is, each basis i-chain is associated with a module such that the modules are connected according to the boundary maps of D. Visually, if one associates faces, edges and vertices to spaces D_2, D_1 and D_0 respectively, then the boundary maps ∂_2, ∂_1 describe the incidence matrices between faces and edges, and edges and vertices respectively. Moreover, if C denotes the chain complex describing the intra-modular connectivity, then by Theorem 5.1 the tensor product $C \otimes D$ represents a hypergraph product code that respects the connectivity of the overall architecture. This general idea is illustrated in Fig. 5.10.

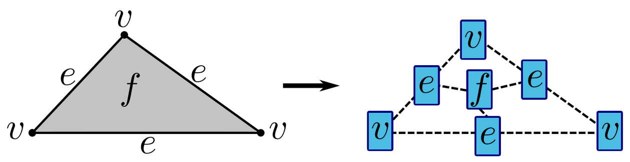


Figure 5.11: Any chain complex (left) can be used to describe the inter-modular connectivity by replacing every basis element of the chain complex with a module (right) and connecting them according to the boundary maps (dashed lines).

With this, I conclude the main idea behind constructing quantum LDPC codes for modular architectures. Let me briefly summarise it. The recipe is to represent intra- and inter-modular connectivity with two LDPC codes. Then, using the tensor product of the chain complexes describing these codes, one constructs a new hypergraph product code which is guaranteed to satisfy the architectural connectivity constraints. This already gives moderately good codes for a given architecture, however, in the next section I will describe an even stronger code construction by altering the definition of inter-modular connectivity. Before doing that, let me remark the following.

The construction presented so far can be viewed from the other direction. For instance, an architecture might pose no connectivity constraints between the modules, apart from demanding that each module is connected to at most a constant number of other modules. Then, a natural question to ask is how to connect them in order to implement a well-performing LDPC code. The perspective that I have introduced in this chapter allows one to partially answer this. Given N number of modules, one can find a good classical/quantum LDPC code with n bits and m checks such that $n + m = N$. Then, the modules are connected according to the chain complex representing this code. Fig. 5.11 depicts the idea. Of course, the overall code also depends on the intra-modular connectivity, but the same reasoning can be applied there.

5.6 Balanced product codes

In the previous section, I explored the idea of viewing modular architectures as hypergraph products of two codes. In order to further generalise this perspective and construct better-performing codes, I will consider modular architectures where the inter-modular connections are not constrained between the respective qubits only. Instead, these connections can be *twisted* in some way that is dictated by the chosen product construction. Hence, let me redefine inter-modular connectivity as follows

Definition 5.5. *A module M_k is connected to a module M_j if the architecture allows us to directly implement two-qubit entangling operations between a qubit $q_i \in M_k$ and its respective qubit $q_l \in M_j$ for all i, l.*

Note that some quantum computing platforms that consider linking modules together with photonic links or similar approaches, already allow this degree of freedom. Using this setting, one can consider more general products of codes than the standard hypergraph product. Specifically, I will show that the newly defined inter-modular connectivity allows one to construct codes that are called *balanced product codes* [124]. As before, these codes will fully respect the given architectural connectivity constraints. Before proceeding to the proof, let me shortly introduce the relevant background on balanced product codes.

5.6.1 Balanced product chain complexes

Balanced product quantum codes were first introduced by Nikolas Breuckmann and Jens Eberhardt in 2020 [124]. Similar to hypergraph product codes, these codes are constructed using balanced products of chain complexes that share ideas with balanced products of topological spaces. To define these products, it is easier to start by defining the balanced product of individual vector spaces.

Let V, W be vector spaces with a linear right and left action respectively of some finite group G. The balanced product is defined as the quotient

$$V \otimes_G W = V \otimes W / \langle vg \otimes w - v \otimes gw \rangle, \tag{5.23}$$

where $v \in V, w \in W$, and $g \in G$. For the rest of the section, I will consider cases where the vector spaces V and W are based and the action of G restricts to an action on these bases. Assuming that, the basis of $V \otimes_G W$ is given by $X \times_G Y =$

$X \times Y/\sim$, where the equivalence relation $\sim$ is defined as $(x, y) \sim (xg, g^{-1}y)$ for all $x \in X, y \in Y$, and $g \in G$.

The same product can be extended to chain complexes. Let C and D be chain complexes with a linear right and left action respectively of some finite group G. Then, the *balanced product double complex* $C \boxtimes_G D$ has vector spaces

$$(C \boxtimes_G D)_{p,q} = C_p \otimes_G D_q, \tag{5.24}$$

with horizontal and vertical boundary maps defined in the same way as in the tensor product double complex Eq. 5.10. Finally, the *balanced product complex* is the total complex

$$C \otimes_G D = \mathrm{Tot}(C \boxtimes_G D), \tag{5.25}$$

which again is defined analogously to the tensor product case. Moreover, if G is a finite group of odd order, the authors of [124] showed that a version of the Künneth formula can be applied to the balanced product as

$$H_n(C \otimes_G D) \cong \bigoplus_{p+q=n} H_p(C) \otimes_G H_q(D). \tag{5.26}$$

5.6.2 Architecture tailored codes from balanced products

To prove that the more general inter-modular connectivity including twists allows us to construct better quantum codes than those constructed as hypergraph products, I consider cases where one can cast the balanced product $C \otimes_G D$ as a fiber bundle complex $B \otimes_\varphi D$. In this chain complex product, B denotes the base, D denotes the fiber and φ the connection that describes the twists of the fiber along the base. Mathematically, the fiber bundle complex $B \otimes_\varphi D$ is constructed in an equivalent way to the tensor product complex (see Eq. 5.10) but with horizontal boundary maps ∂^h acting on spaces $B_1 \otimes D_i$ as $\partial^h = \partial_\varphi(b_1 \otimes d) = \sum_{b_0 \in \partial^B b_1} b_0 \otimes \varphi(b_1, b_0)(d)$, where $b_i \in B_i$ and d is any i-chain of D. Using the homological language, the connection φ represents an automorphism on the fiber D that alters the horizontal boundary maps of the double complex $B \boxtimes D$. Similarly to other products, the

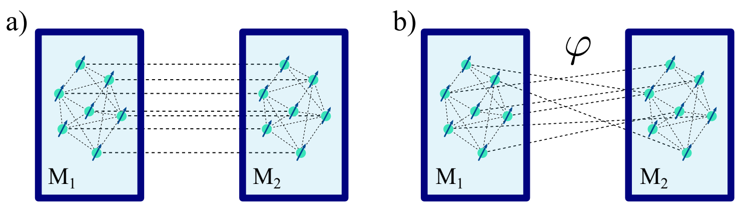


Figure 5.12: Twisted inter-modular connections (b) allow one to create better codes than the untwisted variant (a).

fiber bundle complex can be used to describe a quantum error correcting code once one identifies the data qubits and the parity checks correspondingly. Such codes are called *fiber bundle codes* [121]. In the modular architecture setting, it is crucial to correctly identify the base and fiber chain complex to ensure that connections are twisted between modules only. Fig. 5.12 shows cases of untwisted (a) and twisted (b) inter-modular connectivity. Note that, twisting connections in the "other direction" (i.e., the connections are twisted between layers of respective qubits across all modules) generally results in a low connection count in each module. For many quantum platforms, the intra-modular connections are considered to be faster to implement and less noisy, and therefore, preferential over the qubit connections between distinct modules [54, 55, 147].

Breuckmann and Eberhardt in [124] showed that when C is a two-term complex and H is abelian and acts freely on the bases of each C_i, then there exists a connection φ such that $C \otimes_G D = B \otimes_\varphi D$, where $B_i = C_i/\langle cg - c\rangle$. Therefore, a wide range of codes constructed from balanced products can be recast into the language of fiber bundle codes. For the purpose of code construction for architectures with twisted inter-modular connectivity, I will restrict myself to these cases. Therefore, I prove the following theorem in terms of fiber bundle codes.

Theorem 5.3. *Let D be a two-term and C a two- or three-term chain complex. Let their boundaries ∂^C and ∂^D define the intra- and inter-modular connectivity as given in Def. 5.3 and Def. 5.5 respectively. Then, a fiber bundle code $\mathcal{E}$ corresponding to the chain complex $E = D \otimes_\varphi C$ respects the connectivity constraints of the architecture.*

Proof. The resulting chain complex E (corresponding to $\mathcal{E}$) is at least a three-term chain complex given that both C and D are at least two-term chain complexes. Therefore, we can identify basis chains of E_i with data qubits, E_{i+1} with Z parity checks and E_{i-1} with X parity checks. The required qubit connectivity of $\mathcal{E}$ is defined by the parity check matrices H^E, which are given by the boundary maps

$$\partial^E = \partial^h + \partial^v = \partial_\varphi + \mathrm{id}^D \otimes \partial^C, \tag{5.27}$$

where

$$\partial_\varphi(d_1 \otimes c) = \sum_{d_0 \in \partial^D d_1} d_0 \otimes \varphi(d_1, d_0)(c)$$

in which $d_i \in D_i$ and c is any i-chain of C. Here, φ denotes the connection and hence $\varphi(d_1, d_0)$ describes a specific element of the automorphism group acting on the fiber C.

Note that basis i-chains of C label the qubit c within each module and that basis i-chains of D label each module d. We can ensure that $\mathcal{E}$ respects the connectivity constraints of the architecture if both terms of Eq. 5.27 match the given qubit connectivity. We do so, by looking at both of the terms separately. The first term in Eq. 5.27 describes that each qubit c in a module d_1 is connected to qubits labeled $\varphi(d_1, d_0)c = c'$ in adjacent modules $d_0 \in \partial^D(d_1)$ $\forall d$. This connectivity requirement is fully satisfied by the new definition of inter-modular connectivity Def. 5.5. For this exact reason, I consider the base of the fiber bundle to represent the code describing the inter-modular connectivity as I want to *twist* connections only between the modules. The proof that the second term of Eq. 5.27 describes the intra-modular connectivity is the same as in Theorem 5.1. Therefore, the required qubit connectivity of $\mathcal{E}$ is given by some additive combination of terms which define the intra- and inter-modular connectivity respectively.

□

The above theorem, therefore, establishes a correspondence between modular architectures and a specific class of balanced product quantum codes. While the proof considers codes that are equivalent to fiber bundle codes, one may expect that a wider range of product codes can be constructed that match the connectivity at hand.

For a simple example, let me construct a balanced product code from two classical codes represented by chain complexes C and D. The complex C describes the intra-modular connectivity and corresponds to a $d = 15$ cyclic repetition code as presented in Section 5.4. The code D describes the inter-modular connectivity and was obtained by generating a random sparse parity-check matrix with dimensions 255×450 and a cyclic symmetry of order 15. It encodes $\dim(H_1(D)) = 195$ bits. Since both codes share the cyclic symmetry I take the balanced product over the group $\mathbb{Z}_{15}$. The product can be recast as a fiber bundle code and therefore satisfies Theorem 5.2. The resulting code $D \otimes_{\mathbb{Z}_{15}} C$ has $n = 705$ qubits and encodes at least $k = \dim(H_1(D/\mathbb{Z}_{15})) = 195/15 = 13$ logical qubits, where the calculation follows from the Künneth formula for fiber bundle codes [121]. The X and Z parity checks have approximate average weights of 10 and 6 respectively. An extensive probabilistic distance search with *QDistRnd* [148] showed that the code distance is at most 15, I believe this bound is saturated. Hence, the balanced product code has expected parameters $[\![705, 13, 15]\!]$ and all qubits (including check qubits) can be partitioned into 47 modules with 30 qubits each. In comparison, encoding 13 qubits into rotated surface codes with the same distance require 2925 data qubits. The parity check matrix used for the code D and parity check matrices H_X and H_Z for the balanced product code can be found at a repository [146].

5.7 Discussion

In this chapter, I have proposed a novel correspondence between concepts from two rapidly evolving domains - quantum LDPC code constructions and modular quantum computing architectures. Using tools from homological algebra that have become essential in code design, I provide a way to view and construct quantum codes that respect the given connectivity constraints of a modular architecture. This work constitutes a step towards closing the gap between recent results of asymptotically good families of LDPC codes and practical quantum error correction.

There are several straightforward paths to extend this work. First, one could generalise the definitions of intra- and inter-modular connectivity that I have chosen.

For example, for some codes, it may not be necessary to have the same inter-modular connectivity in each module. Depending on which vector space the module is associated with, some qubits may not have an assignment. This was the case observed in the first example of a 3D surface code, see Table 5.1. Similarly, some connections may not be required between specific modules.

My co-author Lucas Berent proposed an idea to further generalise the architectural structure. For example, one could foresee an architecture with layouts of layouts of modules. In this way, a hierarchical structure is created, and it may be natural to construct quantum LDPC codes that are products of more than two codes. I leave this formalism open for future work.

On the practical side, it would be very valuable to find more and better instances of LDPC codes than the ones I have presented in this chapter. Finding well-performing finite-size quantum codes may greatly influence the design of architectures, especially at such an early stage. In general, many open questions about the practical aspects of quantum LDPC codes remain. For example, it is not yet clear what is the best strategy for implementing fault-tolerant logical gates. Answering this and other practically relevant questions may take us closer to reducing the overheads necessary for fault-tolerant quantum computing.

Chapter 6

Conclusion

In this thesis, I have presented several novel advances in quantum error mitigation and error correction for practical quantum computation. In Chapter 3, I introduced an intuitive way to mitigate errors using the probabilistic error cancellation strategy. The proposed protocol was shown to be able to deal with sophisticated noise profiles such as spatially or temporally correlated noise while assuming no or very minimal knowledge of the error model. In the following chapters, I discussed aspects of quantum error correction. I started by showing that a scalable fault-tolerant quantum computation is possible on a planar qubit array with a finite rate of fabrication defects. Chapter 4 includes both rigorous proof of this statement and extensive numerical results supporting it. Finally, in Chapter 5, I proposed a new perspective on how to view and design quantum error-correcting codes that are tailored for modular architectures. These constructions exploit the similarity between the tensor product code constructions and the tensor-like layout of qubits in modular architectures. These and other practical advancements in QEM and QEC will be crucial in guiding the design of emerging quantum computers.

Let me further comment on some of the overarching ideas presented in this thesis. It is clear that for large-scale quantum computation, QEC is necessary. However, some recent results indicate that QEM may also play an important role in these systems [49]. Many routines that are part of a larger fault-tolerant scheme could benefit from reduced error rates due to some cleverly designed error mitigation strategies. I believe this will be an interesting further direction to research.

It will always be important to devise better and more practical error-correcting codes. Emphasis on practical here, since we already know that there exist LDPC code families that are asymptotically good. However, such asymptotic scaling does not necessarily mean that the code performs well in practice, even when ignoring the qubit connectivity constraints. Therefore, comparing these and other codes using both classical simulations and quantum experiments (to whatever scale one can perform them) will be critical in deciding which codes are practically relevant. In addition, these simulations and experiments should include all elements of fault-tolerant quantum computation. From reliable storing of encoded information to performing fault-tolerant gates and decoding the error syndrome. Only then one can make an informed guess on which QEC protocol to choose for scalable quantum computation.

On top of that, it is important to investigate code performance with more realistic noise profiles that are directly supplied by the experimentalists. In turn, one could construct error-correcting codes that are especially robust against specific types of noise. Some of such codes are already known, for example, the XZZX surface code is well suited against biased noise [95]. Overall, designing a code for a specific platform could strongly guide the experimental teams to design architectural layouts that fit that code. This kind of back-and-forth between theorists and experimentalists will steer us onto a quick path towards scalable and fault-tolerant quantum computation.

On the question of when we will see a large-scale fault-tolerant quantum computer and what it will look like, it is hard to make any predictions. However, I truly believe that it will not take hundreds of years, and I hope to see this elusive machine operating with my own eyes - especially after dedicating years of my past and future life towards making one work. The way I see it, among many other things, this development will be greatly accelerated by finding novel high-performing codes tailored to quantum platforms. It seems unlikely that the same old surface code stands tall after years of new QEC research. Therefore, I find the topic of designing new QEC codes to become more and more exciting, and I invite anyone reading this thesis to try their hand too.

Appendices

Appendix A

The significant error approach

A.1 Clifford circuit overhead

In this passage, I present a numerical study showing the effect of different Clifford overhead constants c for circuits up to 8 qubits using the significant-error approach. See Fig. A.1. While no asymptotic scaling for c can be undeniably determined with respect to the circuit size, the data suggest that for small systems a Clifford overhead constant $c = 7$ is sufficient to introduce only negligible errors due to the truncation of the training set $\mathbb{T}$. For the simulations given in the main text, I chose $c = 3$, since the error due to a finite sampling (shot noise) dominates the truncation error.

A.2 Filtering of Clifford circuits

Consider a Clifford quantum circuit measuring the error-free mean value E^{ef} of some Pauli operator P. One can write is as $E^{\text{ef}} = \mathrm{Tr}(P\left|\psi\right\rangle\left\langle\psi\right|)$, where $\left|\psi\right\rangle$ is the output state. Because the circuit is Clifford, $\left|\psi\right\rangle$ is an eigenstate of some set of Pauli operators S. If $P \in S$ then $E^{\text{ef}} = \pm 1$, otherwise $E^{\text{ef}} = 0$.

Now consider a Pauli error Q happening somewhere in the circuit. This error can be commuted to the end of the Clifford circuit, during which it is changed to a different Pauli error Q'. In this case, the mean value of P is $E = \mathrm{Tr}(PQ'\left|\psi\right\rangle\left\langle\psi\right|Q')$, and depending on whether P commutes or anti-commutes with Q', one obtains $E = \mathrm{Tr}(P\left|\psi\right\rangle\left\langle\psi\right|) = E^{\text{ef}}$ or $E = -\mathrm{Tr}(P\left|\psi\right\rangle\left\langle\psi\right|) = -E^{\text{ef}}$. This means that the effect of error Q is at most a flipped error-free mean value of some operator P.

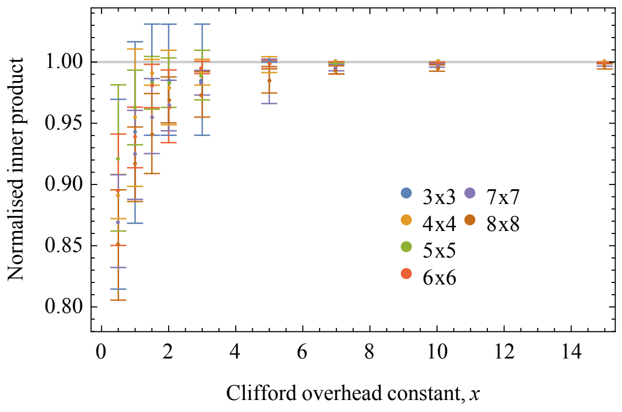


Figure A.1: Normalised inner product $\eta_x.\eta_{20}/(|\eta_x||\eta_{20}|)$ between quasi-probability distributions that are obtained in the learning part of the protocol. The subscripts denote the Clifford overhead constant that was used to learn the distribution. For example, $c = 20$ corresponds to η_{20}. I plot the inner product for different circuit sizes $n \times n$ where n is both the number of qubits and the number of layers of two-qubit gates. Here, I have chosen q_{20} as the reference for other distributions assuming that η_{20} deviates only marginally from η'_{opt} which is obtained from the full training set $\mathbb{T}$.

If $E^{\text{ef}} = 0$, then no Pauli error can change it! Therefore, a circuit that produces $E^{\text{ef}} = 0$ does not contribute to the loss function, and I filter it out from the set of randomly generated circuits.

References

[1] C. Gidney and M. Ekerå, “How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits”, Quantum **5**, 433 (2021).

[2] Y. Kim et al., “Evidence for the utility of quantum computing before fault tolerance”, Nature **618**, 500–505 (2023).

[3] R. Acharya et al., “Suppressing quantum errors by scaling a surface code logical qubit”, Nature **614**, 676–681 (2023).

[4] A. Montanaro, “Quantum algorithms: an overview”, npj Quantum Information **2**, 15023 (2016).

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2010).

[6] C. J. Wood, “Non-completely positive maps: properties and applications”, (2009), arXiv:`0911.3199`.

[7] W. H. Zurek, “Decoherence and the Transition from Quantum to Classical”, Phys. Today **44**, 36–44 (1991).

[8] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, “Quantum computational chemistry”, Rev. Mod. Phys. **92**, 015003 (2020).

[9] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”, SIAM Journal on Computing **26**, 1484–1509 (1997).

[10] F. Arute et al., “Quantum supremacy using a programmable superconducting processor”, Nature **574**, 505–510 (2019).

[11] D. Gottesman, “The heisenberg representation of quantum computers”, (1998), arXiv:`quant-ph/9807006 [quant-ph]`.

[12] S. Aaronson and D. Gottesman, “Improved simulation of stabilizer circuits”, Phys. Rev. A **70**, 052328 (2004).

[13] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, “Trapped-ion quantum computing: Progress and challenges”, Applied Physics Reviews **6**, 021314 (2019).

[14] D. Press, K. De Greve, P. L. McMahon, T. D. Ladd, B. Friess, C. Schneider, M. Kamp, S. Höfling, A. Forchel, and Y. Yamamoto, “Ultrafast optical spin echo in a single quantum dot”, Nat. Photonics **4**, 367–370 (2010).

[15] K. Khodjasteh and D. A. Lidar, “Fault-tolerant quantum dynamical decoupling”, Phys. Rev. Lett. **95**, 180501 (2005).

[16] R. W. Hamming, “Error Detecting and Error Correcting Codes”, Bell Syst. Tech. J. **29**, 147–160 (1950).

[17] P. W. Shor, “Scheme for reducing decoherence in quantum computer memory”, Phys. Rev. A **52**, R2493 (1995).

[18] D. Gottesman, “Stabilizer codes and quantum error correction”, (1997), arXiv:`quant-ph/9705052 [quant-ph]`.

[19] O. Higgott, “Pymatching: a python package for decoding quantum codes with minimum-weight perfect matching”, ACM Transactions on Quantum Computing **3**, `10.1145/3505637` (2022).

[20] N. Delfosse and N. H. Nickerson, “Almost-linear time decoding algorithm for topological codes”, Quantum **5**, 595 (2021).

[21] J. Roffe, D. R. White, S. Burton, and E. Campbell, “Decoding across the quantum low-density parity-check code landscape”, Phys. Rev. Res. **2**, 043423 (2020).

[22] S. Krinner et al., “Realizing repeated quantum error correction in a distance-three surface code”, Nature **605**, 669–674 (2022).

[23] L. Egan et al., “Fault-tolerant control of an error-corrected qubit”, Nature **598**, 281–286 (2021).

[24] B. Eastin and E. Knill, “Restrictions on transversal encoded quantum gate sets”, Phys. Rev. Lett. **102**, 110502 (2009).

[25] S. Bravyi and A. Kitaev, “Universal quantum computation with ideal clifford gates and noisy ancillas”, Phys. Rev. A **71**, 022316 (2005).

[26] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, “Surface codes: towards practical large-scale quantum computation”, Phys. Rev. A **86**, 032324 (2012).

[27] A. Strikis, D. Qin, Y. Chen, S. C. Benjamin, and Y. Li, “Learning-based quantum error mitigation”, PRX Quantum **2**, 040330 (2021).

[28] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O’Brien, “A variational eigenvalue solver on a photonic quantum processor”, Nature Communications **5**, 4213 (2014).

[29] Y. Li and S. C. Benjamin, “Efficient Variational Quantum Simulator Incorporating Active Error Minimization”, Phys. Rev. X **7**, 021050 (2017).

[30] P. J. J. O’Malley et al., “Scalable quantum simulation of molecular energies”, Phys. Rev. X **6**, 031007 (2016).

[31] K. Temme, S. Bravyi, and J. M. Gambetta, “Error Mitigation for Short-Depth Quantum Circuits”, Phys. Rev. Lett. **119**, 180509 (2017).

[32] S. Endo, S. C. Benjamin, and Y. Li, “Practical Quantum Error Mitigation for Near-Future Applications”, Phys. Rev. X **8**, 031027 (2018).

[33] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, “Error mitigation extends the computational reach of a noisy quantum processor”, Nature **567**, 491–495 (2019).

[34] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, “Digital zero noise extrapolation for quantum error mitigation”, in 2020 ieee international conference on quantum computing and engineering (qce) (2020), pp. 306–316.

[35] J. Sun, X. Yuan, T. Tsunoda, V. Vedral, S. C. Benjamin, and S. Endo, “Mitigating realistic noise in practical noisy intermediate-scale quantum devices”, Phys. Rev. Appl. **15**, 034026 (2021).

[36] S. Lloyd, “Universal quantum simulators”, Science **273**, 1073–1078 (1996).

[37] Z. Cai, “Multi-exponential error extrapolation and combining error mitigation techniques for NISQ applications”, npj Quantum Information **7**, 80 (2021).

[38] D. F. Wise, J. J. Morton, and S. Dhomkar, “Using deep learning to understand and mitigate the qubit noise environment”, PRX Quantum **2**, 010316 (2021).

[39] C. Song, J. Cui, H. Wang, J. Hao, H. Feng, and Y. Li, “Quantum computation with universal error mitigation on a superconducting quantum processor”, Science Advances **5**, eaaw5686 (2019).

[40] S. Zhang, Y. Lu, K. Zhang, W. Chen, Y. Li, J. N. Zhang, and K. Kim, “Error-mitigated quantum gates exceeding physical fidelities in a trapped-ion system”, Nat. Commun. **11**, 1–8 (2020).

[41] S. Anders and H. J. Briegel, “Fast simulation of stabilizer circuits using a graph-state representation”, Phys. Rev. A **73**, 022334 (2006).

[42] T. P. Harty, D. T. Allcock, C. J. Ballance, L. Guidoni, H. A. Janacek, N. M. Linke, D. N. Stacey, and D. M. Lucas, “High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit”, Phys. Rev. Lett. **113**, 220501 (2014).

[43] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas, “High-Fidelity Quantum Logic Gates Using Trapped-Ion Hyperfine Qubits”, Phys. Rev. Lett. **117**, 060504 (2016).

[44] J. P. Gaebler et al., “High-Fidelity Universal Gate Set for $^{9}\mathrm{Be}^{+}$ Ion Qubits”, Phys. Rev. Lett. **117**, 060505 (2016).

[45] Z. Cai and S. C. Benjamin, “Constructing Smaller Pauli Twirling Sets for Arbitrary Error Channels”, Scientific Reports **9**, 11281 (2019).

[46] T. Jones and S. Benjamin, “Questlink—mathematica embiggened by a hardware-optimised quantum emulator*”, Quantum Science and Technology **5**, 034012 (2020).

[47] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, “QuEST and High Performance Simulation of Quantum Computers”, Scientific Reports **9**, 10736 (2019).

[48] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O’Brien, “Quantum error mitigation”, (2022), arXiv:`2210.00921 [quant-ph]`.

[49] C. Piveteau, D. Sutter, S. Bravyi, J. M. Gambetta, and K. Temme, “Error mitigation for universal gates on encoded qubits”, Phys. Rev. Lett. **127**, 200505 (2021).

[50] A. Strikis, S. C. Benjamin, and B. J. Brown, “Quantum computing is scalable on a planar array of qubits with fabrication defects”, Phys. Rev. Appl. **19**, 064081 (2023).

[51] A. Siegel, A. Strikis, T. Flatters, and S. Benjamin, “Adaptive surface code for quantum error correction in the presence of temporary or permanent defects”, (2022), arXiv:2211.08468 [quant-ph].

[52] H.-L. Huang, D. Wu, D. Fan, and X. Zhu, “Superconducting quantum computing: a review”, Science China Information Sciences **63**, 180501 (2020).

[53] X. Zhang, H.-O. Li, G. Cao, M. Xiao, G.-C. Guo, and G.-P. Guo, “Semiconductor quantum computation”, National Science Review **6**, 32–54 (2018).

[54] E. T. Campbell, “Distributed quantum-information processing with minimal local resources”, Phys. Rev. A **76**, 040302 (2007).

[55] L. J. Stephenson, D. P. Nadlinger, B. C. Nichol, S. An, P. Drmota, T. G. Ballance, K. Thirumalai, J. F. Goodwin, D. M. Lucas, and C. J. Ballance, “High-rate, high-fidelity entanglement of qubits across an elementary quantum network”, Phys. Rev. Lett. **124**, 110501 (2020).

[56] A. Y. Kitaev, “Fault-tolerant quantum computation by anyons”, Ann. Phys. (N. Y). **303**, 2–30 (2003).

[57] M. Takita, A. W. Cross, A. D. Córcoles, J. M. Chow, and J. M. Gambetta, “Experimental demonstration of fault-tolerant state preparation with superconducting qubits”, Phys. Rev. Lett. **119**, 180501 (2017).

[58] Z. Chen et al., “Exponential suppression of bit or phase errors with cyclic error correction”, Nature 2021 595:7867 **595**, 383–387 (2021).

[59] C. K. Andersen, A. Remm, S. Lazar, S. Krinner, N. Lacroix, G. J. Norris, M. Gabureac, C. Eichler, and A. Wallraff, “Repeated quantum error detection in a surface code”, Nature Physics 2020 16:8 **16**, 875–880 (2020).

[60] D. Bacon, “Operator quantum error-correcting subsystems for self-correcting quantum memories”, Phys. Rev. A **73**, 012340 (2006).

[61] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, “Topological quantum memory”, Journal of Mathematical Physics **43**, 4452 (2002).

[62] B. M. Terhal, “Quantum error correction for quantum memories”, Rev. Mod. Phys. **87**, 307–346 (2015).

[63] T. M. Stace, S. D. Barrett, and A. C. Doherty, “Thresholds for topological codes in the presence of loss”, Physical Review Letters **102**, 200501 (2009).

[64] J. M. Auger, H. Anwar, M. Gimeno-Segovia, T. M. Stace, and D. E. Browne, “Fault-tolerance thresholds for the surface code with fabrication errors”, Physical Review A **96**, 042316 (2017).

[65] S. Nagayama, A. G. Fowler, D. Horsman, S. J. Devitt, and R. V. Meter, “Surface code error correction on a defective lattice”, New Journal of Physics **19**, 023050 (2017).

[66] G. Zhu, T. Jochym-O’Connor, and A. Dua, “Topological order, quantum codes, and quantum computation on fractal geometries”, PRX Quantum **3**, 030338 (2022).

[67] S. Bravyi and J. Haah, “Analytic and numerical demonstration of quantum self-correction in the 3d cubic code”, (2011), arXiv:1112.3252.

[68] Y. Tomita and K. M. Svore, “Low-distance surface codes under realistic quantum noise”, Phys. Rev. A **90**, 062320 (2014).

[69] E. W. Dijkstra, “A note on two problems in connexion with graphs”, Numerische Mathematik **1**, 269–271 (1959).

[70] R. Raussendorf, J. Harrington, and K. Goyal, “A fault-tolerant one-way quantum computer”, Annals of Physics **321**, 2242–2270 (2006).

[71] H. Bombin and M. A. Martin-Delgado, “Quantum measurements and gates by code deformation”, Journal of Physics A: Mathematical and Theoretical **42**, 095302 (2009).

[72] C. Horsman, A. G. Fowler, S. Devitt, and R. V. Meter, “Surface code quantum computing by lattice surgery”, New Journal of Physics **14**, 123011 (2012).

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

[74] D. Litinski, “Magic state distillation: not as costly as you think”, Quantum **3**, 205 (2019).

[75] C. Gidney and A. Fowler, “A slightly smaller surface code S gate”, (2017), arXiv:`1708.00054 [quant-ph]`.

[76] Y. Li, “A magic state’s fidelity can be superior to the operations that created it”, New Journal of Physics **17**, 023037 (2015).

[77] J. Łodyga, P. Mazurek, A. Grudka, and M. Horodecki, “Simple scheme for encoding and decoding a qubit in unknown state for various topological codes”, Scientific Reports 2015 5:1 **5**, 1–10 (2015).

[78] M. McEwen et al., “Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits”, Nature Physics **18**, 107–111 (2022).

[79] J. M. Martinis, “Saving superconducting quantum processors from decay and correlated errors generated by gamma and cosmic rays”, npj Quantum Information **7**, 90 (2021).

[80] M.-X. Huo and Y. Li, “Learning time-dependent noise to reduce logical errors: real time error rate estimation in quantum error correction”, New Journal of Physics **19**, 123032 (2017).

[81] S. T. Spitz, B. Tarasinski, C. W. J. Beenakker, and T. E. O’Brien, “Adaptive weight estimator for quantum error correction in a time-dependent environment”, Advanced Quantum Technologies **1**, 1800012 (2018).

[82] N. H. Nickerson and B. J. Brown, “Analysing correlated noise on the surface code using adaptive decoding algorithms”, Quantum **3**, 131 (2019).

[83] J. Kelly et al., “Scalable in situ qubit calibration during repetitive error detection”, Phys. Rev. A **94**, 032321 (2016).

[84] H. Bombín, “Resilience to time-correlated noise in quantum computation”, Physical Review X **6**, 041034 (2016).

[85] X. Waintal, “What determines the ultimate precision of a quantum computer”, Phys. Rev. A **99**, 042318 (2019).

[86] H. Bombín, “Single-shot fault-tolerant quantum error correction”, Phys. Rev. X **5**, 031043 (2015).

[87] A. Kubica and M. Vasmer, “Single-shot quantum error correction with the three-dimensional subsystem toric code”, Nature Communications **13**, 6272 (2022).

[88] S. F. Lin, J. Viszlai, K. N. Smith, G. S. Ravi, C. Yuan, F. T. Chong, and B. J. Brown, “Empirical overhead of the adapted surface code on defective qubit arrays”, (2023), arXiv:2305.00138 [quant-ph].

[89] S. T. Flammia and J. J. Wallman, “Efficient estimation of pauli channels”, ACM Transactions on Quantum Computing **1**, 10.1145/3408039 (2020).

[90] R. Harper, S. T. Flammia, and J. J. Wallman, “Efficient learning of quantum noise”, Nature Physics **16**, 1184–1188 (2020).

[91] T. Wagner, H. Kampermann, D. Bruß, and M. Kliesch, “Pauli channels can be estimated from syndrome measurements in quantum error correction”, Quantum **6**, 809 (2022).

[92] C. Gidney and A. G. Fowler, “Efficient magic state factories with a catalyzed $|CCZ>$ to $2|T>$ transformation”, Quantum **3**, 135 (2019).

[93] S. Bravyi, G. Duclos-Cianci, D. Poulin, and M. Suchara, “Subsystem surface codes with three-qubit check operators”, Quantum Info. Comput. **13**, 963–985 (2013).

[94] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, “Topological and subsystem codes on low-degree graphs with flag qubits”, Phys. Rev. X **10**, 011022 (2020).

[95] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, “The XZZX surface code”, Nature Communications **12**, 2172 (2021).

[96] O. Higgott and N. P. Breuckmann, “Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead”, Phys. Rev. X **11**, 031039 (2021).

[97] M. A. Levin and X.-G. Wen, “String-net condensation: a physical mechanism for topological phases”, Phys. Rev. B **71**, 045110 (2005).

[98] H. Bombin and M. A. Martin-Delgado, “Topological quantum distillation”, Phys. Rev. Lett. **97**, 180501 (2006).

[99] M. B. Hastings and J. Haah, “Dynamically Generated Logical Qubits”, Quantum **5**, 564 (2021).

[100] S. Beigi, P. W. Shor, and D. Whalen, “The quantum double model with boundary: condensations and symmetries”, Communications in Mathematical Physics **306**, 663–694 (2011).

[101] A. Kitaev and L. Kong, “Models for gapped boundaries and domain walls”, Communications in Mathematical Physics **313**, 351–373 (2012).

[102] M. Levin, “Protected edge modes without symmetry”, Phys. Rev. X **3**, 021009 (2013).

[103] M. Barkeshli, C.-M. Jian, and X.-L. Qi, “Theory of defects in abelian topological states”, Phys. Rev. B **88**, 235103 (2013).

[104] M. S. Kesselring, F. Pastawski, J. Eisert, and B. J. Brown, “The boundaries and twist defects of the color code and their applications to topological quantum computation”, Quantum **2**, 101 (2018).

[105] C. Vuillot, “Planar floquet codes”, (2021), arXiv:2110.05348 [quant-ph].

[106] A. Strikis and L. Berent, “Quantum low-density parity-check codes for modular architectures”, PRX Quantum **4**, 020321 (2023).

[107] C. Monroe, R. Raussendorf, A. Ruthven, K. R. Brown, P. Maunz, L.-M. Duan, and J. Kim, “Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects”, Phys. Rev. A **89**, 022317 (2014).

[108] S. Bartolucci et al., “Fusion-based quantum computation”, Nature Communications **14**, 912 (2023).

[109] A. Gold, J. Paquette, A. Stockklauser, M. J. Reagor, M. S. Alam, A. Bestwick, N. Didier, A. Nersisyan, F. Oruc, A. Razavi, et al., “Entanglement across separate silicon dies in a modular superconducting qubit device”, npj Quantum Information **7**, 1–10 (2021).

[110] D. Hucul, I. V. Inlek, G. Vittorini, C. Crocker, S. Debnath, S. M. Clark, and C. Monroe, “Modular entanglement of atomic qubits using photons and phonons”, Nature Physics **11**, 37–42 (2015).

[111] B. Buonacorsi, Z. Cai, E. B. Ramirez, K. S. Willick, S. M. Walker, J. Li, B. D. Shaw, X. Xu, S. C. Benjamin, and J. Baugh, “Network architecture for a topological quantum computer in silicon”, Quantum Science and Technology **4**, 025003 (2019).

[112] R. Gallager, “Low-density parity-check codes”, IRE Transactions on Information Theory **8**, 21–28 (1962).

[113] C. E. Shannon, “A mathematical theory of communication”, Bell System Technical Journal **27**, 379–423 (1948).

[114] D. Gottesman, “Fault-tolerant quantum computation with constant overhead”, (2014), arXiv:1310.2984 [quant-ph].

[115] N. P. Breuckmann and J. N. Eberhardt, “Quantum low-density parity-check codes”, PRX Quantum **2**, 040101 (2021).

[116] N. P. Breuckmann, C. Vuillot, E. Campbell, A. Krishna, and B. M. Terhal, “Hyperbolic and semi-hyperbolic surface codes for quantum storage”, Quantum Science and Technology **2**, 035007 (2017).

[117] D. A. Lidar and T. A. Brun, *Quantum error correction* (Cambridge university press, 2013).

[118] N. Baspin and A. Krishna, “Connectivity constrains quantum codes”, Quantum **6**, 711 (2022).

[119] N. Baspin and A. Krishna, “Quantifying nonlocality: how outperforming local quantum codes is expensive”, Phys. Rev. Lett. **129**, 050505 (2022).

[120] J.-P. Tillich and G. Zémor, “Quantum LDPC Codes With Positive Rate and Minimum Distance Proportional to the Square Root of the Blocklength”, IEEE Trans. Inf. Theory **60**, 1193–1202 (2014).

[121] M. B. Hastings, J. Haah, and R. O’Donnell, “Fiber bundle codes: breaking the n 1/2 polylog (n) barrier for quantum ldpc codes”, in Proc. 53rd Annual ACM SIGACT Symposium Theory Computing (2021), pp. 1276–1288.

[122] P. Panteleev and G. Kalachev, "Degenerate quantum ldpc codes with good finite length performance", Quantum **5**, 585 (2021).

[123] P. Panteleev and G. Kalachev, "Asymptotically good quantum and locally testable classical LDPC codes", in Proc. 54th Annual ACM SIGACT Symposium Theory Computing (2022), pp. 375–388.

[124] N. P. Breuckmann and J. N. Eberhardt, "Balanced product quantum codes", IEEE Trans. Inf. Theory **67**, 6653–6674 (2021).

[125] A. Leverrier and G. Zemor, "Quantum tanner codes", in 2022 ieee 63rd annual symposium on foundations of computer science (focs) (Nov. 2022), pp. 872–883.

[126] M. A. Tremblay, N. Delfosse, and M. E. Beverland, "Constant-overhead quantum error correction with thin planar connectivity", Physical Review Letters **129**, 050504 (2022).

[127] R. Buyya, *High performance cluster computing: programming and applications, volume 2* (Prentice Hall, 1999).

[128] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics", Journal of big data **2**, 1–20 (2015).

[129] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario, "The future of quantum computing with superconducting qubits", Journal of Applied Physics **132**, 160902 (2022).

[130] P. Murali, D. M. Debroy, K. R. Brown, and M. Martonosi, "Architecting noisy intermediate-scale trapped ion quantum computers", in Proceedings of the acm/ieee 47th annual international symposium on computer architecture, ISCA '20 (2020), pp. 529–542.

[131] H. Bombin, I. H. Kim, D. Litinski, N. Nickerson, M. Pant, F. Pastawski, S. Roberts, and T. Rudolph, "Interleaving: modular architectures for fault-tolerant photonic quantum computing", arXiv preprint arXiv:2103.08612 (2021).

[132] J. Ramette, J. Sinclair, Z. Vendeiro, A. Rudelis, M. Cetina, and V. Vuletić, "Any-to-any connected cavity-mediated architecture for quantum computing with trapped ions or rydberg arrays", PRX Quantum **3**, 010344 (2022).

[133] J. Niu et al., "Low-loss interconnects for modular superconducting quantum processors", Nature Electronics **6**, 235–241 (2023).

[134] F. M. Gambetta, C. Zhang, M. Hennrich, I. Lesanovsky, and W. Li, "Long-range multibody interactions and three-body antiblockade in a trapped rydberg ion chain", Phys. Rev. Lett. **125**, 133602 (2020).

[135] K. A. Landsman, Y. Wu, P. H. Leung, D. Zhu, N. M. Linke, K. R. Brown, L. Duan, and C. Monroe, "Two-qubit entangling gates within arbitrarily long chains of trapped ions", Physical Review A **100**, 022332 (2019).

[136] H. Häffner, C. F. Roos, and R. Blatt, "Quantum computing with trapped ions", Physics reports **469**, 155–203 (2008).

[137] S. H. Choe and R. Koenig, "Long-range data transmission in a fault-tolerant quantum bus architecture", arXiv preprint arXiv:2209.09774 (2022).

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

[139] N. H. Nickerson, Y. Li, and S. C. Benjamin, "Topological quantum computing with a very noisy network and local error rates approaching one percent", Nature communications **4**, 1–5 (2013).

[140] Z. Cai, A. Siegel, and S. Benjamin, "Looped pipelines enabling effective 3d qubit lattices in a strictly 2d device", PRX Quantum **4**, 020345 (2023).

[141] M. Fogarty, personal communication, 2022-04-01.

[142] N. Delfosse, M. E. Beverland, and M. A. Tremblay, "Bounds on stabilizer measurement circuits and obstructions to local implementations of quantum ldpc codes", (2021), arXiv:`2109.14599 [quant-ph]`.

[143] C. A. Pattison, A. Krishna, and J. Preskill, "Hierarchical memories: simulating quantum ldpc codes with local gates", (2023), arXiv:`2303.04798 [quant-ph]`.

[144] W. Zeng and L. P. Pryadko, "Higher-dimensional quantum hypergraph-product codes with finite rates", Phys. Rev. Lett. **122**, 230501 (2019).

[145] O. Higgott and N. P. Breuckmann, "Improved single-shot decoding of higher-dimensional hypergraph-product codes", PRX Quantum **4**, 020332 (2023).

[146] `https://github.com/PurePhys/QuantumPCMs`, 2022.

[147] J. Ramette, J. Sinclair, N. P. Breuckmann, and V. Vuletić, "Fault-tolerant connection of error-corrected qubits with noisy links", arXiv preprint arXiv:2302.01296 (2023).

[148] L. P. Pryadko, V. A. Shabashov, and V. K. Kozin, "Qdistrnd: a gap package for computing the distance of quantum error-correcting codes", Journal of Open Source Software **7**, 4120 (2022).