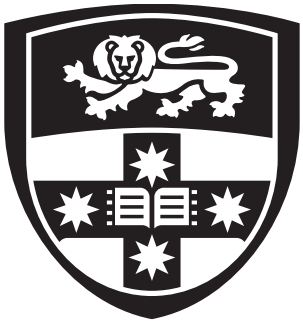
Decoders and circuits for practical quantum error correction



Samuel C. Smith

School of Physics
The University of Sydney

This thesis is submitted to the
Faculty of Science, School of Physics of The University of Sydney
in fulfillment of the requirements for the degree of
Doctor of Philosophy (Physics)

June 2024

Acknowledgements

First and foremost, I would like to thank my supervisors, Prof. Stephen Bartlett and Dr. Ben Brown.

Stephen for his guidance throughout the entirety of my higher education. Stephen was my first university lecturer and has played an instrumental role in my physics education from projectile motion onwards. His precise, clear and approachable manner when discussing physics kept me inspired about my project throughout the year, and his support always helped me to see the bigger picture.

Ben has supervised me throughout the entirety of my PhD, and introduced me to the world of research. He especially told me the importance of having a "compressible soul" in order to be able to bounce back from trial and error when things don't pan out. As well as that, he also showed me around the candlelit and smoky bodega scenes in Copenhagen, for which I am very grateful.

I would also like to thank the entire quantum information group, who I have loved getting to know over the last few years. Felix, Larry, Juliette, Edgar, Tom, Dom, Arkin, Tim, Nick, Evan, Mack, Angela, Sam, and others that I'm absolutely sure I've missed—you all make the workplace an exciting and warm place to be.

Thanks to all my close friends for being around me: Adam, Paul, Tommy, et al., I love seeing you all as much as possible. I thank all of my family: Mum and Dad, for all their love and organizing Sunday dinners that I look forward to every week. Dan and Jess (and of course Teddy and Billy, though Billy can't read this yet), Jeremy and Clare, I'm looking forward to a lot more netball in the future!

Carla, I of course cannot thank you enough for the tremendous amount of support you have showed for me, especially in the last few months (weeks? days? ... hours?) of frantic thesis writing. You have absolutely taken care of me, and also put up with me, and I am very, very grateful. (Also, tea was good—a nice drinking temperature.)

This research reported in this thesis was supported by the award of a Research Training Program scholarship.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged. This thesis contains fewer than 80,000 words, excluding the bibliography.

Samuel C. Smith
June 2024

Author Attribution Statement

This thesis consists of an introduction, three body chapters, and a conclusion. I wrote the introduction and the conclusion. The main body chapters are based on separate research projects, possibly with minor edits. Chapter 2 has been peer-reviewed and published, and Chapter 3 is currently under review. A statement of research contribution for each of these body chapters is listed below.

Chapter 2: Local predecoder to reduce the bandwidth and latency of quantum error correction

Published in: Physical Review Applied **19**, 034050 (2023)

Author list: S. C. Smith, B. J. Brown, and S. D. Bartlett.

Contributions: I am the lead author on this publication. The initial concept for this article was the result of ongoing discussions between all authors. I wrote the code for the numerical simulations, and collected the numerical results. I wrote the first draft of the article, and further editing and revision was done by all authors.

Chapter 3: Mitigating errors in logical qubits

Published in: arXiv preprint arXiv:2405.03766 (2024)

Author list: S. C. Smith, B. J. Brown, and S. D. Bartlett.

Contributions: I am the lead author on this project. The initial concept was the result of discussions between all authors. I developed and ran the numerical simulations, and was responsible for the "splitting individual logicals" method that enabled these simulations. I was also responsible for the analytic calculations. All authors contributed to discussions about use-cases for exclusive-decoding. I wrote the first draft of the article, except for the introduction. All authors contributed to further edits and revision.

Chapter 4: Designing time-optimal syndrome extraction circuits using Latin squares

Author list: S. C. Smith, B. J. Brown, and S. D. Bartlett.

Contributions: I am lead author on this project. All authors contributed to initial and ongoing discussions about optimizing color code circuits to correct against hook errors without additional fault-tolerant gadgetry. I was responsible for the Latin square method that was used for circuit design, and also developed the code used to perform the searches

presented. All authors contributed to discussions about our results, and I wrote the chapter, with feedback from other authors.

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Student signature: _____

Samuel C. Smith
June 2024

Supervisor signature: _____

Stephen D. Bartlett
June 2024

Abstract

In this thesis, we explore practical areas in the theory of quantum error correction that have become especially relevant in light of recent experimental advances. We address the question of whether classical decoding calculations, which are required in order to detect, track and correct errors, can be performed in real-time at a sufficient pace to keep up with the quantum clock speed. Our investigation includes consideration of practical details, such as the communication bandwidth that is required for the classical decoder to interact with the quantum control hardware. We consider the technique of post-selection, which is often used to boost logical performance in small-scale experimental demonstrations of quantum error correction. In particular, we address the common criticism that post-selection is not scalable, and investigate applications of post-selection for medium-scale state preparation circuits, which will continue to be relevant in large-scale quantum computers. Finally, we address the problem of circuit design, whereby theoretical constructs from quantum error correction are translated into a gate-level set of instructions that can be executed on hardware, and in particular we search for time-optimal circuits for detecting errors in topological codes.

Contents

1 Introduction

1.1 The quantum promise

The advent of scalable quantum computing promises to unlock new computational capabilities that will have a far-reaching impact across a wide range of industries and sciences. It has long been speculated that this extra computational power may allow more accurate simulations of nature [1]. The discovery of Shor's algorithm [2] in 1994 provides an early concrete example of a computational problem that cannot be solved using classical methods with any reasonable amount of resources, but that can be solved efficiently on a quantum computer, with real and significant consequences for cryptography. Some other known applications of quantum computing include simulating quantum chemistry for drug discovery, simulating condensed matter simulations for the design of solid state devices, and solving combinatorial optimization problems that are relevant for finance and logistics.

Despite quantum computing existing as a field for 30 years, the exploration of applications of quantum computing is still an active and evolving area of research, and new and more efficient algorithms are being discovered regularly [3]. Recently, the field is also seeing increasing amounts of interactions between industry and research as governments and businesses seek to explore and understand the potential use-cases of quantum computing. It is difficult to characterize in detail the exact impact that quantum computers will have, but what is known is

that the impact is very likely to be large in scale, wide-ranging in scope, and new applications and use-cases are likely to be discovered in a fast-moving and transformative fashion.

1.2 Development of quantum error correction

The primary challenge that must be overcome in order to build a quantum computer is that the individual components are imperfect. The state of a quantum computer is stored in physical systems called qubits. Qubits are prone to error when idling, and also when being manipulated by the gates that make up a quantum computation. This can destroy the accuracy of the computation, making it impossible to extract any useful output. It has always been recognised that the question of whether or not it is possible to build a scalable quantum computer hinges on whether we can find a way to correct errors [4–7].

A keystone result in quantum error correction (QEC) is the much-celebrated threshold theorem [8], which gives conditions under which a perfect quantum circuit can be simulated using a quantum circuit that is built out of noisy components. Importantly, the overhead required is not too large so as to destroy the advantage gained by using a quantum computer. Almost all research in the field of QEC, both leading up to [9–12] and following the threshold theorem, is concerned with the details of how to map a perfect quantum circuit to a noisy simulation of the circuit. Of particular interest is how this can be done as efficiently as possible, and with the most relaxed assumptions about hardware.

The possibility of correcting errors in a computation is crucial not only from a theoretical perspective, but also from a practical perspective. In order to perform a large quantum computation, the qubits and gates used must be of sufficiently high quality that an error is unlikely to occur at any point in the computation. Unfortunately, the physical qubits of today do not meet that standard, and the physical qubits of the foreseeable future are unlikely to meet that standard, barring major and unforeseen breakthroughs in the field of qubit engineering. As such, the most impactful potential use-cases of quantum computing remain out of reach until quantum error correction can be implemented at scale.

Despite the tantalizing and encouraging nature of the threshold theorem, in the past the gap between what is possible in theory and what can be achieved experimentally has been very large. As Steane initially observed in 1996 in regards to error correction, “the experimental production of such states . . . is a demanding task which remains to be addressed.” [10], and it has remained so until very recently. As such, experimental quantum computing in the so-called NISQ era (noisy intermediate-scale quantum era [13]) has instead been characterized by a focus on finding application-specific and scale-specific advantage, including by supremacy experiments [14, 15], by development of hybrid quantum-classical algorithms such as the VQE [16, 17] or QAOA [18, 19], and by using error mitigation techniques [20] in place of scalable quantum error correction.

1.3 Current state of play

It is now a very exciting time for experimental QEC, as advancements in hardware capabilities begin to allow early demonstrations of QEC primitives that will be used by a general fault-tolerant quantum computer. This includes, for instance, performing multiple rounds of error correction or demonstrating a fault-tolerant gate. Results of this kind have been shown across a wide range of hardware platforms, including qubits build from superconducting circuitry [21–28], trapped-ion qubits [29–33], spins in diamond nitrogen-vacancy centres [34], as well as neutral atoms [35]. The theory of QEC requires certain assumptions to be made about hardware capabilities, including for example that parallel gate operations are available, that noise is approximately local, and that qubits satisfy all of the DiVincenzo criteria [36]. These experimental results are exciting, since they indicate that many hardware platforms now have available these fundamental capabilities, and that they are of a sufficiently high quality that a meaningful error correction result can be extracted. In some cases, a *break-even* point has been reached [24–26, 30–33], whereby quantum error correction techniques are used to implement an encoded operation that is better than the equivalent physical operation that is available natively. These exciting results validate that quantum error correction is indeed possible across a broad range of hardware platforms, and can be used to increase tolerance to errors.

More recently, a particularly encouraging class of results are *scaling* results. These state-of-the-art demonstrations show quantum error correction being used to improve the performance of a logical primitive, as the number of qubits use in the primitive is increased [35, 37]. These results are impressive for two reasons. First, scaling results are cutting-edge in size, necessarily involving quantum error correction using a large number of physical qubits. It is very encouraging that hardware platforms are continuing to grow in number of qubits, whilst maintaining or improving the ability to perform parallel high-quality operations on those qubits. Second, scaling experiments allow an apples-to-apples comparison to be made between the encoded primitive built using X physical qubits, and the encoded primitive built using $Y > X$ physical qubits. This comparison validates a fundamental hypothesis of quantum error correction: that the additional error correction capability available in large systems can outweigh the additional sources of error due to having more qubits. The implication is that it will be possible to make the encoded operation arbitrarily accurate provided that hardware continues to improve and scale.

Nevertheless, it remains a challenge for the future to demonstrate a scalable quantum error correction capability that is also sufficiently flexible to be useful in a general quantum circuit. In a recent talk entitled “What is your logical qubit?” [38], Jeongwan Haah takes an operational interpretation of fault-tolerance, suggesting that “rather than saying by QEC we realized this many logical qubits” we should instead say “by QEC we can do this, or that, more and/or better”. Today’s experimental devices are still limited in size and error budget, which means

that when choosing a primitive for a scaling experiment, experimentalists do not have the luxury of considering how their chosen primitive will fit into a larger general purpose circuit. Instead, primitives are carefully chosen and optimized to fit within the constraints of today's devices. From a theory perspective, this makes it difficult to use recent scaling experiments to extrapolate overheads for general purpose error corrected quantum computation. Instead, these recent experiments are best characterized as heralding a transitional period, where we are beginning to see evidence that hardware is approaching the level required for general and scalable fault-tolerant quantum computation.

These pioneering achievements in experimental QEC are still of value for theorists since they provide key lessons that will continue to inform the ongoing development of the field, especially regarding how to optimize QEC. These experiments can validate or challenge assumptions, for instance they provide a high level of detail regarding error budgets for error correction protocols, as well as what hardware platforms have available in terms of connectivity and gates. Certain aspects of QEC come into sharp contrast in an experimental setting, such as the difficulty of performing decoding calculations in real-time, or the incredible usefulness of post-selection for minimizing logical failure probabilities. It should be recognized that the lessons learned from these early demonstrations are not limited in scope to attempting to just run this or that experiment on a particular device. It is still far from clear what the pathway to a large-scale error corrected quantum computer will ultimately look like, and it will certainly depend on current and future experimental capabilities. In terms of optimization, we remark that a quantum computer is always going to be finite size—the cost function associated with a fault-tolerant primitive in a large-scale quantum computer may not be the same as for a current-day device, but it will still be important to compactify and optimize QEC, in order to extract as much useful computation as possible. At this time, it is an opportunity for co-design between QEC theory and experiments, with fertile data and guidance from experiments, and the motivating prospect of seeing rapid and tangible reward for effort.

1.4 A framework for quantum error correction

In this section, we provide a brief overview of key concepts in quantum error correction. We start in Subsection 1.4.1 by introducing some preliminary notation, before moving on to a description of stabilizer codes, including their logical operators, errors and decoding in Subsection 1.4.2. In Subsection 1.4.3, we introduce the toric code, a prototypical example of a topological code, which is a focus of this thesis. Finally, in Subsection 1.4.4, we describe error-correction in the circuit model, which is a framework for specifying gate-level instructions that can be executed on hardware.

1.4.1 Preliminary notation

We assume a level of familiarity with quantum mechanics and the stabilizer formalism, reviewed in Ref. [39], and provide here only preliminary notation.

A *qubit* is a two-level system whose state space is the Hilbert space $\mathbb{C}^2$, spanned by a computational basis commonly denoted by $|0\rangle, |1\rangle$. A system of n qubits has a state space given by the tensor product of the state spaces of each of the n qubits. A 3-qubit basis state can be written for example as $|0\rangle \otimes |1\rangle \otimes |1\rangle$ or as $|011\rangle$.

We define the single-qubit *Pauli group* in terms of its generators $\mathcal{P}_1 = \langle iI, X, Z\rangle$, where

$$Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle, \quad X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle, \tag{1.1}$$

and I is the identity operator, and $i = \sqrt{-1}$. We also introduce notation for the Pauli operator $Y = iXZ$, as well as the X-type Pauli eigenstates $X|\pm\rangle = \pm|\pm\rangle$. Finally, the n-qubit Pauli group consists of all n-fold tensor products of single-qubit Pauli operators. A 3-qubit Pauli can be written for example as $Z \otimes I \otimes Y$, or as Z_1Y_3, although in the latter case care should be taken not to confuse this with matrix multiplication.

1.4.2 Logical qubits

The key insight of quantum error correction according to John Preskill is that we can "fight entanglement with entanglement" [40], meaning that if we encode quantum information into patterns of entanglement that extend over many qubits, then that information will be inaccessible by noise models that affect only a relatively few qubits. The stabilizer formalism provides a powerful and flexible tool to compactly describe highly entangled subspaces of many-body systems, making it extremely well suited to the task of quantum error correction.

In this subsection, we introduce stabilizer codes and their logical operators, which are a central construct in quantum error correction. Then we discuss what kinds of errors can occur on stabilizer codes, and how they can be detected and corrected.

Codespace A *stabilizer code* over n physical qubits is specified by writing down a set of $n-k$ independent and commuting Pauli operators with real prefactors, for any $k \geq 0$. These operators generate the stabilizer group, denoted $\mathcal{S}$. The *codespace* is the shared $(+1)$-eigenspace of all the stabilizers, which can be written

$$S|\psi\rangle = |\psi\rangle \tag{1.2}$$

for all stabiilizers S. The codespace is a 2^k-dimensional subspace of the full Hilbert space. We can visualise the stabilizers as holding fixed the state of $n-k$ virtual qubits, leaving k logical qubits free for us to encode information into. For $k=0$, the codespace is 1-dimensional, which defines a stabilizer state.

Logical operators The encoded quantum information can be manipulated by *logical operators*. Logical operators are usually denoted with a bar, as in $\bar{L}$, although the correspondence between physical and logical operators depends on a choice of basis for the codespace. A logical operator $\bar{L}$ must map the codespace into itself, and therefore satisfies

$$S\bar{L}\left|\psi\right\rangle = \bar{L}\left|\psi\right\rangle, \tag{1.3}$$

for all stabilizers S and code states $|\psi\rangle$. Equivalently, we can say that a Pauli logical operator $\bar{L}$ satisfies the condition that, for all stabilizers S, the operator $\bar{L}S\bar{L}$ is also a stabilizer. Inspection of Eq. (1.3) yields the interesting observation that logical operators that differ only by stabilizer multiplication are equivalent on the codespace. As a result, we sometimes use the term "logical operator" to refer to a whole equivalence class of logical operators. The exact meaning is usually clear from context, though we sometimes disambiguate by using "logical representative" to refer to a specific operator within a class. In particular, we define the *code distance*, denoted d, as the weight of the least-weight logical Pauli operator, minimized over all logical representatives from all non-trivial classes. The list $[\![n,k,d]\!]$ are collectively called the *code parameters*.

Errors Remarkably, all errors that can affect the physical qubits of a stabilizer code can be effectively modelled as *Pauli errors*, due to the dynamics induced by stabilizer measurement. Although the physical qubits may undergo more general interactions with the environment, the subsequent act of measuring the stabilizers projects the system back onto a mutual eigenspace of the stabilizer operators, and the net result is equivalent to acting on a code state by a Pauli error. After acting on a code state $|\psi\rangle$ with a Pauli error E, the system will be in an eigenspace of stabilizer S that is determined by the commutation relation between E and S. In particular, we have

$$SE\left|\psi\right\rangle = \pm E\left|\psi\right\rangle, \tag{1.4}$$

where the plus sign holds if E commutes with S, and the minus sign holds if E anti-commutes with S. We remark that, similarly to logical operators, errors that differ only by stabilizer multiplication are equivalent on the codespace, up to a physically meaningless global phase. In accordance with convention, a measurement outcome $m=0,1$ of a given stabilizer indicates a post-measurement state that is in the $(-1)^m$-eigenspace of the stabilizer. With this convention, if E commutes with S, then a measurement outcome $m=0$ will be observed, which we sometimes refer to as a *trivial* outcome. Otherwise, if E anti-commutes with S, then a measurement

outcome $m = 1$ will be observed, which we sometimes refer to as a *non-trivial* outcome, or *defect*. The set of all measurement outcomes is called the *error syndrome*.

Decoding The inference problem of determining an error that is likely to have caused a given syndrome is known as *decoding*. The aim of successful decoding is, given a code affected by a Pauli error E, to find a *Pauli correction* C that returns the system state back to its initial state. We can write this as

$$CE\,|\psi\rangle = |\psi\rangle\,, \tag{1.5}$$

or equivalently, we can say that CE should be a stabilizer. In practice, of course, we do not know the actual error E, we only know how it commutes with the stabilizers by the error syndrome. This means that we cannot rule out the possibility that a correction operator that was intended to satisfy Eq. (1.5), instead satisfies $CE\,|\psi\rangle = \bar{L}\,|\psi\rangle$, where $\bar{L}$ is a non-trivial logical operator that corrupts the encoded state. The best we can do is to use knowledge of an error model to find a correction that is likely to return the system to its initial state. If we first write down any dummy correction operator C' that is consistent with the syndrome, then the probability that applying C' will result in a logical error equivalent to $\bar{L}$ can be computed by explicitly summing over errors, as below:

$$P_L = \sum_{EC' \sim \bar{L}} \mathbb{P}(E). \tag{1.6}$$

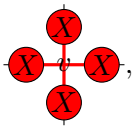
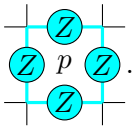
where $\mathbb{P}(E)$ denotes the probability of an error occurring according to an assumed noise model, and $\sim$ denotes equivalence up to multiplication by stabilizers. Then, the actual correction operator we apply is $C = C'\bar{L}$, where $\bar{L}$ is chosen to maximize Eq. (1.6). A decoder that finds an exact solution to Eq. (1.6) is called a *maximum likelihood decoder*. However, maximum likelihood decoding usually takes a prohibitively long time since there are exponentially many terms involved in the calculation, and research into decoding mainly involves finding efficient approximations to maximum likelihood decoding.

1.4.3 The toric code

A concrete and well-studied example of a stabilizer code is provided by the *toric code* (also called *surface code*), due to Kitaev [41]. Conventionally, the toric code is defined over a square lattice with periodic boundary conditions, and with qubits attached to every edge in the lattice. In the remainder of this discussion, we will refer to "edges" and "qubits" interchangeable.

The stabilizer group for the toric code is generated by two types of operators. For every vertex v in the lattice there exists a *star operator*, denoted A_v, which consists of the tensor product of Pauli X-type operators on every edge that is incident to v. For every plaquette p in

the lattice there exists a *plaquette operator*, denoted B_p, which consist of the tensor product of Pauli Z-type operators on every edge contained in the boundary of p. Pictorially, this is expressed as

$A_v =$  , $B_p =$  . (1.7)

The full stabilizer group is generated by the set of star operators and plaquette operators, and can be written

$$\mathcal{S}_{\mathrm{TC}} = \langle A_v, B_p \rangle, \tag{1.8}$$

where we understand v as running over all vertices, and p as running over all plaquettes. It can be checked that these stabilizers commute, since star operators and plaquette operators always overlap on an even number of qubits. The toric code belongs to a special class of stabilizer codes known as *Calderbank-Shor-Steane* (CSS) codes, since its stabilizer group can be generated using only pure X-type operators, and pure Z-type operators. We exclusively study CSS codes in this work.

On the toric code, errors and logical operators are geometrically *string-like* and *topological* in nature. The star operator at a vertex, for example, will detect any string of Z-type operators that terminates at that vertex. Thus, Z-type errors correspond to open strings on the lattice with defects located at their endpoints, and Z-type logicals (trivial or otherwise) correspond to closed loops. Importantly, two Z-type strings are equivalent by stabilizer multiplication if they (i) share the same defects, and (ii) can be smoothly deformed into one another. This means that it is only the topological properties of a string that carry physical meaning. A similar description holds for X-type operators, however in that case strings must be defined on the *dual* lattice, where the roles of plaquettes and vertices are interchanged. A Pauli basis of logical operators on the toric code can be expressed pictorially

$\bar{X}_1 =$  , $\bar{Z}_1 =$  , $\bar{X}_2 =$  , $\bar{Z}_2 =$  (1.9)

where the first diagram, for example, denotes an X-type string that wraps non-trivially around the torus in the horizontal direction. We remark that the distance is given by the linear size of the lattice. In particular, on a $L \times L$ lattice, the full set of code parameters is given $[\![2L^2, 2, L]\!]$.

An important property of the toric code is that it exhibits a *threshold* [42]. This means that, under reasonable locality assumptions about the noise model, the probability of making a mistake using Eq. (1.6) vanishes in the code distance, provided that the noise is sufficiently weak. In that case, we refer to the noise strength as being below threshold. Equally important from a practical perspective is the fact that the toric code admits an *efficient* decoder that can replace Eq. (1.6), whilst still exhibiting threshold behaviour, although the threshold value

will depend on the decoder being used. Broadly speaking, efficient decoders tend to exploit *materialised symmetries* of topological codes (see Ref. [43] for a recent review). For example, in relation to decoding Z-type errors (X-type errors can be decoded separately due to the CSS property of the code), the product of all star operators on the toric code is the identity, since each qubit is included in the support of two star operators. Thus we have

$$\left(\prod_v A_v\right)|\psi\rangle = |\psi\rangle\,, \tag{1.10}$$

for all states $|\psi\rangle$, which constraints the parity of X-type defects to be even. This means that decoding can be viewed as a *matching* problem, requiring us to pair up nearby defects, and this can be solved efficiently, for example by the *minimum-weight perfect matching* (MWPM) decoder [42], or by the *union-find* (UF) decoder [44]. In this work, wherever we use a specific decoding algorithim, we will provide a more detailed description of how it works.

1.4.4 The circuit model

Our treatment of quantum dynamics here is based around the *quantum circuit model* for computation. In this model, complex quantum dynamics is composed from three basic components:

1. Preparation of a qubit into a pre-defined state (commonly $|0\rangle$).
2. Evolution by a unitary gate from a pre-defined gate set
3. Measurement of a qubit in a pre-defined basis (commonly $|0\rangle\,/\,|1\rangle$)

Two gates that we use in this thesis are the Hadamard (H) gate, and the controlled-not (CNOT) gate. We define them in terms of their action under conjugation on Pauli operators. The Hadamard gate is defined:

$$Z \mapsto X, \quad X \mapsto Z, \tag{1.11}$$

and the CNOT gate is defined

$$Z \otimes I \mapsto Z \otimes I, \quad I \otimes Z \mapsto Z \otimes Z, \quad X \otimes I \mapsto X \otimes X, \quad I \otimes X \mapsto I \otimes X, \tag{1.12}$$

where the first qubits is called the control qubit and the second qubit is called the target qubit.

A quantum circuit can be represented by a circuit diagram, for example pictured in Fig. 1.1. Each horizontal line is a wire that corresponds to the state of the qubit over time, with time flowing from left to right. A basis for initialization and measurement can be provided by labelling qubit wires on the left and right, respectively, and each unitary operator is represented

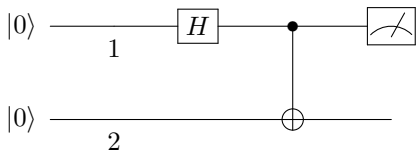


Fig. 1.1 A circuit with two qubits, labelled 0 and 1, both initialized into the $|0\rangle$ state. Qubit 1 undergoes a Hadamard, and then a CNOT gate is performed, controlled by qubit 1 and targeted on qubit 2. Finally, qubit 1 is measured in the computational basis, which will yield a random outcome.

by a block or other symbol in the bulk of the circuit. The action of a circuit can be understood by considering how a complete set of stabilizers flows through the circuit.

In terms of fault-tolerance, the circuit model provides a gate-level specification of instructions that can be implemented on hardware, and noise in each component gate can be individually characterized and modelled. This framework is both accurate and flexible enough to quantitatively model the results of small-scale demonstrations of quantum error correction. Then, a rough definition of a fault-tolerant circuit is one that can still operate as designed even when one or more individual circuit components experiences an error. In particular, we discuss circuits that extract syndrome information for topological codes in detail in Chapter 4.

We remark that the quantum circuit model provides a concrete setting to develop, study and describe quantum dynamics including quantum error correction, and also allows a range of different physical platforms for quantum computing to be compared on a common footing. However, by using the circuit model, we are implicitly centering our discussion around *active error correction* in *gate-based* quantum computers, although there do exist other modalities for quantum computation.

1.5 Avenues of research

This thesis explores strategies for quantum error correction, especially regarding how to ensure that QEC theory can be applied in practice. We focus on fundamental aspects of QEC that will be required by any memory experiment, such as decoding and syndrome extraction. A large part of this thesis can be described as compactifying QEC: reducing qubit and spacetime overheads but also relaxing its hardware demands such as communication bandwidth requirements or error budgets. This will be important for near-term demonstrations and also for quantum computing at scale. We provide here a brief summary of each chapter, aligned with this theme:

Chapter 2: We address the question of whether classical decoding calculations, which are required in order to detect, track and correct errors, can be performed in real-time at a sufficient

pace to keep up with the quantum clock speed. Our investigation includes consideration of practical details, such as the communication bandwidth that is required for the classical decoder to interact with the quantum control hardware.

Chapter 3: We consider the technique of post-selection, which is often used to boost logical performance in small-scale experimental demonstrations of quantum error correction. In particular, we address the common criticism that post-selection is not scalable, and investigate applications of post-selection for medium-scale state preparation circuits, which will continue to be relevant in large-scale quantum computers.

Chapter 4: We address the problem of circuit design, whereby theoretical constructs from quantum error correction are translated into a gate-level set of instructions that can be executed on hardware, and in particular we search for time-optimal circuits for detecting errors in topological codes.

2

Local predecoder to reduce the bandwidth and latency of quantum error correction

Abstract

A fault-tolerant quantum computer will be supported by a classical decoding system interfacing with quantum hardware to perform quantum error correction. It is important that the decoder can keep pace with the quantum clock speed, within the limitations on communication that are imposed by the physical architecture. To this end we propose a local 'predecoder', which makes greedy corrections to reduce the amount of syndrome data sent to a standard matching decoder. We study these classical overheads for the surface code under a phenomenological phase-flip noise model with imperfect measurements. We find substantial improvements in the runtime of the global decoder and the communication bandwidth by using the predecoder. For instance, to achieve a logical failure probability of $f = 10^{-15}$ using qubits with physical error rate $p = 10^{-3}$ and a distance $d = 22$ code, we find that the bandwidth cost is reduced by a factor of 1000, and the time taken by a matching decoder is sped up by a factor of 200. To achieve this target failure probability, the predecoding approach requires a 50% increase in the qubit count compared with a standard matching decoder.

2.1 Introduction

Performing complex calculations on a quantum computer will require maintaining quantum information coherently for long periods of time. To achieve this with noisy devices, quantum error correction (QEC) can be used, wherein measurements of code stabilizers throughout the computation give rise to a syndrome that can be interpreted by decoding software, allowing errors to be tracked and corrected [40, 45]. Current demonstrations of real-time error correction [22, 28, 30, 46, 47] are constrained by several pragmatic concerns. First, the classical decoder must keep pace with the high rate at which syndrome data is produced. In addition, the amount of syndrome measurement data transferred between the quantum layer and the decoding unit may be limited by some maximum bandwidth set by device constraints. These constraints are sufficiently stringent that it can be difficult to find a software-based solution to accelerate classical decoding such that it runs at the speed that is demanded by the quantum hardware.

To accelerate the decoding problem and reduce the communication costs, we develop a two-level decoding scheme for the surface code [41, 42] involving a local predecoder based on cellular automata (CA) [42, 48–56]. Our decoding scheme is designed to correct sparsely distributed errors by making greedy decoding decisions at the predecoder stage. The predecoder corrects these simple errors, however, its local nature means that locations where multiple errors have occurred may remain uncorrected. The predecoder sends the updated syndrome after it applies its correction to a minimum-weight perfect matching (MWPM) decoder to complete the decoding process.

As the CA can be implemented in parallel on dedicated hardware, without requiring long-range communication with a central processing unit, we find that our implementation has a significant impact on the latency and bandwidth requirements of the global decoding problem. Specifically, by reducing the number of errors, and in turn the number of syndrome defects, global decoding [42–44, 57–63, 63–73] is simplified and can therefore be completed at a higher speed. Moreover, by reducing the number of syndrome defects per unit volume, i.e., the defect density, we can reduce the size of the message that needs to be passed to the global decoder using a suitable strategy for data compression.

Previous and concurrent work has considered using the full hardware stack to distribute and pipeline the QEC workload [48, 49, 51, 74–85]. Although some processing of the syndrome measurement data can be carried out adjacent to the quantum layer, the amount of processing that can be done is subject to device constraints such as thermal budgets. Within this setting, several proposals have been made for two-level decoding schemes, including approaches where the first stage has been implemented either by sophisticated neural networks [76–79], or by lightweight decoders that correct the easy error configurations [80–83]. There also exist decoding

schemes that attempt to correct all errors using on-chip cryogenic hardware at some expense to the accuracy of the decoder [84–86].

In our approach, our local predecoder will always attempt to correct the error, and is designed to be accurate wherever the error is sufficiently sparse. For almost all error rates below threshold, up to $p < 1.8\%$, this results in a reduction to the defect density of the syndrome, with the defect density scaling quadratically in the error rate when predecoding is used, and a $1000\times$ reduction being attained at error rates $p = 10^{-3}$. The corresponding runtime of the main MWPM decoder is made to scale quartically in the error rate, instead of quadratically as for the case without predecoding. In the limit of large surface code patches and at error rate $p = 10^{-3}$, a resulting speed-up of a factor of 1900 is possible, with a $200\times$ speed-up realised at this error rate for the $d = 22$ code.

In assessing the cost of the scheme, we find that entropic factors and small-size effects are very important to consider when calculating the magnitude of logical failure rates [87]. As a result, the cost of the predecoder is modest for systems that we expect may be experimentally accessible in the near term, with only a 50% increase in the qubit count being required over a wide range of error rates and code distances. For smaller codes with $d < 9$, only a 15% increase is required.

The structure of this paper is as follows. In Section 2.2, we discuss fault-tolerant decoding on the surface code and define the error model that we will use. In Section 2.3, we develop the predecoding algorithm and its action on the syndrome history. In Section 2.4, we demonstrate evidence of a threshold and study sub-threshold failure probabilities. In Section 2.5 we present numerical results on the bandwidth reduction that is observed, and on a corresponding saving in the runtime of a MWPM main decoder. In Section 2.6, we discuss the performance of the predecoder over a practical regime. We identify combinatoric and finite-size effects that are important to consider and provide evidence that the asymptotic scaling of failure probabilities is a pessimistic estimator of performance in this regime. Finally, in Section 2.7, we conclude and discuss future work. In Appendix 2.A, we generalize the predecoder to a parameterized family of predecoders, allowing a smaller logical failure probability to be recovered by trading off some of the bandwidth usage and latency savings.

2.2 Fault-tolerant error correction on the surface code

Our scheme can be defined over any topological code with arbitrary boundaries. For concreteness, we choose here to focus on the rotated surface code with periodic boundaries, which allows a distance-d code to be encoded using d^2 physical qubits (see Fig. 2.1). Since the surface code is of Calderbank-Shor Steane (CSS) type, we can deal with the correction procedure for Pauli-Z and Pauli-X errors separately. We will concentrate on Pauli-Z type errors, but remark that an

equivalent discussion will hold for the Pauli-X errors due to the duality of the different types of stabilizer.

We demonstrate our decoding algorithm using an independent and identically distributed (i.i.d.) noise model that introduces phase-flip errors, and where stabilizer measurements may also give unreliable outcomes. This simplified noise model captures many of the phenomenological features of the gate-based noise model that reflects the physics of locally interacting hardware. We assume that all of the stabilizer generators of the code are measured simultaneously at a fixed frequency, whose clock cycle defines a natural unit of time. All qubits experience an i.i.d. dephasing noise channel, where the channel will introduce a Pauli Z error to a qubit with probability p per unit time, otherwise the qubit experiences no error. Further, we assume that the stabilizer measurements are imperfect, and that each measurement returns an incorrect outcome with probability p_m, and we set $p_m = p$.

In order to protect encoded information we need to perform active error correction. As we are considering faulty measurements, we cannot rely on any individual measurement outcome. The standard approach, which we employ here, is to repeat each stabilizer measurement d times in order to build up a (2+1)-dimensional set of outcomes referred to as the *syndrome history*. The syndrome history will serve as the input to the decoding algorithm, and an example syndrome history is shown in Fig. 2.1. A syndrome history variable is specified by an index v over X-type stabilizers, and a time t, and is denoted $s_v(t)$. In particular, $s_v(t)$ is defined as the parity of measurement outcomes of the X-type stabilizer associated to the index v taken at times $t, t-1$. Phase flip and measurement errors correspond to space-like and time-like edges in the syndrome history respectively, and string-like error configurations create defects at their endpoints.

A decoder takes this syndrome history and proposes a correction that will recover the encoded state with high probability. That is, a decoder is an algorithm whose input is a (2+1)-dimensional syndrome history and whose output is a Pauli operator that will return the code to a code state. The decoder fails when the net result of the error and the correction does not return the system to the same state that it started in, and an important quantity of interest for a decoder is its failure probability with respect to a given noise model.

We remark that with this definition, a decoder is not concerned with the possibility that a string of measurement errors will wrap non-trivially through the syndrome history in the time-like direction. Although temporal errors can lead to logical failures during the execution of fault-tolerant gates, the definition of logical failure that we use is well-suited to the situation where the quantum code is being used as a memory. Since the decoders we consider are isotropic in spacetime, the inclusion of temporal failures would not effect the results substantially, and would increase logical failure probabilities by a constant factor of at most 3/2.

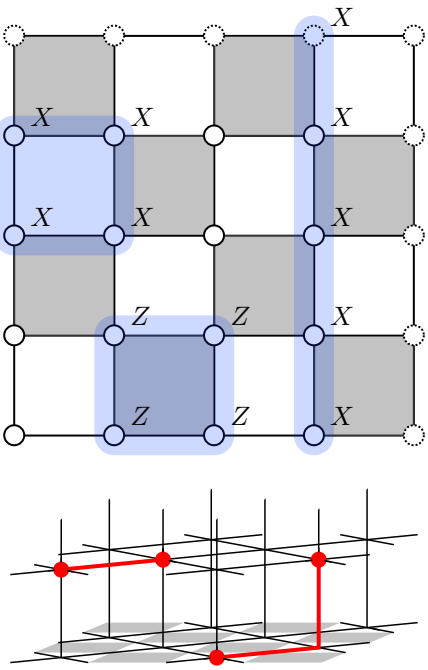


Fig. 2.1 (top) The distance-4 rotated surface code. Qubits are denoted by dashed or solid white circles. Periodic boundary conditions are enforced by identifying the dashed qubits along the top row with the qubits along the bottom row, and identifying the dashed qubits along the right column with the qubits along the left column. Pauli X-type (Z-type) stabilizers are given by the product of Pauli X-type (Z-type) operators over white (shaded) squares. One stabilizer of each Pauli type is shown. The surface code encodes two logical qubits into the shared $(+1)$-eigenspace of the stabilizers. Logical operators are given by the product of Pauli operators around a non-trivial cycle, and a Pauli X-type logical operator is shown. Logical operators commute with all the stabilizers, implying that they have a non-trivial action that preserves the logical code space. (bottom) The decoding lattice used for decoding phase-flip noise and faulty X-type stabilizer measurements. Only two rounds of stabilizer measurement are shown, although for a distance-4 code, four rounds of stabilizer measurement would be required. Space-like edges correspond to possible locations of phase-flip errors, and time-like edges correspond to possible locations of measurement errors. When an error occurs on an edge, whether time-like or space-like, a pair of defects is created on the vertices contained in the boundary of the edge. A single phase-flip error is shown, as well as an extended string-like error consisting of a phase-flip error and a measurement error. The string-like error creates defects at its endpoints.

2.3 Decoding and predecoding

In this work we propose the use of a local predecoder to reduce the bandwidth demands and latency for a global decoding algorithm, specifically, the MWPM decoder. We first briefly review the well-studied matching decoder, before introducing our predecoder.

2.3.1 Minimum-weight perfect matching

Decoding the surface code for a local noise model involves pairing nearby defects of the syndrome. This problem is particularly well suited for MWPM [42]; see Refs. [43, 88] for a review.

A perfect matching is a subgraph of some input graph such that each vertex of the output subgraph has exactly one incident edge. Efficient algorithms are known to find a MWPM for a graph, where the input graph has weighted edges, and the output graph is such that the sum of the weights of its edges is minimal [89, 90].

It is well-known that an algorithm for MWPM can be used to correct the surface code reliably [42]. In order to do so, we create the complete graph whose vertices correspond to defects in the syndrome history. Weights are assigned to edges that connect defect pairs in proportion to the probability that the defect pairs were created at the endpoints of one string-like error. For an i.i.d. noise model together with measurement errors at the same rate $p_m = p$, we have the simplification that weights are calculated from the taxicab metric on the syndrome history. The output of the MWPM algorithm is a choice of edges that matches each vertex to exactly one other vertex. From the output matching, a string-like Pauli operator can be computed that will restore the system to the code state.

Assuming the MWPM algorithm has prior information about the error model, the decoder requires a list of defects from the $(2+1)$-dimensional syndrome history. The latency and the bandwidth requirements depend on the number of defects from this volume, as follows. Firstly, the runtime of the MWPM decoder scales polynomially with the number of vertices of its input graph, which we denote by $|W|$. For instance, we use the Blossom V implementation [91], which returns matchings over complete graphs, and find a typical runtime scaling like $O(|W|^2)$. Second, the bandwidth requirements for communicating the syndrome to the decoder may also depend on the number of defects. Although sending the full record of all syndrome measurements depends only on the spacetime volume V of the syndrome history, this is very inefficient in the limit that the error rate is very small such that defects are sparsely distributed. Such a syndrome history can be compressed such that the length of the message is proportional to the number of defects produced by an error configuration, simply by sending a list of addresses where defects are identified. Given that we can describe the address of a defect in the spacetime volume V with $\log V$ bits, we have that the syndrome can be communicated with a message of

total length $O(\bar{\rho}V \log V)$ on average, where $\bar{\rho}$ is the mean density of defects in the syndrome history and we anticipate that $\bar{\rho} = 2p$. (Note that fluctuations around this average message size need to be considered, and we point the reader to Ref. [75] where this and other strategies for syndrome compression are considered.)

As we have argued, we can improve the latency and the bandwidth required to use a global decoder by minimising the number of defect locations that are communicated beyond the quantum device. The goal of predecoding is to attempt to correct small errors locally, and only to pass difficult error cases to the global decoder.

2.3.2 Predecoding

The predecoder is the first stage of a pipelined, two-stage decoding scheme. It can be understood as a map on syndrome histories that returns a modified syndrome history and a partial correction. The modified syndrome history then gets passed to a global decoder which returns a correction. The total correction for the code is given by combining the partial correction returned by the predecoder and the correction returned by the global decoder. Although it is useful to think of the predecoder as a map on syndrome histories, it is also important to keep in mind that it is implemented locally in space and time by a CA that can sit adjacent to the quantum layer. Additionally, the CA and global decoder can both operate concurrently, with the predecoder cleaning up the syndrome history as stabilizer measurements are taking place, and at the same time the global decoder is working on processing the syndrome history from the previous error correction round.

The predecoder is based on the greedy matching of nearest-neighbour defect pairs to reduce the defect density. Such an approach works well when combined with syndrome compression techniques to ensure that the bandwidth scales with the number of defects in the code. We remark that, due to the ease and parallelisable nature of nearest-neighbour matchings, they have attracted some attention recently. For example, the predecoder can be viewed as a truncated version of a HDRG decoder [61], and a similar algorithm has been used to compute correlations amongst errors for MWPM [83].

We now define the predecoder precisely. At each edge on the decoding lattice (space-like and time-like) we define an update rule that determines whether a matching should be made on that edge. In particular, a matching will be made if and only if the boundary of the edge is contained inside the syndrome. If the edge is time-like, the defects on the boundary are removed from the syndrome and no further action is taken. If the edge is space-like, the defects are removed from the syndrome and a phase-flip is recorded. It is only the parity of phase-flips that needs to be updated here, so only a single bit of local memory is required.

As described thus far, these update rules are not well-defined because we have not specified the order in which these matchings get carried out. To resolve this, we have that the matchings get carried out in one concurrent step. A matching at any edge in the syndrome history is always determined by the syndrome data prior to predecoding in the boundary of that edge. More formally, the rules define a map on syndrome histories that outputs a modified syndrome and a partial correction. The modified syndrome is given,

$$s'_v(t) = s_v(t) + s_v(t)\Big(\sum_{u\sim v} s_u(t) + s_v(t+1) + s_v(t-1)\Big), \tag{2.1}$$

where sums are taken modulo 2. The partial correction returned by the predecoder is a Pauli operator on the surface code. The partial correction on a qubit is determined by the parity of all matchings made on space-like edges in the syndrome history corresponding to that qubit. The partial correction can be specified by a bitstring $x_{(u,v)}$, where non-zero entries of $x_{(u,v)}$ indicate the presence of a correction. Specifically, we have:

$$x_{(u,v)} = \sum_t s_u(t)s_v(t), \tag{2.2}$$

where sums are taken modulo 2. One important quantity that characterizes the predecoder is its *isolation volume*, which we denote V_0. The predecoder will only accurately correct a single error if there are no other errors within a spacetime volume V_0 of the error. By drawing an isolated defect pair and by calculating the effect of other errors on nearby edges, we can compute $V_0 = 57$, and this is shown in Appendix 2.A. If error rates are large such that $pV_0 \gg 1$, then there are very few isolated defect pairs in the syndrome history and predecoding is ineffective. On the other hand, if error rates are sufficiently low that $pV_0 \ll 1$, then most defect pairs are isolated and predecoding can significantly reduce the bandwidth and latency of the scheme. We refer to these as the *high error-rate* and *low error-rate* regimes respectively. Importantly, we remark that the effectiveness of predecoding depends only on the error rate, and not on the code size. When predecoding does result in savings, these savings do not saturate for large code sizes.

We can now turn our investigation into the costs and savings of using the predecoder as the first stage in a two-stage decoding scheme. We will explore the asymptotic QEC performance in Sec. 2.4, the expected gains in latency and bandwidth in Sec. 2.5, and finally we will investigate how this predecoder is expected to function in practice for fault-tolerant quantum computing in Sec. 2.6.

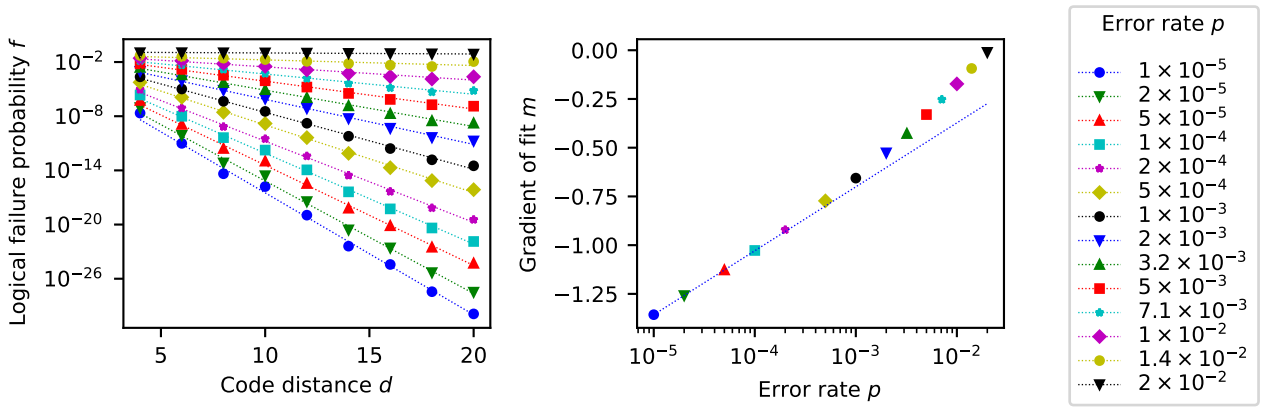


Fig. 2.2 (left) Failure probability as a function of code distance when the predecoder is used together with MWPM for a range of error rates. We observe that the failure probability decays exponentially in the code distance, indicating the presence of a threshold. (right) Gradients m extracted from a linear fit $\log(f) = md$ of the decay of the failure probability, as a function of physical error rate p. For small error rates ($p \lesssim 10^{-3}$), we find $m = k \log(p)$ for $k = 0.33(1)$.

2.4 Asymptotic performance

In this section, we investigate the expected performance of predecoding for large code families. We first demonstrate that failure probabilities are exponentially suppressed in the code distance for small error rates when predecoding is used, indicating the existence of a threshold for the scheme. We also compute the decay constant associated with the sub-threshold scaling of the failure probability that characterizes this suppression in the limit of large codes and small error rates. These should be understood as preliminary results that indicate the viability of the scheme as a FTQEC protocol. A more in-depth analysis of the costs and benefits of the scheme for real-time decoding applications are given in Secs. 2.5 and 2.6.

We follow Ref. [92] and use an ansatz for the sub-threshold scaling of the failure probability given by

$$f = C \left(\frac{p}{p_{\text{th}}} \right)^{kd} \tag{2.3}$$

where k is the decay constant, p_{th} is the threshold and C is just a fitting constant. This empirical ansatz captures the scaling of logical failure probabilities with the code distance. At the threshold error rate $p = p_c$, we have that the logical failure probability is independent of the code distance. For sub-threshold error rates $p < p_c$, the ansatz describes a logical failure probability that decays exponentially with the code distance.

For small error rates, the dominant failure mode for the scheme will be errors that have low weight. Then, the exponent kd can be interpreted as the weight of the least-weight error that results in a logical failure. We can make predictions for k for the scheme based on the

greediness of the matchings made by the predecoder, and verify these predictions by exploring logical failure probabilities at low error rates. For MWPM without predecoding, there exists string-like errors of weight-$d/2$ that can cause the decoder to fail, so we have $k = 1/2$ in that case. Consider taking such an error and removing a single phase-flip somewhere away from the boundary of the string. This error will now be correctly decoded by the MWPM algorithm. However, if we first run the predecoder over such an error, it will see a pair of adjacent defects where the phase-flip was removed and will re-introduce the phase-flip back into the code. If we follow that up with MWPM then the whole scheme will fail. This describes a failing error of weight-$(d/2 - 1)$. In fact, we can remove every third phase-flip from the bulk of an error string and the predecoder will re-introduce those phase-flips back into the code. This construction is similar to the Cantor set errors observed in hard-decision renormalization group (HDRG)-style decoders [62], and leads us to predict a decay constant $k = 1/3$. We also expect that the threshold of the scheme should not suffer drastically from predecoding. The threshold error rate for standard MWPM with phenomenological noise and without predecoding is $p = 2.9\%$ [57]. This error rate, which gives $pV_0 = 1.7$, is in the transition regime between low and high error-rates as defined in Sec. 2.3. We therefore expect that many of the corrections returned by the predecoder at this error rate are still accurate.

We use various numerical methods to extract the fitting parameters k, p_{th} in Eq. (2.3). In this and other simulations in this work, we used the qecsim software package [93], along with other scientific [94] and matching software [91] to perform numerical computations. For each code distance in the range $4 \leq d \leq 22$, we performed a direct Monte Carlo simulation to attain a failure probability at the error rate $p = 0.02$. In each case, the simulation was run until ten logical failures were observed. For smaller error rates, it was not feasible to use direct Monte Carlo simulation because it would take a prohibitively long time to observe any logical failures. Instead, we used the splitting method [87, 95, 96]. The splitting method allows us to compute the ratio of the failure probabilities at two different error rates by sampling from the set of failing errors at each error rate. We multiply these ratios together to attain logical failure probabilities at very small error rates, and we use the result of the direct Monte Carlo simulation at $p = 0.02$ as a reference. At each error rate, we collected 1000 independent samples via a Metropolis-Hastings algorithm and computed logical failure probabilities at error rates down to $p = 1 \times 10^{-5}$.

We plot the failure probability against the code distance in Fig. 2.2. The failure probability is exponentially suppressed in the code distance, indicating that the threshold is persistent when the predecoder is run in combination with MWPM. We can also determine that the threshold is approximately $p = 2\%$, since the logical failure probability is independent of code distance at this error rate. We compare this to the known threshold for MWPM under a phenomenological phase flip error model which is 2.9% [57].

We also extract the decay constant from our simulations in Fig. 2.2, and find $k = 0.33(1)$, consistent with our analysis above. We contrast this value against the decay constant for MWPM, which is $k = 0.5$. In particular, we remark that the value $k = 0.33(1)$ is extracted from a fit at small error rates $p < 5 \times 10^{-3}$. We find that closer to threshold, logical failure probabilities are suppressed more strongly with the code distance, as seen in Fig 2.2. We believe that this is because the number of least-weight failing errors under predecoding is relatively small, and they become less important at larger error rates. Despite being designed to correct sparse errors, we observe throughout this work that the predecoder is surprisingly accurate even near threshold, and in Sec.2.6 we discuss the combinatoric factors and finite-size effects that are responsible for this behaviour.

2.5 Bandwidth and latency

In this section, we study the effect that predecoding has on the defect density in a syndrome history, as well as on the runtime of the main MWPM decoder. As the syndrome is defined in a spacetime volume, the defect density can be thought of as the bandwidth per unit area of the quantum layer required to transmit the syndrome. The defect density also determines the size of the problem that must be solved by the main decoder. We provide the mean savings to the defect density and MWPM runtime that are realized in the low error-rate regime, and additionally provide the full defect count distribution since the worst-case bandwidth usage and runtime are also important quantities of interest for real-time decoding.

2.5.1 Defect count distribution

We first develop a heuristic model for the distribution of defect counts and treat the defect density as a parameter of this model. Let M be the random variable corresponding to the number of defects sent to the main MWPM decoder in one syndrome history of spacetime volume V. We do not yet specify whether predecoding is being used. We note that neither the noise model nor the predecoder introduce any long-range correlations into the syndrome data. If we coarse grain the syndrome history into suitably sized blocks, the total defect count is the sum of many independent binomial random variables corresponding to whether or not there is a defect pair in each block. The total expected number of defects is fixed by the defect density. Since the blocks can be made small, up to a size V_0, the total defect count can be modelled using a Poisson random variable. Technically, since the total defect number is constrained to be even by an emergent symmetry [41], it is really $M/2$ that is Poisson-distributed and we have

$$\frac{M}{2} \sim \text{Poisson}\left(\frac{\bar{\rho}V}{2}\right) \tag{2.4}$$

where the mean defect density $\bar{\rho}$ depends on the error rate, as well as whether or not predecoding is being used. It should be noted that Eq. (2.4) describes the number of defects in a full cycle of fault-tolerant error correction. The distribution that describes the bandwidth usage during each round of measurement is given by replacing V by V/t, where t is the number of rounds of stabilizer measurement and we expect to set $t = d$.

2.5.2 Defect density with and without predecoding

We now study the defect density under MWPM-only and under predecoding. For the case where no predecoding is performed, or in the high error-rate regime when predecoding is not useful, the bandwidth density is proportional to the error rate with a coefficient of two to account for the fact that a single fault creates two defects. We have

$$\bar{\rho}_{\text{no pre}} = 2p. \tag{2.5}$$

In the low error-rate regime and when predecoding is used, the defect density can be significantly reduced. The probability of two errors occuring within a given spacetime volume V_0 is given by $(V_0 p)^2/2$. Most of the error configurations that result in an inaccurate correction by the predecoder leave behind two defects in the syndrome history after predecoding, and this is shown in Appendix 2.A. These considerations lead to a defect density given by

$$\bar{\rho}_{\text{pre}} = p^2 V_0. \tag{2.6}$$

We verified these predictions numerically. We used direct Monte Carlo methods to estimate the defect density over a range of physical error rates when syndrome compression is used, and when syndrome compression and predecoding are used together. For each physical error rate, we computed the mean number of defects in the syndrome histories for codes of different sizes, averaged over at least 10000 decoding cycles, with more decoding cycles used for small error rates. We then fit a linear model in order to extract the defect density. In Fig. 2.3, we observe that the defect density is reduced by a factor that scales quadratically in the physical error rate, as predicted by Eq. (2.6). For small error rates $p = 10^{-4}$, the defect density is reduced by a factor 10^5 as compared to when no predecoding or syndrome compression is used. At more moderate error rates of $p = 10^{-3}$, we still see reductions of order 1000.

We make two observations about these bandwidth savings. First, the mean bandwidth savings do not depend on the size of the code, because when the predecoder encounters a difficult error configuration in the syndrome history, it will only be deterred from performing greedy matchings in a very small neighbourhood of that error. Second, bandwidth savings are made at

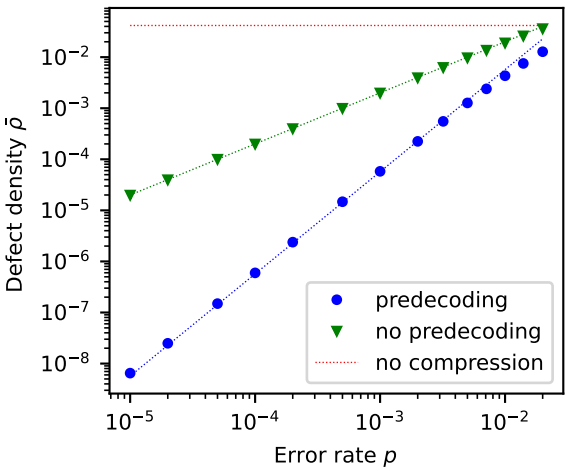


Fig. 2.3 Defect density as a function of error rate. The defect density determines the required bandwidth for transmitting syndrome information per unit area. When syndrome compression is used (green), the defect density scales linearly in the error rate. When the predecoder is also used (blue), the defect density scales quadratically in the error rate. The dotted green line corresponds to Eq. (2.5), and the dotted blue line corresponds to Eq. (2.6) with $V_0 = 57$. (red) The bandwidth usage associated with an uncompressed syndrome history is independent of the error rate. In order to make a fair comparison, we show density of vertices in the syndrome history divided by 16. The factor 16 assumes that addresses require 16 bits to transmit, whereas measurement data requires only one bit.

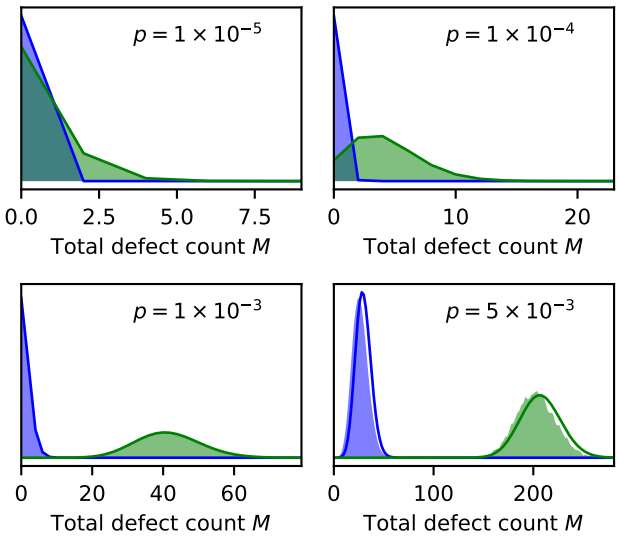


Fig. 2.4 Histograms of the number of defects contained in a distance-20 syndrome history with predecoding (blue) and without predecoding (green). The solid blue and green lines correspond to the random variables described by Eq. (2.4) for $\bar{\rho} = p^2 V_0$ and $\bar{\rho} = 2p$ respectively.

all error rates below threshold, because the low error-rate regime is set by $p = V_0^{-1} = 1.75\%$ and this value happens to be very close to the threshold for this scheme.

2.5.3 Worst-case performance for finite-size quantum computations

In a real device, it may be a requirement that the bandwidth usage never exceed some maximum value. If the distribution of bandwidth used in a decoding cycle is highly skewed towards large bandwidths, then the mean bandwidth saving is not a good metric for the efficacy of the scheme. Instead, the efficacy depends on the frequency with which large bandwidth-demanding events occur in the system.

We quantify this by following Ref. [80] and studying the top $(1 - f)$-percentile of the defect count, denoted $M_{\max}$ and defined by $P(M > M_{\max}) = f$, where we set $f = 10^{-15}$ as a target logical failure probability.

We collected data on the full histogram of defect counts generated by the Monte Carlo simulations. In Fig. 2.4 we compare this data to our heuristic model in Eq. (2.4) for a few select error rates and a spacetime volume corresponding to a distance $d = 20$ surface code.

When the spacetime volume is large, we can appeal to the law of large numbers. In that case, the observed defect count will be close to its mean value with high probability, on a logarithmic scale. For instance, for $\bar{\rho}V > 1000$, we have that $M_{\max}/\bar{\rho}V < 1.4$. In that case, we may freely replace $\bar{\rho}$ with $M_{\max}/V$ in Fig. 2.3, and the error bars would not be visible.

For small values of $\bar{\rho}V/2$, the distribution can be seen to be skewed towards high bandwidths, and this will slightly erode the savings to the bandwidth usage that are realized. This effect can be straightforwardly calculated using properties of the Poisson distribution. We emphasize that provided that multiplexed resource-sharing techniques are used [75, 80], the defect count can be averaged over a spacetime volume that corresponds to the syndrome histories for all logical qubits sharing a line of communication with the main decoder. This results in a bandwidth distribution with smaller variation than would be observed for a single logical qubit.

We remark on the particular applicability of this scheme to performing entangling gates in quantum computations. Certain schemes for performing entangling gates require that different patches of surface code be fused together to form bigger patches of surface code. It may then be the case that we require fault-tolerant quantum error correction to be carried out over a large surface code patch. This will be difficult if the bandwidth savings saturate at large code sizes due to any global conditionals in the predecoding algorithm. The local and greedy nature of the CA predecoder is essential to it being able to reduce the density of defects, and it lends itself naturally to this setting.

2.5.4 MWPM runtime and latency

The reduction to the defect density translates directly to a speed-up in the runtime of the global decoder. In the implementation of MWPM that we use [91], the worst-case runtime when applied to complete graphs is $O(|W|^3)$, where $|W|$ is the number of vertices. However, significant effort has gone into ensuring that this algorithm runs faster in most cases. Matching algorithms based on complete graphs generated by quantum error correction have been reported to have more favourable scaling in practice than their worst-case behaviour would suggest [97]. In the inset to Fig. 2.5, we demonstrate that the runtime is well described by $\mathrm{RT} \sim |W|^2$.

We can determine from Eqs. (2.5) and (2.6) that the factor speed-up in the MWPM decoder is expected to depend on the error rate as

$$\frac{\mathrm{RT}_{\mathrm{pre}}}{\mathrm{RT}_{\mathrm{no\ pre}}} = \left(\frac{pV_0}{2}\right)^2 \tag{2.7}$$

We can understand this speed-up by noting that this MWPM implementation requires a complete graph to be constructed for every error configuration (although cf. [59]). This implementation does not exploit the fact that isolated errors are easier to correct than large clusters of errors. The runtime savings from predecoding arise by matching isolated pairs of defects greedily at the start so they do not need to be included in the complete graph. In Fig. 2.5, we demonstrate the speed-up in the limit where decoding is performed over a large spacetime volume. At error rates $p = 10^{-3}$, we observe that the MWPM decoder is sped up by a factor of 1900. At error rates $p = 10^{-4}$, we observe a speed-up by a factor of 3.1×10^5.

There are two important caveats to note here. First is that these speed-ups are only fully realised when the spacetime volume of the syndrome history is large. Specifically, the speed-up arises only if there are enough defects in the syndrome history that the runtime of the MWPM decoder is dominated by its asymptotic behaviour. For the implementation of MWPM that we use, this is the case when there are more than 50 defects in the syndrome history. The relevant spacetime volume here is the spacetime volume corresponding to a connected patch of surface code that is being decoded as one logical unit. If there are not more than 50 defects in the syndrome history both with and without predecoding, then the runtime savings reported above are not fully realized. At error rate $p = 5 \times 10^{-3}$ and code distance $d = 30$ (sufficient to reach logical failure probability $f = 10^{-15}$) this condition is met. A 64× runtime speedup is realized, and this is the full speed-up attainable at that error rate. At smaller error rates, this condition is unlikely to be met when using a reasonably-sized surface code as a memory, but may be met when decoding over a large surface code patch, for example when performing an entangling gate. Nonetheless, at error rates $p = 10^{-3}$ and code distances $d = 22$ (sufficient to reach logical failure probability $f = 10^{-15}$), a direct comparison of runtimes still revealed a speed-up by a factor of 200.

The second caveat is that the worst-case runtime is going to depend on the tail end of the full defect distribution. This discussion is equally relevant to determining the worst-case runtime for the MWPM under predecoding. If the worst-case runtime is significantly larger than the mean runtime, then the mean runtime can be recovered by using multiplexed resource-sharing techniques [75, 80].

2.6 Predecoding in practice

The diminished decay constant $k = 1/3$ in the logical failure probability scaling of Eq. (2.3) associated with the predecoder, compared with $k = 1/2$ for MWPM, means that larger codes will be needed to achieve a target logical failure probability. We note, however, that this scaling coefficient is describing the asymptotic performance and may not be capturing the relative performance of the predecoding approach for practical code sizes. We now investigate the costs associated with using the predecoder in regimes that are relevant to fault-tolerant quantum computing.

2.6.1 Practical qubit cost of predecoding

In this section we develop an ansatz that explicitly includes entropic and finite-size effects. In Fig. 2.6, we compare the code distance required to reach target logical failure probabilities of $f = 10^{-15}$ both with and without predecoding. We find that, in order to accommodate

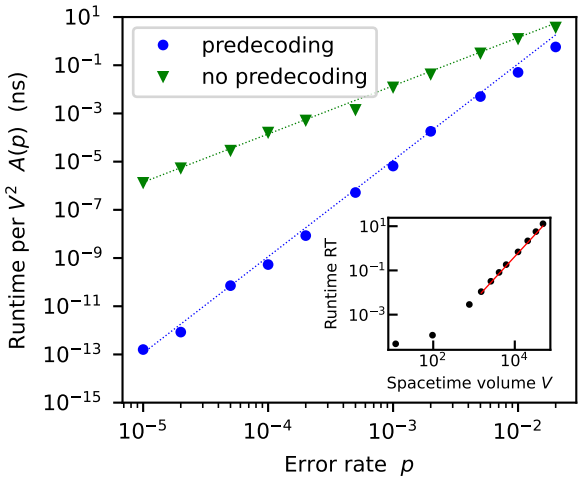


Fig. 2.5 For each error rate, we plot the constant of proportionality extracted from a fit of runtime $RT = A(p)V^2$, where V is the spacetime volume of the syndrome history and the runtime is for the global matching part of the decoding scheme, and is given in nanoseconds. (blue) Predecoding is used. Markers are data points, and the dotted line is the fit $A(p) = A(V_0 p/2)^2 p^2$, for $A = 1.4 \times 10^{-4}$. (green) No predecoding is used. Markers are data points, and the dotted line is the fit $A(p) = Ap^2$ for the same $A = 1.4 \times 10^{-4}$. (inset) we demonstrate the goodness of the fit $RT = A(p)V^2$ (red) to data (black) for the example case of no predecoding at error rate $p = 2 \times 10^{-2}$.

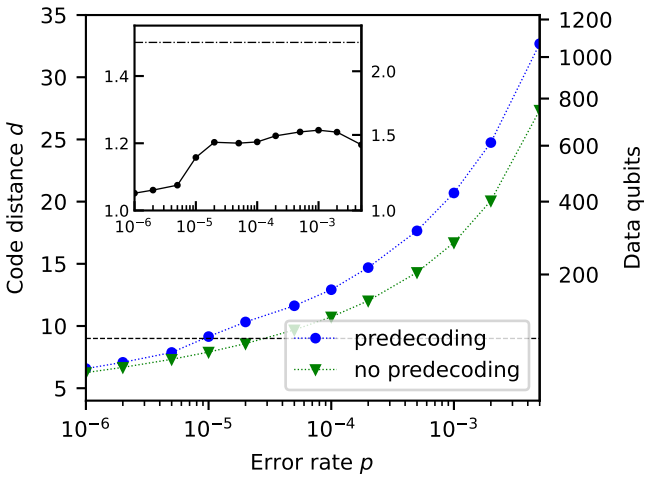


Fig. 2.6 (main) The code distance (left axis) and number of qubits (right axis) required to reach a target logical failure probability of $f = 10^{-15}$ when predecoding is used (blue) and when predecoding is not used (green). Distances are calculated by inverting Eq. (2.8), and using ansätze Eq. (2.9) and Eq. (2.10), (2.11). The dashed line marks $d = 9$, below which predecoding performs comparably to pure MWPM. (inset) the factor increase in the code distance (left axis) and number of data qubits (right axis) required with predecoding compared to without predecoding. The dashed-dotted line denotes the expected behaviour using Eq. (2.3) with $k = 1/3$.

the predecoding scheme, we must increase the qubit count by no more than 50%. We also remark that as error rates become very small, there is no qubit cost associated with running the predecoder. The decay constant $k = 1/3$ is therefore a pessimistic indicator of performance (it would suggest that the qubit count need be increased by a factor $2.25\times$). Importantly, we remark that Fig. 2.6 was generated by an ansatz we develop throughout this section, and not by Eq. (2.3).

2.6.2 Structure of the failing error set: weight and multiplicities

To see why the increase in overhead is modest, we study in detail the structure of the failing error set for small codes, which reveals a more complex resource cost for predecoding.

First, we note the importance of explicitly including entropic factors associated with the number of errors that result in a logical failure (see [42, 87]). The reason for this is that, at modest system sizes, we find that the multiplicity of low-weight errors is relatively small. These multiplicities impact significantly on the leading order scaling due to the contribution from low-weight logical errors. A second consideration is that the decay constant to fit Eq. (2.3) assumes that the weight of the least-weight failing error is linear in the code distance. Although this assumption is usually justified, in our case this is not strictly true for small codes due to finite size effects that we will explore.

To capture these effects, we make use of the fact that the weight and multiplicity of the least-weight failing error often provides much of the crucial information required in QEC for decoding moderately sized codes [63]. That is, we can construct an ansatz for the logical failure probabilities based on these data that will be relevant for a practical regime. We denote by $\mathrm{lw}(d)$ the weight of the least-weight error that results in a logical failure, and we denote by $A(d)$ the multiplicity of such errors. We emphasize that both of these quantities depend not only on the code, but also on the decoding scheme that is used. In particular, we have generally that $\mathrm{lw}(d) \leq d/2$, with the equality satisfied in the case of MWPM without predecoding. We will use subscripts where necessary to indicate the decoding scheme. The failure probability of the code is then given:

$$f = A(d)p^{\mathrm{lw}(d)}. \tag{2.8}$$

We now investigate the minimum weight of such failing errors, followed by the multiplicity of these errors.

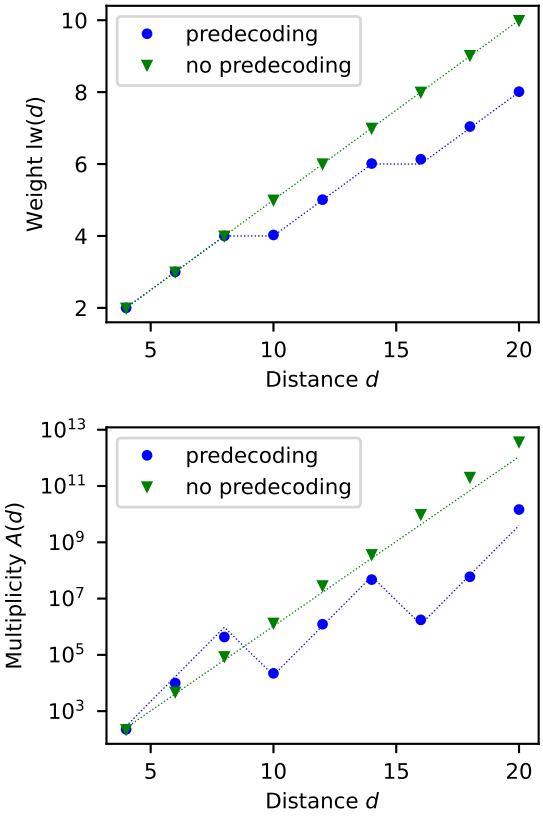


Fig. 2.7 (top) The weight of the minimum weight failing error, as a function of code distance. The solid lines are analytic and are given by by Eq. (2.9) for the case of predecoding (blue), or by $d/2$ for MWPM only (green). Data points are extracted as the gradients of linear fits of the log failure probability to the log error rate (see Eq. (2.8)). (bottom) Multiplicity of least-weight failing errors as a function of code distance. Data points are extracted as the intercepts of linear fits of the log failure probability to the log error rate (see Eq. (2.8)). The dashed green line is a fit for MWPM of form Eq. (2.11). The dashed blue line is a fit of form $A(d) \sim 2^{6a-2+\alpha b}$, where $d = 6a + b - 2$ for non-negative integer a and non-negative integer $b < 6$ and fitting parameter α. When $b = 0$, the scaling of $A(d)$ is described by Eq. 2.10, and this occurs here for distances $d = 4, 10, 16$.

2.6.3 Minimum weight of failing error

If we follow the prescription in Sec. 2.4 for constructing least-weight failing errors, then taking into account finite size effects leads to an ansatz:

$$\mathrm{lw}_{\mathrm{pre}}(d) = \left\lceil \frac{2}{3}\left(\frac{d}{2}+1\right)\right\rceil \tag{2.9}$$

where $\mathrm{lw}_{\mathrm{pre}}(d) = kd$ for $k = 1/3$ is recovered for large codes. We plot our data with the ansatz in Fig. 2.7 to show the correlation between our model and our results. Since we have generally that $\mathrm{lw}(d) > d/3$, we remark that $k = 1/3$ is overly pessimistic for predicting logical failure probabilities. It is particularly significant however that for $d < 9$ we have $\mathrm{lw}(d) = d/2$. We therefore expect the predecoder to perform comparably to MWPM for code sizes $d < 9$, with performance differences arising only from combinatoric factors. In Fig. 2.8, we plot the code distance required for a MWPM-only decoder to reach the same logical failure probability as when predecoding is used. We call this quantity the effective MWPM distance and denote it d'. We can see that the effective MWPM distance approaches the actual code distance for $d < 9$ and low error rates, indicating that predecoding comes at almost no cost in this regime.

2.6.4 Multiplicities of failing errors

Another important structural feature of the failing error set under predecoding is that there are very few least-weight failing errors compared to higher-weight errors. This can be easily visualized by considering some weight-d logical operator on the surface code. A weight-$d/2$ error that will cause a logical failure under MWPM can be constructed by placing a phase-flip on any $d/2$ data qubits in the support of the logical. There are approximately $\binom{d}{d/2} \lesssim 2^d$ of making this selection. On the other hand, with predecoding in place a weight-$d/3$ error will only result in a logical failure if it is carefully constructed by taking a length $d/2$ contiguous string of affected qubits, and then removing every third qubit from the support of the error. This means that there are far fewer least-weight failing errors under predecoding.

We can formulate these statements into an ansatz for the multiplicites of failing errors with and without predecoding. In both cases, there is a factor 2^d that arises by counting the number of logical operators in the surface code (see [87] for an explicit calculation of these combinatoric factors). For the case of MWPM, there is an additional factor 2^d due to the arguments above. This gives the rough ansatz for the multiplicites

$$A_{\mathrm{pre}}(d) \sim 2^d = 2^{3\mathrm{lw}_{\mathrm{pre}}(d)-2} \tag{2.10}$$

$$A_{\mathrm{no\ pre}}(d) \sim 4^d = 2^{4\mathrm{lw}_{\mathrm{no\ pre}}(d)} \tag{2.11}$$

where we have ignored the ceiling function in the second equality in Eq. (2.10). We remark that strictly speaking, the above ansatz applies to predecoding only on code distances where the ceiling function in Eq. 2.9 can be ignored. In a sense, this captures a situation where we can only just form a failing error of weight $\mathrm{lw}_{\mathrm{pre}}(d)$. We give a more sophisticated ansatz in Fig. 2.7 and also compare these ansätze to numerical data.

To demonstrate the effect that these entropic factors can have on the scaling of logical failure probabilities, we consider the probability that decoding will fail as a result of an error of weight $\mathrm{lw}_{\mathrm{pre}}$, as compared to an error of weight $\mathrm{lw}_{\mathrm{no\ pre}}$. Using Eq. (2.8), along with Eqs. (2.10), (2.11), we have

$$\frac{A_{\mathrm{pre}}(d)p^{\mathrm{lw}_{\mathrm{pre}}(d)}}{A_{\mathrm{no\ pre}}(d)p^{\mathrm{lw}_{\mathrm{no\ pre}}(d)}} \sim \left(\frac{1}{2p^{1/6}}\right)^{d}. \tag{2.12}$$

For $p > 0.015$, this predicts that the scaling of logical failure probabilities with code size will be dominated by the more common weight $d/2$ errors, not by the much rarer weight $d/3$ errors.

In practice, a comparison that is relevant for assessing decoding performance for finite-size codes is to fix the weight of the least-weight failing error and then compare the multiplicities under predecoding to the multiplicities without predecoding. The second equalities above allow us to make that comparison and indicate that the multiplicities are more favourable when predecoding is performed for moderately sized codes. The distance-16 surface code under predecoding highlights this point. Since we have that $\mathrm{lw}_{\mathrm{pre}}(16) = \mathrm{lw}_{\mathrm{no\ pre}}(12)$, it would be reasonable to expect that predecoding over a distance-16 surface code should perform similarly to pure MWPM over a distance-12 surface code. In fact, from Fig. 2.8, we see that predecoding over a distance-16 surface code outperforms pure MWPM over even a distance-13 surface code due to the favourable multiplicities under predecoding. This effect becomes more pronounced for larger code sizes.

We remark that in a practical regime set by some fixed target logical failure probability, the exact form of $\mathrm{lw}_{\mathrm{pre}}(d)$ and the effect of multiplicities are both important to consider. Which effect is more important depends on the underlying error rate. If error rates are low, then large codes are not required and the precise form of $\mathrm{lw}_{\mathrm{pre}}(d)$ for small codes must be considered. On the other hand, if error rates are high, then large codes are required. The multiplicities of least-weight failing errors becomes a significant effect that must be accounted for.

In this and the previous section we have developed numerical methods and ansätze that have allowed us to understand the bandwidth usage and runtime savings resulting frome predecoding, as well as the logical failure probabilities. We go on in Appendix 2.A to extend this analysis to a parameterized family of predecoders that interpolates between the predecoder studied here, and an implementation of a standard MWPM decoder, and we detail a trade-off between the resource savings and the accuracy of the predecoder.

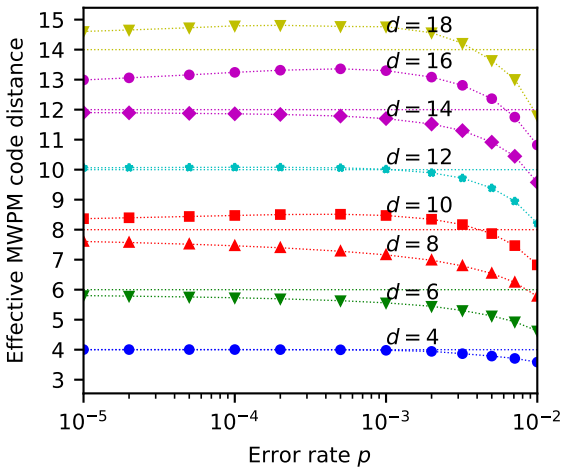


Fig. 2.8 Effective MWPM code distance of the predecoding scheme as a function of error rate, plotted for code distances $d = 4, 6, 8, 10, 12, 14, 16, 18$. Solid lines are color-coded to correspond to dashed lines that provide the low error rate asymptotic behavior, as determined by the weight of the least-weight failing error and given by Eq. (2.9).

2.7 Discussion

We have proposed a two-level decoding architecture for the surface code based on a local predecoder that makes greedy matchings designed to clean up sparse errors. Our numerical simulations demonstrate that the predecoder reduces the complexity and runtime of the global matching decoder's task, and also delivers substantial savings in the bandwidth density. Remarkably, despite the predecoder's intended design for low error rates, we find that these advantages persist up to threshold error rates. These order of magnitude improvements come at a modest increase in the qubit overhead to recover a commensurate logical failure probability compared with global approaches to decoding. Our results therefore show that augmenting our global decoder with a local predecoding system can help to address many of the practical challenges involved with producing a large-scale quantum computing architecture.

We conclude by identifying some future directions for new research in this area. First, it has been observed that quantum error-correcting systems can be markedly improved by specialising a decoder to correct for the errors of a tailored code [98–101]. It will be interesting to find more sophisticated predecoders that account for more realistic noise models. Furthermore, we expect that we can design predecoders that are sensitive to hook error syndromes [98], or perhaps flag stabilizer readings [102, 103], that indicate where a circuit noise model may introduce a correlated error. We also note that a variant of the predecoder presented here can be implemented *within* the quantum layer with Toffoli gates and qubit reset, offering the prospect of predecoders requiring no mid-circuit measurement or classical processing.

Lastly, it will be interesting to determine the role of predecoders as quantum technology develops. In our work, we have proposed a very greedy decoder designed to significantly reduce the bandwidth cost at the expense of logical failure probability. We contrast this with the approach of Ref. [80], wherein a fast all-or-nothing decoder was studied that looks for a globally optimal correction for certain errors, and calls for support from the global decoder where this is not possible. Where the latter approach maintains a better logical failure probability, the former demonstrates good performance for minimising the bandwidth cost, and particularly for noisier near-term devices. Ideally, we may want to balance these considerations, and in Appendix 2.A we study a method to interpolate between these two types of predecoders. Given that recent experiments have demonstrated classical feedforward together with mid-circuit measurements [22, 28, 30, 47], we are already in a position to begin to design predecoder experiments with real experimental hardware, to begin addressing these questions.

Acknowledgements

This work is supported by the Australian Research Council via the Centre of Excellence in Engineered Quantum Systems (EQUS) project number CE170100009, and by the ARO under Grant Number: W911NF-21-1-0007. Access to high-performance computing resources was provided by the National Computational Infrastructure (NCI Australia), an NCRIS enabled capability supported by the Australian Government, and the Sydney Informatics Hub, a Core Research Facility of the University of Sydney. BJB is grateful for the hospitality of the Center for Quantum Devices, at the University of Copenhagen where parts of this work were completed. BJB is now affiliated with IBM Quantum.

2.A Recovering the subthreshold failure rate

If the performance reduction associated with using the predecoder is deemed too costly, then it is possible to trade off decoding accuracy against bandwidth and runtime savings by reducing the greediness of the matchings made by the predecoder. We do this by requiring that the predecoder check that a particular defect pair is sufficiently well-isolated from any other defects in the code before performing a correction.

More formally, we introduce a family of predecoders parameterized by a radius r, representing this isolation distance. This family interpolates between the case $r = 0$, which was studied in the main text, and the case $r = \infty$, which is an implementation of the standard global MWPM decoding algorithm. To define this family, we introduce a projector that is equal to

one whenever a particular defect in a defect pair is well-isolated:

$$\Pi_r(v,t) = \begin{cases} 1 & \text{if } \sum_{(u,t')\in B_r(v,t)} s_u(t') \leq 2 \\ 0 & \text{otherwise} \end{cases}, \tag{2.13}$$

where $B_r(v,t)$ is a ball with centre (v,t) and radius r on the decoding lattice, defined with respect to the taxicab metric. Then, writing $\tilde{s}_v(t) = \Pi_r(v,t)s_v(t)$, we have that the action of the radius-r predecoder results in a modified syndrome:

$$s'_v(t) = s_v(t) + \tilde{s}_v(t)\left(\sum_{u\sim v} \tilde{s}_u(t) + \tilde{s}_v(t+1) + \tilde{s}_v(t-1)\right), \tag{2.14}$$

where sums are taken modulo 2. Further, the total correction is specified by a bitstring as in Eq. (2.2), which is now given as

$$x_{(u,v)} = \sum_t \tilde{s}_u(t)\tilde{s}_v(t), \tag{2.15}$$

where sums are taken modulo 2. We roughly sketch how these rules can be implemented using CA with only local communications. The CA will attempt to correct errors according to the rules developed in Sec. 2.3, with a modification. When a CA sees a defect at a particular time, it sends out its address to nearby CA up to a distance r from itself. With this message passing in place, a CA will not immediately make a correction when it sees a pair of neighbouring defects. Instead, it will wait r syndrome extraction cycles in order to gather more information. If in that time it receives a signal indicating some nearby defect other than the defect pair being corrected, it will not make the correction. These CAs require some additional memory and processing overheads in order to store and propagate addresses.

The key parameter that characterizes a predecoder in this family is its isolation volume, denoted by V_r. This determines both the accuracy of a given predecoder, as well as its bandwidth and runtime savings in terms of a reduction to the defect density. The isolation volume can be computed as in the main text for any r, and we show this in Fig. 2.9. We fit a cubic polynomial to this data for $r = 1, 2, 3$ and derive

$$V_r = 4(\tilde{r}+1)^3 + 6(\tilde{r}+1)^2 + 1 \tag{2.16}$$

where $\tilde{r} = \max(r, 1)$. We remark that $\tilde{r}$ appears instead of r because there is a sense in which the $r = 0$ predecoder is singular in this family of predecoders, as discussed in Fig. 2.9.

We follow similar methods as in the main text to study the distribution of defect count under the scheme for different values of the isolation radius r. The distribution is Poisson distributed, as in Eq. 2.4, however the defect density that appears now depends on the isolation

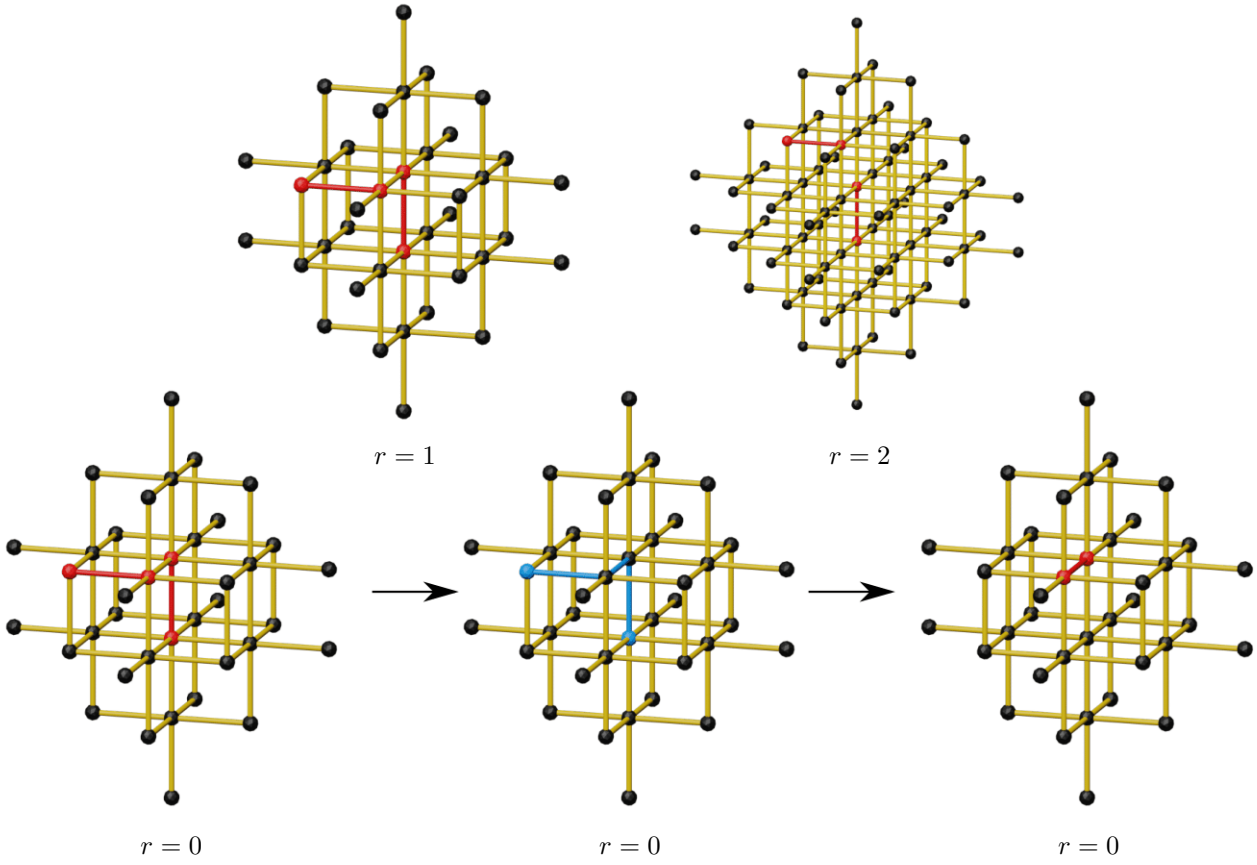


Fig. 2.9 Isolation volume of the predecoder for $r = 0, 1, 2$. Black balls correspond to vertices of the syndrome history, and sticks correspond to edges. For the case $r = 1$, $r = 2$, the red vertical stick depicts a measurement error that has occurred. Another error on any yellow stick will produce a defect that is within a distance r from the defects produced by the measurement error, and will lead to an error configuration that will not be corrected by the predecoder. Examples of such uncorrectable error configurations are shown in red. In this case, four defects will be left behind in the syndrome history. By counting the yellow sticks, we determine $V_1 = 57$ and $V_2 = 163$. The $r = 0$ case requires a special treatment. We again have a red vertical stick depicting a measurement error, and yellow sticks represent sites of possible errors that would result in error configurations that are not fully corrected by the predecoder. However, the mechanism by which this occurs is slightly different, and at the bottom left we show an error configuration that cannot be fully cleaned up by the predecoder. The predecoder attempts to place a correction between all neighbouring defects, shown in blue at bottom centre, however this introduces a correction at a site at which there was no error initially, shown at bottom right. As such, two defects remain in the syndrome history. In this case, we have $V_0 = V_1 = 57$, but since here most weight-2 errors result in only two defects left in the lattice, we have that Eq. (2.17) contains an additional factor 2 as compared to Eq. (2.6).

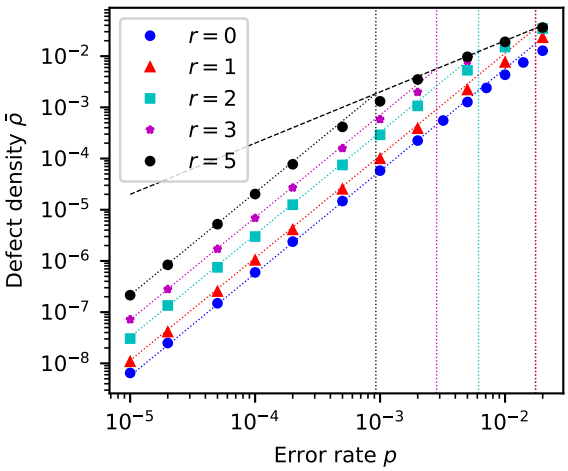


Fig. 2.10 The defect density associated with predecoders for $r = 0, 1, 2, 3, 5$. The black dashed line corresponds to MWPM-only with syndrome compression. The markers are data points, and the dotted lines correspond to Eq. (2.6) for $r = 0$, and to Eq. (2.17) for $r > 0$, where V_r is determined by Eq. (2.16). All predecoders exhibit bandwidth scaling with p^2 for sufficiently low-error rates. Vertical lines can be traced to the bottom axis to determine the error rate at which bandwidth savings take effect for a given r.

radius. In particular, if $r > 0$, we replace Eq. 2.6 with

$$\bar{\rho}_{\text{pre}}(r) = 2\rho^2 V_r \tag{2.17}$$

We remark that the defect density scales quadratically in the error rate for all values of r, provided that the error rate is low enough. However, since Eq. (2.17) is only valid for $p < 1/V_r$, the error rate at which these bandwidth savings take effect is dependent on r. As we increase r, the predecoder is more cautious and requires defect pairs to be more isolated before it will make a greedy matching. This means that error rates must be lower before the defect density is reduced. We compare this model to our numerical data in Fig. 2.10.

The increased caution taken by the predecoder for $r > 0$ also means that it is less likely to introduce erroneous corrections that cause the scheme to fail. For instance, a least-weight error that will cause the predecoder of radius r to fail can be constructed by taking a weight-$d/2$ contiguous error that fails under pure minimum-weight perfect matching and removing every $(\tilde{r}+1)/(\tilde{r}+2)$ phase flip. We generalize Eq. (2.9) and write

$$\text{lw}_{\text{pre, r}}(d) = \left\lceil \frac{\tilde{r}+1}{\tilde{r}+2}\left(\frac{d}{2}+1\right)\right\rceil \tag{2.18}$$

where $\tilde{r} = \max(r, 1)$. The qubit cost associated with running the predecoding scheme will therefore decrease with r. In the limit of large code sizes and ignoring finite size effects, we have

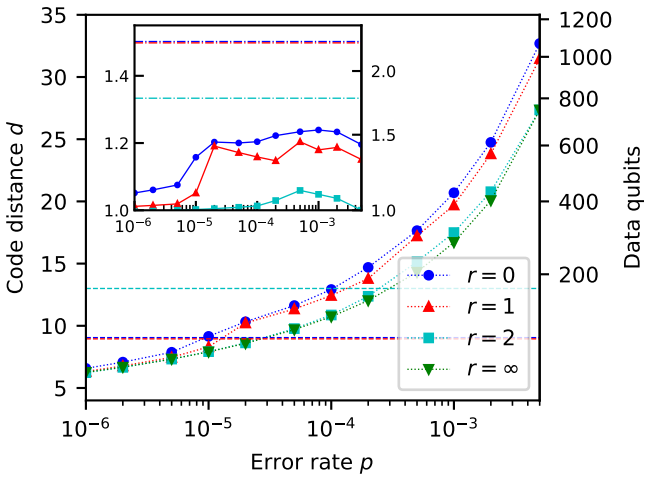


Fig. 2.11 (main) The code distance (left axis) and number of qubits (right axis) required to reach a target logical failure probability of $f = 10^{-15}$ for predecoders of radius $r = 0, 1, 2, \infty$, where $r = \infty$ indicates that no predecoding was performed. Distances are calculated by inverting Eq. (2.8), and using ansatz Eq. (2.18) along with a similar ansatz to that described in Fig. 2.7 for the multiplicities. These ansätze are linear fits to the logical failure probability at low error rates. The dashed lines are color-coded and mark the maximum distance such that $\mathrm{lw}_{\mathrm{pre,\,r}}(d) = d/2$, below which the predecoder of radius r are expected to perform comparably to pure MWPM. (inset) the factor increase in the code distance (left axis) and number of data qubits (right axis) required with predecoding compared to without predecoding. The dashed-dotted lines are color-coded and mark the expected behaviour using Eq. (2.3) with $k = (\tilde{r} + 1)/(\tilde{r} + 2)$.

that the weight of the least-weight failing error is linear in the code distance, with decay constant $k = (\tilde{r}+1)/(2(\tilde{r}+2))$. For large r, the predecoder makes very few greedy matchings, and the decay constant approaches the $k = 1/2$ that characterizes failure probabilites for MWPM. Of course, for predecoding in practice, finite size and combinatoric effects are important, and this can be seen in Fig. 2.11. In particular, we see that for the $r = 2$ predecoder and for code distances $d < 13$, there is almost no qubit cost associated with running the predecoder at low error rates.

Real devices are subject to multiple constraints simultaneously, and we may want to reduce the defect density to some maximum allowed value whilst keeping the logical failure probability as low as possible. This family of predecoders allows this trade-off by interpolating between pure MWPM, which provides a high decoding accuracy, and the maximally greedy $r = 0$ predecoder, which provides large savings to the bandwidth usage and runtime.

2.B Local implementation of predecoder

In this section, we provide a sketch of a local implementation of the predecoder update rules. We envision using simple predecoder modules that each have access only to local syndrome information, and are also capable of nearest-neighbour message passing.

In order to decode bitflip errors, say, we imagine a predecoder module for every Z-type stabilizer in the code. The predecoder modules are equipped with some small amount of memory for storing recent measurement outcomes and local corrections, and additionally in each gate cycle the modules are equipped with the following capabilities:

- Receiving the new measurement outcome from the relevant stabilizer.
- Broadcasting a single bit to all neighbouring predecoder modules.
- Communicating to a global decoder in some cases, although the predecoders are designed so that this is rare.

We provide a pseudocode implementation of the predecoder module in Algorithm 1, and in Figure. 2.12 we illustrate the predecoder module in action, implementing the predecoder update rules defined in Eqs. (2.1), (2.2).

Algorithm 1: The predecoder module. Note that in this implementation, a local correction on a qubit is recorded once for each predecoder module attached to a stabilizer that contains that qubit in its support. These corrections will always agree and should be included into a Pauli frame when a correction is applied. In general, flips due to measurement errors do not need to be recorded by the predecoder modules, except in certain circumstances such as during the final round of measurements in a lattice surgery operation.

Internal State : syndrome data $(s_{\text{new}}, s, s_{\text{old}})$, measurement data (m), round (t), corrections $(c_{\text{n}}, c_{\text{e}}, c_{\text{s}}, c_{\text{w}})$.
Constants : address (v)

Function `MainLoop` **is**
  Wait for new measurement outcome (m_{new})
  $t \longleftarrow t + 1$
  $s_{\text{old}} \longleftarrow s$
  $s \longleftarrow s_{\text{new}}$
  $s_{\text{new}} \longleftarrow m + m_{\text{new}} \pmod 2$
  $m \longleftarrow m_{\text{new}}$
  `BroadcastAllNeighbours(`s`)`

  Wait for syndrome data from neighbours $(s_{\text{n}}, s_{\text{e}}, s_{\text{s}}, s_{\text{w}})$
  $s' \longleftarrow s + s(s_{\text{old}} + s_{\text{new}} + s_{\text{n}} + s_{\text{e}} + s_{\text{s}} + s_{\text{w}}) \pmod 2$
  for i *in* $\{n, e, s, w\}$ **do**
    $c_i \longleftarrow c_i + ss_i \pmod 2$
  end
  if $s' \neq 0$ **then**
    `SendToGlobalDecoder(`v, t`)`
  end
end

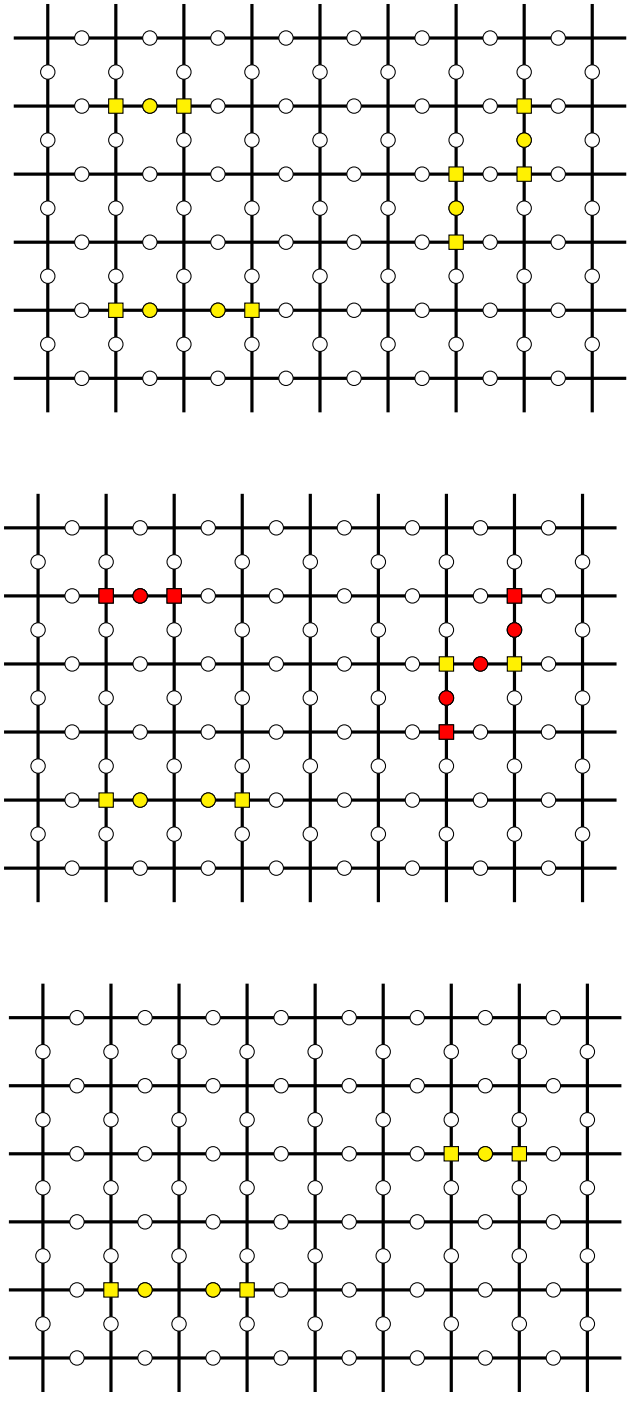


Fig. 2.12 The predecoder in action. For clarity of presentation, we show only one round of syndrome extraction, assuming perfect measurements and trivial syndrome information in all other rounds. In the top panel, yellow circles are qubits that have experienced an error, and yellow squares are predecoder modules with non-trivial syndrome information, corresponding to $s = 1$ and $s_{\text{new}} = s_{\text{old}} = 0$ by line 9 in Algorithm 1. In the middle panel, red squares are predecoder modules that have flipped their syndromes, corresponding to $s(s_{\text{old}} + s_{\text{new}} + s_{\text{n}} + s_{\text{e}} + s_{\text{s}} + s_{\text{w}}) = 1$ in Algorithm 1. Red circles are qubits in the support of a local correction, corresponding to, for instance, $c_n = 1$ for a predecoder module south of the qubit after line 13 of Algorithm 1. Finally, in the bottom panel, yellow squares are predecoder modules with syndrome information to be passed to a global decoder, corresponding to $s' = 1$ after line 11 in Algorithm 1. Yellow circles are qubits with as-yet uncorrected errors.

3

Mitigating errors in logical qubits

Abstract

Quantum error correcting codes protect quantum information, allowing for large quantum computations provided that physical error rates are sufficiently low. We combine post-selection with surface code error correction through the use of a parameterized family of *exclusive decoders*, which are able to abort on decoding instances that are deemed too difficult. We develop new numerical sampling methods to quantify logical failure rates with exclusive decoders as well as the trade-off in terms of the amount of post-selection required. For the most discriminating of exclusive decoders, we demonstrate a threshold of 50% under depolarizing noise for the surface code (or 32(1)% for the fault-tolerant case with phenomenological measurement errors), and up to a quadratic improvement in logical failure rates below threshold. Furthermore, surprisingly, with a modest exclusion criterion, we identify a regime at low error rates where the exclusion rate decays with code distance, providing a pathway for scalable and time-efficient quantum computing with post-selection. We apply our exclusive decoder to the 15-to-1 magic state distillation protocol, and report a 75% reduction in the number of physical qubits required, and a 60% reduction in the total spacetime volume required, including accounting for repetitions required for post-selection. We also consider other applications, as an error mitigation technique, and in concatenated schemes. Our work highlights the importance of post-selection as a powerful tool in quantum error correction.

3.1 Introduction

Quantum error correction (QEC) can facilitate large-scale quantum computations even with relatively noisy components. Strategies for practical quantum error correction should have good performance in terms of both the threshold and the rate that logical failure rates decay below threshold, as well as modest hardware overheads such as the number of physical qubits per logical qubit and the complexity of the circuits used to extract the error syndrome.

Correcting errors in a quantum code is a complex process with significant resource overheads, and some strategies for practical quantum computing make use of the much easier task of *detecting* errors, post-selecting the cases where no errors have occurred [104]. Post-selection has played a key role in a number of recent experimental demonstrations of quantum error correction [22–27, 29, 32, 35]. Unfortunately, many direct applications of quantum error detection and post-selection are not scalable, as the probability of post-selection in a large circuit becomes vanishingly small. Scalable approaches typically require small, fixed-distance codes to ensure a reasonable overall probability of success.

Here, we investigate how to use the rich information contained in the error syndrome itself to make better decisions on how to post-select results, essentially interpolating between the extreme cases of error correction and error detection. Such an approach has been considered for preparing magic states for state injection, since the circuits are so small that it is more efficient to restart the procedure when errors are detected, in order to attain higher quality final states or to reduce the overheads [105–109]. The numerical simulations required to study this regime are challenging, and existing tools for QEC simulations such as the splitting method [87, 95] are not directly applicable. As a result, little is known about the impact of such post-selection methods on the performance of codes of interest such as the surface code, including thresholds and logical failure probabilities, compared with the deterministic case.

We introduce an *exclusive decoder* that aborts for syndromes for which the uncertainty in decoding exceeds some tolerance. We develop new numerical methods to characterise the thresholds and sub-threshold behaviour of the surface code with the exclusive decoder, demonstrating that exclusive decoding can give small surface codes the logical performance of a much larger code with a fraction of the overhead.

A common (and well-warranted) criticism of many uses of post-selection in quantum error correction and fault-tolerance is that they are not scalable. We numerically investigate the abort probability of our exclusive decoder and find—remarkably—that the abort probability of the exclusive decoder also exhibits threshold behaviour, for all but the most intolerant instance of the decoder. That is, there is a threshold error probability, below which the abort probability decreases exponentially in the distance of the surface code. This behaviour suggest that an

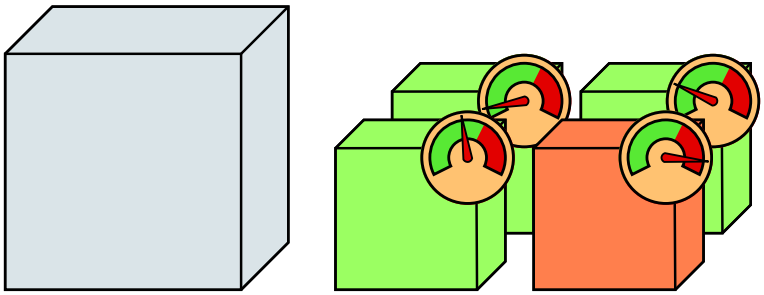


Fig. 3.1 (left) A monolithic attempt at a medium-sized error-corrected computation, that is guaranteed to return a result but requires large overheads to reach the desired target accuracy. (right) A collection of lightweight, error-corrected attempts at the same computation. Each attempt is run at some tolerance, and if the uncertainty associated with any correction exceeds the tolerance, then the attempt is aborted. This approach is not guaranteed to return a result but failure is heralded, and when a result is returned the accuracy can be very high with low physical overheads.

exclusive decoding strategy can find application in large-scale fault-tolerant quantum computing more broadly.

We investigate the use of exclusive decoding for medium-sized logical circuits, especially relevant on near-term devices with limited logical qubits, and for state preparation and distillation. The main idea is to break up a medium-sized error-corrected computation into many light-weight attempts at the computation, which may or may not succeed but for which failure can be heralded; see Fig. 3.1. The result is that the large-scale error-corrected computations can be made accessible on smaller quantum devices and with lower spacetime overheads. While the computation may need to be repeated for success, much like other error mitigation strategies, the favorable behavior of the abort probabilities for the exclusive decoder suggests that such an approach will be useful in near-term implementations of fault-tolerant logic. We also explore applications of this decoding strategy to concatenated codes, and in quantum communication.

3.2 Results

3.2.1 Exclusive decoding

Our exclusive decoder generalizes the concept of a decoder as used in quantum error correction. A standard decoder is designed to take syndrome measurement data and output a logical correction, and its performance is captured by the accuracy of this correction. However the full set of syndrome data is a large, rich data set that contains more information than just the most likely logical Pauli correction [110, 111], and specifically it includes information

about the likelihood of correct decoding [112]. An exclusive decoder takes this same syndrome measurement data, and outputs either (1) a recommended logical correction; or (2) an abort mode where no logical correction can be identified as being correct to a sufficient degree of certainty. The degree of certainty required is a design choice that will depend on the use-case, and is a controllable parameter in the exclusive decoder.

Concretely, the system we study here is a generalisation of the minimum-weight perfect matching (MWPM) decoder [42, 91], and we study its performance over the rotated surface code with boundaries, which allows a distance-d code to be constructed out of $n = d^2$ physical qubits. The standard MWPM decoder infers errors by matching defects into pairs, with a minimum-weight matching corresponding to a most-likely error. However, on a surface code with boundaries, it is possible to modify the MWPM decoder so that it returns a minimum-weight correction for each logical equivalence class (see Appendix 3.A for implementation details). With this modification, our exclusive decoder finds a global minimum-weight correction C, as well as alternate corrections from each other equivalence class $C\bar{L}$, where $\bar{L}$ is an X-type or Z-type logical operator. Let Δ be the unsigned weight difference between the global least-weight correction C and the next-least-weight alternate correction, C' that is logically inequivalent to C, i.e. such that $C'C$ is a logical operator. Then, the exclusive MWPM will abort if

$$1 - \frac{\Delta}{d} > c \tag{3.1}$$

where $0 < c \leq 1$ is a tuneable parameter that we call the *exclusive tolerance*. We also define the $c = 0$ case, which is a decoder that aborts if there is any non-trivial syndrome data. With this definition, we have that a small value of c means that C and C' must have a very large weight difference, whereas an exclusive decoder with $c = 1$ is the standard MWPM decoder that will attempt to decode all syndrome errors, independent of their success probability. While the exclusive MWPM decoder assumes perfect measurements as presented here, it can be generalised to a fault-tolerant version (see Sec. 3.2.4).

The construction we have presented is similar to post-selection rules that have been studied previously [109, 112], and is motivated by the fact that alternate corrections that are close together in weight have a similar likelihood of occurring, making it impossible to be confident which correction is the right one. In these situations, the exclusive tolerance is used to tune how much uncertainty the decoder will tolerate before it decides to abort. Since the abort condition for the exclusive decoder depends on Δ as a fraction of the code distance, we expect the choice of tolerance to have an effect on asymptotic quantities, such as thresholds and asymptotic overheads. In the following sections, we use sophisticated numerical tools (see Appendix 3.B) to study the exclusive decoder as we take large code distances.

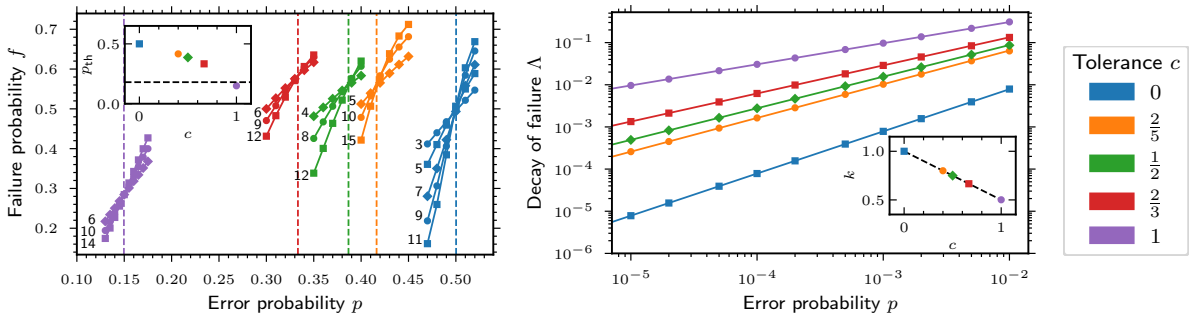


Fig. 3.2 Probability of logical failure for the exclusive MWPM decoder used with the rotated surface code, against depolarizing noise and in the code-capacity setting. Logical failures are computed only on post-selected error configurations. The $c = 0$ decoder is the zero-tolerance decoder that aborts on non-trivial syndrome data, and the $c = 1$ decoder is the standard MWPM decoder with no post-selection. (left) Markers are data points computed using the splitting method. Solid lines are failure probabilities extracted from a critical exponents ansatz, and dashed vertical lines are the fitted threshold values. For values of the tolerance $c = 0,\ 2/5,\ 1/2,\ 2/3$, and 1, we find threshold values $p_{\text{th}} = 50\%,\ 42\%,\ 38\%,\ 33\%$, and 15%, respectively. (left inset) Threshold values plotted against the exclusive tolerance. The horizontal dashed line is the threshold derived from the zero-rate Hashing bound in the non-post-selected setting. (right) Decay of logical failure probabilities below threshold. For each error probability below threshold, we fit the very general ansatz $\log(f) = d \log \Lambda(p) + \log C(p)$ using data from simulations over a range of code distances (see Appendix 3.D for details). The decay constants $\Lambda(p)$ are plotted with markers for each error probability. Solid lines correspond to the further ansatz Eq. (3.2), which corresponds to writing $\log \Lambda(p) = k \log p + \log A$. This ansatz is applicable far below threshold and only counts contributions to the failure probabilities from least-weight failing errors. The parameters k and A are determined by fitting, and in (right inset) we plot the fitted values k against the tolerance c. The dashed line is Eq. (3.3). We have that $k > 1/2$ for the exclusive decoder for all $c < 1$, whereas standard error correction techniques can achieve at best $k = 1/2$.

3.2.2 Exclusive decoder performance

Using simulations, we demonstrate that our exclusive decoder can lead to logical thresholds that are substantially higher than those of a standard decoder. Equally striking, we demonstrate that the scaling behaviour of the logical failure probability below threshold can also be significantly enhanced by using an exclusive decoder. Together, these results indicate that post-selection can provide significant additional power to error correction.

Thresholds. We estimate the threshold for the rotated surface code with the exclusive decoder under a depolarizing noise channel in the code-capacity setting (meaning perfect syndrome measurements). Logical failure probabilities are found by using the splitting method to extrapolate down from logical failure probabilities determined analytically at $p = 75\%$; see Appendix 3.B for details. Our results are summarised in Fig. 3.2. The threshold value is

$p_{\text{th}} = 50\%$ at zero tolerance, and decreases smoothly down to the standard MWPM decoder value of $p_{\text{th}} = 15\%$ as the tolerance increases to one. We remark that the depolarizing channel is known to have zero capacity for depolarizing strength in excess of $p = 25\%$, due to the no-cloning theorem [113–115]. The exclusive decoder is able to exceed this bound due to the use of post-selection, and we discuss the exclusive decoder further in the context of quantum channel capacities in Sec. 3.3.

One can gain some intuition about this improved decoder performance as follows. The performance of a decoder is determined by probabilistic factors, which describe the likelihood of a particular failing error occurring, and entropic factors, which arise by counting the number of errors that typically cause failure [42]. Errors that cause failure with the exclusive decoder are typically much less likely than errors that cause failure with a standard decoder, as will be discussed in Sec. 3.2.2. Additionally, entropic considerations are more favourable towards exclusive MWPM compared to standard MWPM. For both exclusive and standard MWPM, there are entropic factors arising from (i) the number of least-weight logical operators in the code, and (ii) the number of ways that w single-qubit Pauli operators can be placed onto the support of a given least-weight logical operator, where w is the weight of a least-weight failing error that is not aborted. Although the contribution from (i) is the same for exclusive decoding as it is for standard decoding, the contribution from (ii) is smaller for exclusive decoding than for standard decoding, and is not present at all in the case of a zero-tolerance decoder. This means that both probabilistic considerations and entropic considerations are more favourable towards the exclusive decoder. Large error probabilities are therefore required before entropic factors will favour a failing error configuration, and this leads to a large threshold.

Sub-threshold scaling of logical failure probabilities. Along with improved thresholds, our exclusive decoder provides up to a quadratic reduction in the observed logical failure probability for error probabilities below threshold. Since this suppression is more rapid than can be achieved with standard error correction techniques on the surface code, we say that errors are super-suppressed below threshold. Scaling improvements of this type can be used to reduce resource overheads, since fewer physical qubits are required to achieve commensurate logical failure probabilities compared to the standard approach.

The failure probabilities of quantum error correcting codes well below threshold are determined by the weight and number of least-weight failing errors. In particular, if the weight of the least-weight failing error is some fraction k of the code distance, then an ansatz logical failure probabilities below threshold can be written

$$f = C(Ap)^{kd} \tag{3.2}$$

where A^{kd} counts the number of least-weight failing errors. Since we expect to operate quantum error correction well below threshold, this ansatz is the key relation that determines the resource overheads for fault-tolerant quantum computation. In particular, the least fractional weight k is a key quantity of interest.

For standard surface code error correction, there is no decoder that can correct all errors of weight up to and including $w = \lceil d/2 \rceil$, which sets an upper bound $k \leq 1/2$. To see this, take a logical operator $\bar{L}$ of weight d. We can split $\bar{L}$ into two parts $\bar{L} = E + E'$ with weights $w = \lceil d/2 \rceil$ and $w' = \lfloor (d-1)/2 \rfloor$. Since E and E' have the same syndrome, it is impossible to simultaneously correct both of them. A signature of this problematic error configuration is that the two alternate corrections are very close in weight; in the example above the weight difference between the alternate corrections was equal to one. The exclusive decoder safeguards against this by aborting if there exists minimum weight corrections in alternate sectors that are close together in weight.

With the exclusive decoder, let w denote the weight of a failing error E that does not lead to an abort. Let w' denote the least-weight correction E' in an alternate sector, then the failure condition implies that $w - w' \geq d(1-c)$. Additionally, we must have $w + w' \geq d$, since $E + E'$ gives a logical operator. This gives a lower bound on the weight of failing errors. The exclusive decoder can abort or correct any error with weight $w < w_{\text{min}}$, which expressed as a fraction of the code distance is $w_{\text{min}} = kd$ with k given

$$k = 1 - \frac{c}{2}\,. \tag{3.3}$$

We show in Fig. 3.2 the subthreshold failure probabilities, and in particular we extract the least fractional weight k as a function of the exclusive tolerance, which shows the agreement with Eq. (3.3).

3.2.3 Abort probabilities

A key quantity of interest for an exclusive decoder is the probability with which it will abort. The behavior of the abort probability will capture the trade-off being made for reduced logical failure probabilities and boosted logical thresholds, in terms of (say) an increased time cost in a 'repeat until success' approach. Our analysis shows that the abort probabilities for non-zero tolerance exhibits threshold behaviour, meaning that the probability of aborting decreases exponentially as a function of the code distance for error probabilities less than the abort threshold. The existence of such an abort threshold suggests that the exclusive decoder may find application in scalable approaches to fault-tolerance as well as near-term uses.

Abort thresholds. In Fig. 3.3, we demonstrate numerical evidence of abort thresholds for non-zero tolerance $c > 0$. For the case of zero-tolerance, the exclusive decoder does not have an abort threshold, and we characterize the abort probabilities for the zero-tolerance decoder in App. 3.C.

For all tolerances $c > 0$, the value of the abort threshold is much smaller than the value of the logical threshold, and also for the value of the logical threshold for standard MWPM decoding. This is as expected; the exclusive decoder will definitely abort on any low-weight error that causes standard MWPM decoding to fail, but additionally will also abort on some errors that would have been successfully corrected by standard MWPM decoding, since the exclusive decoder can be quite cautious. This fact is significant, and has consequences for the usefulness of the exclusive decoder for large codes. There are two regimes in which we can imagine operating the exclusive decoder: (i) below the abort threshold; and (ii) above the abort threshold but still below the logical threshold. In the first regime, the exclusive decoder can be used on large codes since the time-cost does not blow up with the size of the code, and we discuss these asymptotic overheads in Sec. 3.3.1. In the second regime, the practical usefulness of the exclusive decoder is likely limited to small codes. We further discuss the scope of use-cases in Sec. 3.3.

Scaling of abort probabilities. As with logical failure rates, the abort thresholds only tell part of the story, and we also investigate the scaling of the abort probability as a function of code distance both above and below the abort threshold. Below the abort threshold, we study the decay of abort probabilities asymptotically. Above the abort threshold, we develop an empirical ansatz to calculate abort probabilities for finite-size codes.

In the regime of error probabilities below the abort threshold of the protocol, we can expect a similar ansatz to Eq. (3.2) to hold. That is, we expect to describe abort probabilities by

$$g = \tilde{C}p(\tilde{A}p)^{\tilde{k}d}\,. \tag{3.4}$$

Here, $\tilde{k}d$ is one less than the weight of the least-weight error that will lead to an abort, which gives rise to an additional factor p as compared to Eq. (3.2). The factor $\tilde{A}^{\tilde{k}d}$ counts the number of least-weight aborting errors. To find $\tilde{k}$, take an error E of weight $w \leq d/2$ that leads to an abort, and let w' denote the weight of the least-weight correction E' that is inequivalent to E. We have $w' - w < (1-c)d$ and $w + w' \geq d$ holding simultaneously, which gives $2w > cd$, or equivalently

$$\tilde{k} = \frac{c}{2}\,. \tag{3.5}$$

We remark that the more exclusive the decoder is, in terms of being intolerant of uncertainty, the more slowly the abort probability decays in the code distance. In Fig. 3.3, we show the scaling

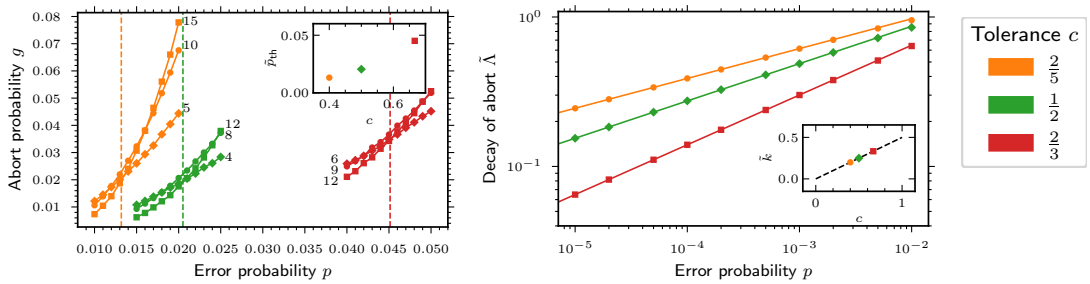


Fig. 3.3 Probability that the exclusive MWPM decoder over the rotated surface code will abort. The noise model is depolarizing noise in the code-capacity setting. (left) Markers are data points computed using direct Monte Carlo estimation, and crossings indicate the existence of a threshold with respect to abort probabilities. Solid lines are abort probabilities extracted from a critical exponents ansatz, and dashed vertical lines are the fitted threshold values. For tolerances $c = 2/5,\ 1/2,\ 2/3$, we observe abort threshold values $\tilde{p}_{\text{th}} = 1.3(1)\%,\ 2.1(1)\%,\ 4.5(1)\%$, respectively. Note that only data for tolerances in the range $0 < c < 1$ are shown. The zero-tolerance case is omitted because it does not possess a threshold for aborts, and the standard MWPM case is omitted because there are no aborts. (left inset) Threshold values for the abort probability plotted against the exclusive tolerance. (right) Decay of abort probabilities below threshold. For each error probability below threshold, we fit the very general ansatz $\log(g) = d \log \tilde{\Lambda}(p) + \log \tilde{C}(p)$. The decay constants $\tilde{\Lambda}(p)$ are plotted with markers for each error probability. Solid lines correspond to the further ansatz Eq. (3.4), applicable far below threshold, which corresponds to writing $\log \tilde{\Lambda}(p) = \tilde{k} \log p + \log \tilde{A}$. The parameters $\tilde{k}$ and $\tilde{A}$ are determined by fitting, and in (right inset) we plot the fitted values of $\tilde{k}$ against the tolerance c. The dashed line is Eq. (3.5). We remark that the abort probability decays more slowly as the tolerance to uncertainty is decreased.

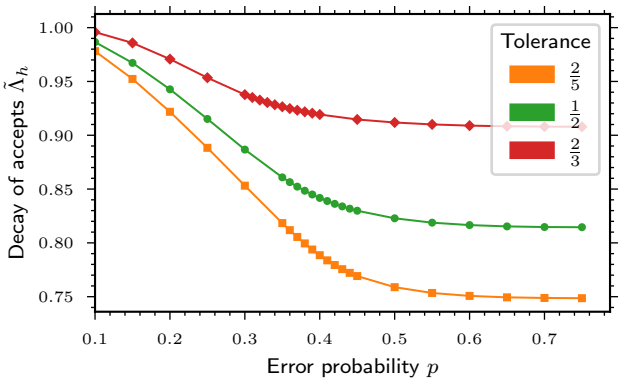


Fig. 3.4 The decay of acceptance probabilities as a function of the number of qubits in the code, above the abort threshold. Data is for the exclusive decoder with depolarizing noise on the rotated surface code, in the code-capacity setting. For each error probability above the abort threshold, we fit the ansatz Eq. (3.6), which can also be written $\log h = n \log \tilde{\Lambda}_h(p) + \log \tilde{C}_h(p)$, where n is the number of qubits and in this case we have $n = d^2$. Details of these fits can be found in Appendix 3.D. We extract and plot $\tilde{\Lambda}_h$ as a function of the error probability and for each value of the tolerance.

of the abort probabilities with the code distance, and in particular we see good agreement with Eq. (3.5).

We are also interested in the abort probabilities for finite-size codes for error probabilities that are larger than the abort threshold, but smaller than the logical threshold. It is less clear a priori how abort probabilities will behave above the abort threshold. However, we empirically observe that the acceptance probability, defined $h = 1 - g$, decays exponentially in the number of qubits n in the code. That is, we have

$$h = \tilde{C}_h(p)\tilde{\Lambda}_h(p)^n \tag{3.6}$$

where $\tilde{C}_h$ can be interpreted as a finite-size offset, and $\tilde{\Lambda}_h < 1$ is the decay factor, and n is the number of qubits with $n = d^2$ for the rotated surface code. In Fig. 3.4, we extract decay factor $\tilde{\Lambda}_h$ for a range of error probabilities and exclusive tolerances. Since above the abort threshold, the practical use of the exclusive decoder is limited to finite-size codes, Eq. (3.6) can be useful to apply to small code instances.

3.2.4 Fault tolerant syndrome measurements

We now turn to the question of fault-tolerance and imperfect measurements. The general idea of an exclusive decoder maps straightforwardly into the fault-tolerant case, and in particular the MWPM-based decoders used for the surface code are naturally made fault-tolerant [42].

There are a few subtleties that offer challenges for characterising the fault-tolerant case numerically. A common setting in which decoders are numerically benchmarked in the fault-tolerant setting involves taking periodic time-like and space-like boundary conditions. A minor difficulty arises when generalizing exclusive MWPM to this setting, since it involves fixing the parity of matches over an arbitrary subset of edges in the decoding graph. This should be compared to the code-capacity regime with rough and smooth boundaries, where we only needed to fix the parity of matches between bulk and boundary vertices (see Appendix 3.A). This problem is still solvable in the fault-tolerant setting, since the decoder studied in Ref. [73] utilizes a heuristic solution to this problem that can find low-weight corrections in alternate sectors, and this subroutine can be used to implement a fault-tolerant version of exclusive MWPM. Then, the abort condition can treat time-like errors in exactly the same way as space-like errors.

We do study the exclusive decoder at zero-tolerance and with imperfect measurements in Appendix 3.C. There, we set faulty measurements to occur with probability $p_m = 2p/3$, and we report a fault-tolerant threshold value $p = 32(1)\%$. As well as this, we develop a sophisticated ansatz that captures subthreshold failure probabilities, and observe the scaling relationship $f \propto p^d$, as expected. Additionally, we introduce in Appendix 3.E the *exclusive union-find decoder*, which also generalizes to the fault-tolerant regime straightforwardly. We expect Eq. (3.3) and Eq. (3.5) to still hold for the fault-tolerant exclusive MWPM decoder, and the fault-tolerant exclusive union-find decoder.

Finally, we remark that future work is still required to generalize homology-based decoders, such as maximum-likelihood decoding or exclusive decoding, to the fault-tolerant setting with open time-like boundaries in a sliding-windows fashion [116, 117].

3.3 Discussion

The post-selection scheme presented here is a promising approach to protecting against noise in the nascent era of fault-tolerance, where logical qubits may be limited in size. Post-selection can enable more reliable calculations out of early-stage logical qubits that may not be sufficient for a particular use-case using standard decoders. They can also allow for a reduction in resource overheads. We now explore these applications in some detail.

3.3.1 Resource overheads

The exclusive decoder filters out the noisiest shots in a quantum computation. In the regime of limited logical qubits, this decoder allows access to deeper logical circuits, and can reduce footprints for fault-tolerant primitives. We view this regime of circuits as a natural sequel to

NISQ (noisy intermediate-scale quantum) circuits, where logical encodings are combined with post-selection to access early-stage fault tolerance on noisy intermediate-scale devices.

Deeper logical circuits. We analyze this trade-off in the low-error-probability regime. A circuit volume q can be executed using logical qubits at failure probability f, up to a target accuracy $\epsilon = qf$. In the non-exclusive case, the scaling of f in the code distance is at best $f = p^{d/2}$. In the exclusive case, the scaling is improved to $f = p^{kd}$ for some $1/2 \leq k \leq 1$. However, this comes at the cost of repetitions that must be performed

$$R = (1 - p^{\tilde{k}d})^{-q} \tag{3.7}$$

where $\tilde{k} = 1 - k$, since the computation must be restarted if any error correction cycle returns an abort mode. We remark that R appearing in Eq. (3.7) is the number of repetitions that are expected to be required. The actual number of repetitions required is described by an exponential decaying random variable, so observing a number of repetitions much larger than R becomes very unlikely.

If we take a fixed number of physical qubits, then using surface codes of a certain distance will set an accessible circuit volume $q_0 = \epsilon p^{-d/2}$. This can be increased to $q = \epsilon p^{-kd}$ using exclusive decoding techniques. If we only attempt to achieve a constant factor improvement, then we can set a fixed $a = \sqrt{\epsilon} q/q_0$. We also define the rescaled circuit volume $x = q_0/\sqrt{\epsilon}$. We can rewrite the number of repetitions required by Eq. (3.7) in terms of x and a, giving $R = (1 - a/x)^{-ax} \approx \exp(a^2)$. This approximation is accurate provided that q is small compared to q_0^2/ϵ. Finally, we can re-arrange to express the factor increase in circuit depth achievable in terms of the number of repetitions we are willing to perform

$$q = \sqrt{\frac{\log R}{\epsilon}} q_0 \,. \tag{3.8}$$

For an entire algorithm where we might consider $\epsilon \approx 1$, then this trade-off allows, for instance, a 2X increase in the accessible circuit depth, in exchange for repeating the circuit approximately 55 times. We remark that this trade-off is only available up to an upper bound $q \leq q_0^2/\epsilon$ that arises from the fact that the maximum boost to logical failure probabilities is quadratic, which is achieved by the zero-tolerance decoder.

Reduced qubit overheads. An alternative but equivalent view is that we may want to execute a fixed size circuit, but using a smaller number of physical qubits, i.e., using a smaller distance code. We start with a fixed circuit of size q, executed using surface codes of distance d_0, and with m_0 physical qubits. Using a exclusive decoder with tolerance $c = 1 - 2k$, this

circuit can alternatively be implemented using distance $d = d_0/(2k)$ codes, and requiring $m = m_0/(4k)^2$ physical qubits. The target accuracy ϵ remains unchanged.

If we assume that the number of repetitions required is not unreasonably large, so that $\log R$ is small compared to q, then we can rewrite Eq. (3.7) in terms of circuit quantities as

$$\log R = \epsilon^{2\sqrt{m/m_0}-1} q^{2(1-\sqrt{m/m_0})} . \tag{3.9}$$

This equation suggests that the exclusive decoder will allow large footprint reductions, at a modest time-cost, when applied to circuits with low target accuracy ϵ, but that also have a small number of gates q.

Key application: distilling magic states. Motivated by conclusions drawn from Eq. (3.9), we consider the exclusive decoder applied to medium-sized state preparation circuits. The exclusive decoder is well-suited to this use-case, since the output of a state preparation circuit is a quantum state that needs to be extremely high fidelity, since it will get consumed in a much larger circuit. This small target accuracy allows an exclusive decoder to be used for a footprint reduction, without also incurring a large overhead in terms of repetitions required.

As an example, we do a rough analysis of the 15-to-1 magic state distillation circuit [118], in the fault-tolerant setting. We simulate the zero-tolerance exclusive decoder using a distance $d = 4$ code in the fault-tolerant regime, and compare to a standard decoder using a distance $d = 8$ code. We assume the decoders have similar logical performance, since the weight of the least-weight failing error is equal to four in both cases. We have verified though direct comparison that this assumption is quite generous towards the standard decoder since there are many more failure mechanisms on the distance-8 code compared to the distance-4 code. We use values for q and ϵ based on the 15-to-1 construction from Ref. [119], that has $q \approx 90$ error correction cycles, an output accuracy limited by $\epsilon_T \approx 10p^3 \approx 10^{-11}$, due to errors that occur during faulty T measurement. We set an error probability $p = 10^{-4}$. Using the detailed ansatz in Appendix 3.C, we have an expression for f and g, and can compute the total failure probability of the circuit with the zero-tolerance decoder, $\epsilon = fq \approx 5.8 \times 10^{-14}$. This means approximately one in two hundred failures will be due to surface code failures, as opposed to faulty T measurement failures, so surface code failures contribute negligibly to the total circuit accuracy with these parameters. We can also compute the amount to circuit repetitions required using Eq. (3.17) with $R = (1-g)^{-q}$, and report that only 3.2 circuit repetitions are required on average due to the use of the zero-tolerance decoder. We compare to the case of a standard decoder using a distance $d = 8$ code with no repetitions, and assume this comparison gives equal logical performance. Making this comparison, we report that the exclusive decoder requires 25% of the physical qubits used by the standard decoder, and also requires 40% of the total spacetime volume, including with repetitions, in order to achieve commensurate

logical performance with the standard decoder. We remark that the logical performance of the zero-tolerance decoder was better than it needed to be, since the logical accuracy of the distillation protocol is bottlenecked by faulty T measurement, not by surface code failures. We could have chosen a non-zero tolerance in order to reduce the number of repetitions, at the expense of larger logical failure probabilities, provided we maintain $\epsilon \ll \epsilon_T$. In general the optimal tolerance will balance savings that result from using a small code distance, with the repetitions that are required when the decoder aborts.

We emphasize the usefulness of exclusive decoding applied to a range of medium-sized state preparation tasks, beyond magic state distillation. Especially, the exclusive decoder is well suited to quantum computing paradigms that involve breaking large computational tasks into smaller and independent state preparation tasks, such as with error-correcting teleportation [104] or quantum dynamical programming [120]. These paradigms can be cheap in terms of the on-line part of the computation [121, 122], and the off-line part of the computation can be done using low-distance codes with exclusive decoding.

3.3.2 Other applications

The exclusive decoder could also prove useful in schemes that use code-concatenation to achieve higher encoding rates, such as Refs. [112, 123, 124]. An approach explored recently in Ref. [124] involves soft information from an exclusive-style decoder being passed to the decoding algorithm of the outer code. Our results can also be understood in this setting, since in the practical limit where tolerance is large, abort modes can be effectively regarded as erasures on the outer code. The exclusive decoder can then be viewed as a gadget that converts Pauli noise into erasure noise. This is similar in spirit to previous work on erasure conversion via qubit engineering [125–127], or using concatenation with small inner codes [128]. An exclusive-style decoder that extracts soft information about logical Pauli noise for code-concatenation was also studied in Ref. [112]. The success of this approach is based on the fact that structured or biased noise can be exploited to achieve very large performance gains [100]. The quantum erasure channel, in particular, is arguably the easiest-to-correct source of noise that can affect a quantum computation—it has non-zero capacity up to $p < 50\%$ [129], it requires a weight-d erasure error to fail a distance-d code, and it admits high performing and accurate decoders [130]. The erasure channel is also conceptually straightforward relative to Pauli noise. Insights gained from first considering erasure are often found also to be useful in the more general noise setting, both in the context of designing decoders [44, 130–132], and designing error correcting codes [133–135]. The conversion of Pauli noise to erasure noise at the surface code level by the exclusive decoder may help to simplify the decoding problem faced by the outer code in a concatenated scheme. If the outer code is a more exotic kind of code, for which decoding is more challenging, then this will be an especially useful approach.

Another use-case of the abort mode is to signal the level of sophistication of the decoder that should be used to correct the error. A common design paradigm for high-performance real-time decoders involves hierarchical decoding [76–83, 136], and we point the reader to Ref. [137] for a recent perspective. In this paradigm, the first decoding stage is fast and efficient, but may make mistakes more readily due to a lack of global syndrome information, for example. This first stage is designed to correct the majority of sparse error configurations. The second stage is powerful but costly, correcting the difficult corrections that are leftover by the first stage. The exclusive techniques explored here are capable of filtering the easy errors from the difficult errors, allowing sophisticated decoding to be called only when needed. As an example, the pre-decoder studied in Ref. [136] is fast and efficient, but can in rare circumstances grow the weight of an adversarial error by a factor 3/2. This leads to failure at the second stage of decoding, and reduces the effective code distance. However, when combined with suitable exclusive decoding, the protocol can correct up to the full code distance, whilst also correcting typical sparse errors with high performance, maintaining the best of both worlds.

We also consider how our exclusive decoder fits into the existing framework that describes quantum communication through noisy channels. In that framework, a channel with capacity Q can be used to transmit m qubits with n uses of the channel, provided that $m/n \leq Q$, and with error vanishing in n (see Ref. [138, 139] for two surveys). However, the capacity depends on the presence or absence of free classical communication. A forward classical side-channel from sender to receiver will not increase the capacity [140]. A backwards classical side-channel will increase the capacity to $Q_B \geq Q$ [141], and a two-way classical side channel will increase the capacity further to $Q_2 \geq Q_B \geq Q$ [129, 140]. A quantum error-correcting memory with post-selection can be viewed as a communication protocol with two-way classical communication via error-correcting teleportation [104, 142]. In this setting, transmission of a qubit is achieved by first establishing a high-fidelity logical Bell pair shared between sender and receiver, and then a logical qubit can be teleported through the Bell pair, requiring some forward classical communication to pick a suitable Pauli correction [143]. Many attempts can be made at establishing this Bell pair, since the receiver can ask the sender to keep trying until an exclusive decoder does not return an abort on the receivers half of the Bell pair. In this picture, QEC techniques are directly used to protect against noisy channels. This approach complements the approach of Ref. [144, 145], where QEC techniques are used to fault-tolerantly implement encoding and decoding circuits for communication with noisy local gates. Since we have used the surface code in the current work, which has zero rate, we cannot hope to learn anything about the region $Q_B > 0$. However, it does raise an interesting question: is there a separation between $Q_B > 0$, and the region where communication is not possible. Put another way, is it possible to pass a constant number of qubits through a channel that has zero capacity? Unfortunately, the zero-tolerance surface code protocol is not an example, since the 50% depolarizing channel with a backward side-channel is known to have positive capacity [140]. However, the numerical

tools we have developed allow us to study the asymptotics of error-correcting codes with huge amounts of post-selection, and future work could explore better-performing codes to transmit information through very noisy channels.

Acknowledgements

We are grateful to V. Siddhu, G. Smith and M. Tomamichel for helpful conversations and comments. BJB thanks the Center for Quantum Devices at the University of Copenhagen for their hospitality. This work is supported by the Australian Research Council via the Centre of Excellence in Engineered Quantum Systems (EQUS) project number CE170100009, and by the ARO under Grant Number: W911NF-21-1-0007.

3.A Implementing exclusive MWPM decoding

Here we describe how to construct a matching graph so that the matching returned by a MWPM algorithm will correspond to a least-weight logical operator within a specified logical sector. We follow the construction of Ref. [146], and focus on matching graphs for one sector of the surface code, with boundaries.

The matching graph used for standard MWPM is shown in Fig. 3.5. It has bulk vertices that correspond to defects in the code. Each pair of bulk defects is connected with an edge, with a weight that reflects the probability of an error chain leading to a creation of the defect pair. The standard matching graph also has boundary vertices that allow defects to be matched to the boundary of the code. The connectivity of the boundary vertices allows every bulk node the freedom to be matched to its nearest boundary, whilst placing no further constraint on the matching.

We want to modify this graph in order to fix the parity of matches that connect defects to a particular boundary. Very generally, if we choose some subset of the vertices of a graph V', then the parity of matches included in a perfect matching that pair vertices in V' with vertices not in V' is fixed by the size of V'. We therefore need to modify the matching graph to fix the number of virtual boundary nodes at each boundary. A matching graph that does this is shown in Fig. 3.5.

Note the small detail that, for every bulk defect, it is now necessary to include an edge between the defect and both boundaries. This inclusion is important because, in the exclusive setting, it can sometimes be optimal to match a defect to the more distant boundary. This can be necessary in order to satisfy the parity constraints on that boundary, so that other bulk defects are free to be matched with each other in a low-weight correction. This is contrasted to

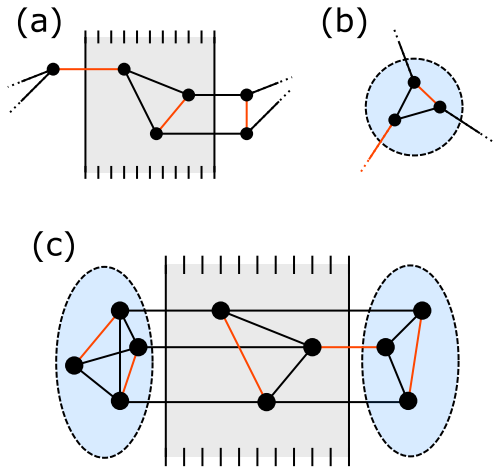


Fig. 3.5 (a) The matching graph for standard MWPM for defects created by X-type Pauli errors that condense on the smooth boundaries. There are three bulk defects, and therefore three boundary defects, with each bulk defect connected to its nearest boundary. The boundary defects are fully connected with each other, with some connections wrapping around the figure, as indicated by edges that terminate in dots. A minimum weight perfect matching is indicated in red. (b) A subset of a graph containing an odd parity of vertices. The parity of matches between the subset and its complement is fixed to be odd. (c) The exclusive matching graph has one set of boundary nodes for each boundary. On the right boundary, there is one boundary vertex for every defect in the code, which fixes an odd parity of matches to the right boundary. On the left boundary, there is one boundary vertex for every bulk defect, plus an additionally boundary vertex to fix an even parity of matches to the left boundary.

the standard MWPM decoder, where it is never optimal to match a defect to the more distant boundary.

We remark that the time-cost of this exclusive matching decoder on the surface code is a constant factor overhead compared with a standard MWPM decoder. This is because the MWPM subroutine must be called two times for both the X- and Z-type logical sectors in the code. In the case where there are multiple logical qubits, the time-cost will scale linearly in the number of logical qubits.

3.B Numerical methods

In this work, we develop new numerical methods that are used to compute the post-selected failure probabilities and acceptance probabilities presented in the main text. These methods make use of the splitting method [87, 95] as a subroutine, which we review briefly in subsection 3.B.1. In subsection 3.B.2 we show how the splitting method is used to develop the numerical methods that are used in this work.

Algorithm 2: The exclusive decoder for non-zero tolerance.

Input : List of defect locations $b = (b_i)_{i=1...n}$.
Input : Exclusive tolerance $c > 0$
Output : A correction C_X, or abort.

Function `Decode(`b, c`):`
  `// base graph construction`
  $l_i \longleftarrow$ location on left boundary that is closest to b_i.
  $r_i \longleftarrow$ location on right boundary that is closest to b_i.
  $g \longleftarrow$ graph with vertices $V = l \sqcup b \sqcup r$.
  for i, j *in* $1..n$ *with* $i \neq j$ **do**
    Add edge (l_i, l_j) with weight 0 to g.
    Add edge (b_i, b_j) with weight $d(b_i, b_j)$ to g. `// d is distance on lattice`
    Add edge (r_i, r_j) with weight 0 to g.
  end
  for i *in* $1...n$ **do**
    Add edge (l_i, b_i) with weight $d(l_i, b_i)$ to g
    Add edge (r_i, b_i) with weight $d(r_i, b_i)$ to g
  end

  `// modification to force parity of boundary matchings`
  $\pi_n \longleftarrow n \bmod 2$
  for π_l *in* $\{0, 1\}$ **do** `// parity of left boundary matchings`
    $g_{\pi_l} \longleftarrow$ a copy of g
    if $\pi_l \neq \pi_n$ **then**
      Add vertex l_{n+1} to g_{π_l} `// dummy vertex, no location`
      for $i = 1...n$ **do**
        Add edge (l_{n+1}, l_i) to g_{π_l} with weight 0
      end
    end
    $\pi_r \longleftarrow \pi_l + \pi_n \pmod 2$ `// ensure even number of vertices`
    if $\pi_r \neq \pi_n$ **then**
      `// same as above`
    end
    $C_X^{\pi_l} \longleftarrow \mathsf{MWPM}(g_{\pi_l})$ `// two alternate corrections`
  end

  `// pick a correction or abort`
  $(C_X, C'_X) \longleftarrow$ corrections $C_X^{\pi_l}$ ordered by increasing weight.
  if $(w(C'_X) - w(C_X)) < d(1 - c)$ **then** `// w is Pauli weight`
    return abort
  else
    return C_X
  end
end

3.B.1 The splitting method

The splitting method allows for the simulation of extremely rare events by computing and multiplying together the ratios of probabilities of a sequence of increasingly rare events. We give only a high-level overview of aspects of the splitting method that are relevant to this work, and we point the interested reader to Ref. [95] or Ref. [87] for a more detailed explanation of this method applied to quantum error correction.

The splitting method is very useful, for instance, for computing failure probabilities of quantum error-correcting codes at small error probabilities. In particular, denote by $\mathbb{P}_j$ a family of error models, where we may imagine large j corresponding to a low-error probability regime, and denote by $\mathcal{F}$ the set of error configurations that will cause a given decoder to fail. The splitting method makes use of the acceptance ratio method from [96], for estimating the free-energy difference between two canonical ensembles. The acceptance ratio method provides a means to compute the ratio

$$R_j = \frac{\mathbb{P}_{j+1}(\mathcal{F})}{\mathbb{P}_j(\mathcal{F})}. \tag{3.10}$$

provided samples of failing errors, i.e., samples drawn from $\mathbb{P}_j(E|\mathcal{F})$ and from $\mathbb{P}_{j+1}(E|\mathcal{F})$. Then, provided that one has access to the direct probability $\mathbb{P}_0(\mathcal{F})$, for example by direct Monte-Carlo estimation, then the probability of failure at any error probability can be computed as a product of these ratios:

$$\mathbb{P}_j(\mathcal{F}) = \prod_{j'=0}^{j-1} R_{j'} \mathbb{P}_0(\mathcal{F}) \tag{3.11}$$

The quantity $\mathbb{P}_0(\mathcal{F})$ may be computed using standard Monte-Carlo techniques at some error probability where it is feasible to do so. This will likely be close to threshold, where the failure probability does not vanish in the code distance, and in practice a rough estimate of the threshold can be found by a preliminary sweep. Alternatively, $\mathbb{P}_0(\mathcal{F})$ may be calculated using analytic methods where possible.

We remark that the splitting method was used to compute abort probabilities of the exclusive decoder below the abort threshold. The method was used as described, except with the failing error set $\mathcal{F}$ replaced with the abort set $\mathcal{A}$. Direct Monte-Carlo methods were used to compute abort probabilities close to the abort threshold. More sophisticated methods were required to compute post-selected failure probabilities, as well as for acceptance probabilities above the abort threshold, and these methods are developed in the next section.

3.B.2 Splitting individual logicals

In this section, we develop a new method that is used to compute the post-selected failure probabilities and acceptance probabilities presented in the main text.

The splitting method cannot be directly applied in the current setting, for two reasons. The first reason is that post-selected failure probabilities are conditional on the decoder accepting the syndrome history. That is, if we denote by $\mathcal{A}$ the set of errors that lead to aborts, then post-selected failure probabilities are given $f = \mathbb{P}(\mathcal{F}|\mathcal{A}^c)$. To use the splitting method, we would need to compute the ratios of conditional probabilities, and this would require being able to efficiently compute the conditional probability density function $\mathbb{P}_j(E|\mathcal{A}^c)$, which we cannot do. The second problem is that it is difficult to even sample from $P_j(E|\mathcal{A}^c)$ for low error probabilities. If only local steps are used by a sampler, the sampler is no longer ergodic because it cannot couple failing errors from different logical sectors without passing through some error that causes the decoder to abort. On the other hand, if non-local steps are used by a sampler, then a step that couples logical sectors is going to occur with a probability that is exponentially small in the code distance.

To overcome these difficulties, we introduce a method that samples from each logical sector separately, and at the end combines this sample data using a normalization procedure to give the probabilities of failure due to each logical sector. Denote by $\mathcal{L}$ the set of all error configurations E that are accepted by the decoder and are corrected by a Pauli correction C resulting in a logical operator $EC = \bar{L}$. We then have, for instance, that $\mathcal{F} = \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}$ for a code that contains only one logical qubit. Applying the splitting method to each logical sector separately allows us to calculate the unconditional sector probabilities:

$$\tilde{\mathbb{P}}_j(\mathcal{L}) = c_{\mathcal{L}} \mathbb{P}_j(\mathcal{L}) \,, \tag{3.12}$$

where $c_{\mathcal{L}}$ are as-yet unknown constants of proportionality that do not depend on j.

We now make use of our knowledge of the noise model, which in the main text was a depolarizing noise model, with $\mathbb{P}_0$ corresponding to a depolarizing parameter $p = 75\%$. At this error probability, the noise model has a symmetry, which is that all Pauli operators are equally likely to occur. This fact can be exploited to relate the constants of proportionality. In particular, it guarantees that $\mathbb{P}_0(\mathcal{L})$ is independent of $\mathcal{L}$. If we fix $\tilde{P}_0(\mathcal{L}) = 1$ without loss of generality, then inspecting Eq. (3.12) leads us to conclude that $c_{\mathcal{L}}$ must also be independent of $\mathcal{L}$, and we drop the subscript and write $c_{\mathcal{L}} = c$.

Finally, since post-selected failure probabilities involve ratios of unconditional sector probabilities, the constant of proportionality cancels out, and we have:

$$f_j = \frac{\sum_{\mathcal{L} \neq \mathcal{I}} \tilde{\mathbb{P}}_j(\mathcal{L})}{\sum_{\mathcal{L}} \tilde{\mathbb{P}}_j(\mathcal{L})} \,. \tag{3.13}$$

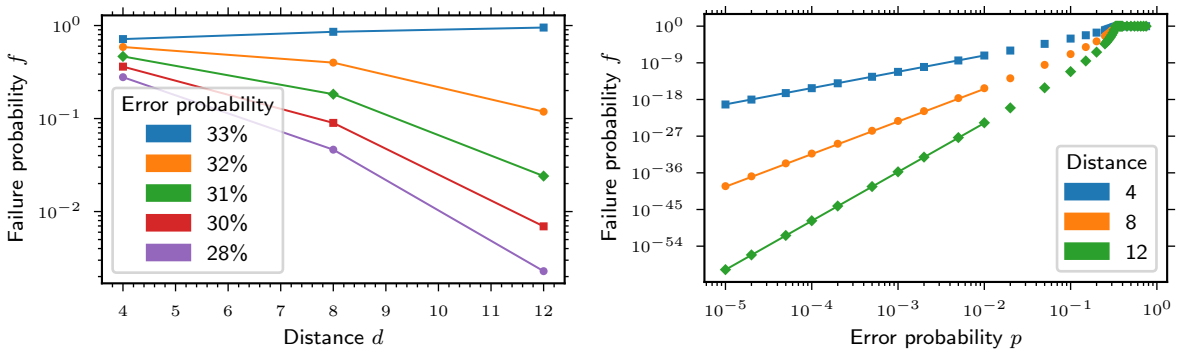


Fig. 3.6 Fault-tolerant logical failure probabilities for the zero-tolerance exclusive decoder. The decoder is used with the rotated surface code with $n = d^2$, and with $t = d$ rounds of syndrome measurement. The noise model is depolarizing noise of strength p, with measurement errors occuring with probability $p_m = 2p/3$. (top) We plot the logical failure probability against the code distance for a range of error probabilities near threshold, observing a threshold value of $p \approx 32\%$ (bottom) We show the scaling of logical failure probabilities below threshold. Markers correspond to data points, and the solid lines are the low-error probability ansatz Eq. (3.15), along with Eq. (3.16).

Acceptance probabilities above the abort threshold can be computed up to a constant of proportionality:

$$h_j := 1 - g_j = \frac{1}{c} \sum_{\mathcal{L}} \tilde{\mathbb{P}}_j(\mathcal{L}) \,, \tag{3.14}$$

and the constant of proportionality can be removed by fixing to direct Monte-Carlo estimates of the abort probability taken near threshold.

We made special use of properties of the depolarizing noise channel at $p = 75\%$. We remark that this is not a significant limitation on the flexibility of the method. In general, any splitting-style numerical method for computing probabilities of rare events will require some regime where the probabilities of events can be calculated directly, either by analytic methods or by Monte-Carlo estimation. All that is required is that a path of noise models is constructed that connects a target noise model to some 'nice' noise model, such as the completely depolarizing channel, and that samples are collected from every noise model along the path.

3.C Additional numerical results

In this section, we expand on results relating to the zero-tolerance decoder. The additional results that we show are:

1. We demonstrate fault-tolerant thresholds for logical failure probabilities using the zero-tolerance decoder, and show the subthreshold scaling behaviour of the failure probability.

2. We provide data on abort probabilities in the fault-tolerant regime.

3. We provide data on abort probabilities in the code-capacity regime, which was omitted from the main text on account of there being no thresholds for aborts.

3.C.1 Zero-tolerance fault-tolerant performance

In this section we look at the fault-tolerant version of the exclusive decoder. We choose periodic time-like and rotated periodic space-like boundary conditions, which are often convenient for numerical work. We repeat $t = d$ rounds of syndrome measurement. The noise model is depolarizing noise of strength p, with measurement errors occuring with probability $p_m = 2p/3$.

In Fig. 3.6, we demonstrate a fault-tolerant threshold value of 32%. This is smaller than the code-capacity threshold, which is to be expected as a general principle since there are more failure mechanisms in the fault-tolerant regime than there are in the code-capacity regime.

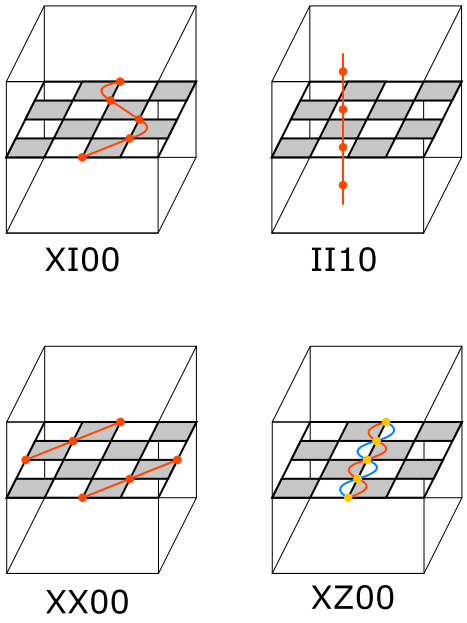
In the regime of small error probabilities, the logical failure probability is determined by the number of least-weight failing errors, and we propose the ansatz

$$f = A(d)p^d \tag{3.15}$$

where $A(d)$ is related to the number of least-weight failing errors, and will be discussed momentarily. In Fig. 3.8, we plot failure probabilities below threshold, and demonstrate good agreement with Eq. (3.15). We observe that failures scale with p^d instead of $p^{d/2}$, as would be the case with a standard decoder. In the fault-tolerant regime, just as in the code-capacity regime, the zero-tolerance decoder attains a quadratic improvement to logical failure probabilities below threshold, as compared to standard decoders.

We now look more closely at the combinatoric pre-factor $A(d)$. We remark that the ansatz Eq. (3.15) is slightly more general than, for instance, Eq. (3.2), since the form of $A(d)$ is as-yet unspecified. This more general ansatz is necessary, since an ansatz of form $A(d) = CA^d$ was not observed to satisfactorily fit the data for small code distances, and we require an accurate ansatz for fault-tolerant overhead estimation in Sec. 3.3.1. We now construct a sophisticated ansatz for $A(d)$ by explicitly counting least-weight failing errors.

In the zero-tolerance regime, a logical failure must result from either a logical operator in the code, or a time-like string of measurement errors that wraps around the time-like direction. It is possible to count the number of least-weight failing errors in this setting exactly, and derive an ansatz for the post-selected failure rates that becomes exact in the low-error rate



Sector	Multiplicity	Probability
$II00$ (x1)	1	$(1-p)^{d^2t}(1-p_m)^{d^2t}$
$XI00$ (x4)	$\frac{dt}{2}\binom{d}{d/2}$	$(\frac{p}{3})^d$
$II10$ (x2)	$\frac{d^2}{2}$	$(\frac{2p}{3})^d$
$XX00$ (x2)	dt	$(\frac{d}{3})^d$
$XZ00$ (x2)	dt	$(\frac{p}{3})^d$

Fig. 3.7 Analytic data used for the low-error probability ansatz Eq. (3.16), which sets the failure probability of the zero-tolerance decoder in the fault-tolerant regime as per Eq. (3.15). Sectors are labelled first by a logical Pauli on each qubit, and then by the parity of Z-type (then X-type) measurements in any time-slice. (top) Example error configurations from each sector that contains a minimum-weight error. Red balls denote Pauli X-type errors or measurement faults on Z-type stabilizers. Yellow balls denote Pauli Y-type errors. (bottom) Tabulated data for each sector that contains a minimum-weight failing error. The times symbol next to each sector label counts the number of other sectors that are equivalent to the given sector by a combination of transversal hadamard and/or rotation.

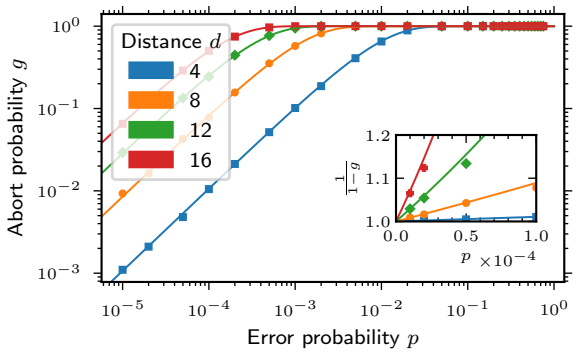


Fig. 3.8 Fault-tolerant abort probabilities for the zero-tolerance decoder. The decoder is used with the rotated surface code with $n = d^2$ qubits, and with $t = d$ rounds of syndrome measurement. The noise model is depolarizing noise of strength p, with measurement errors occurring with probability $p_m = 2p/3$. Data points are markers, and the solid line is the ansatz Eq. (3.17) . (inset) A zoomed in view of the inverse of the acceptance probability $(1 - g_0)^{-1}$. This is the factor increase in the time cost of the protocol as a result of exclusive decoding for one decoding cycle, since the computation may need to be repeated until the decoder accepts.

regime. For example, we consider the logical sector that involves no errors on the logical qubits, but with a non-trivial string of faulty Z-type measurement outcomes, as shown in the top right of Fig. 3.7. These error chains must be straight in order to be least-weight. Therefore, there are only $d/2$ least-weight errors in this sector, one for each Z-type stabilizer in the code. Each of these least-weight errors has probability $p_m^d = (2/3)^d p^d$ of occurring, since they are made up of measurement errors. Therefore, failure mechanisms from this sector contribute a term $(d/2)(2/3)^d$ to the function $A(d)$. All the least-weight failing errors are tabulated in Fig. 3.7, and a careful consideration of all the failure mechanisms in the code leads to the following ansatz

$$A(d) = \frac{4d^2}{2}\binom{d}{d/2}\left(\frac{1}{3}\right)^d + d^2\left(\frac{2}{3}\right)^d + 4d^2\left(\frac{1}{3}\right)^d \tag{3.16}$$

We plot subthreshold failure probabilities in Fig. 3.6, and see good agreement with the ansatz for error probabilities $p \leq 10^{-3}$.

3.C.2 Zero-tolerance fault-tolerant abort probabilities

The fault-tolerant exclusive decoder at zero-tolerance does not possess a threshold for aborts, since increasing the code distance will always increase the probability that a single error will result in an aborted outcome. Despite the lack of threshold, the abort probability is still an important quantity that sets the cost of the scheme.

A first-order expression for the abort probability at zero-tolerance is given by the probability that any error occurs in the code. To first order, we have that the abort probability is given:

$$g = 1 - (1-p)^{td^2}(1-p_m)^{td^2} \approx td^2(p+p_m) \tag{3.17}$$

where the approximation in the second equality holds when $pd^2t \ll 1$. The abort probabilities for the zero-tolerance case are sufficiently high that they can be studied effectively using pure Monte Carlo sampling, and we compare Eq. (3.18) to numerical data in Fig. 3.8. We observe that in this heavily post-selected regime, the abort probability increases quadratically with the code distance at all error probabilities.

3.C.3 Zero-tolerance code-capacity abort probabilities

The zero-tolerance exclusive decoder in the code capacity setting does not possess a threshold for aborts. We remark that abort probabilities in the code-capacity setting for the zero-tolerance decoder are qualitatively identical to the abort probabilities in the fault-tolerant regime, which were discussed in the previous section.

A first order expression for the abort probability is given by considering the probability of any error occurring in the code:

$$g = 1 - (1-p)^{d^2} \approx pd^2 \tag{3.18}$$

where the approximation in the second equality holds when $pd^2 \ll 1$. We study the abort probabilities using Monte Carlo sampling, and compare Eq. (3.18) in Fig. 3.9. As for the fault-tolerant case, there is no threshold behaviour and the abort probability increases quadratically with the code distance. The key difference between the fault-tolerant regime and the code-capacity regime is that abort probabilities are an order of magnitude smaller in the code-capacity regime, since there are fewer locations for errors to occur.

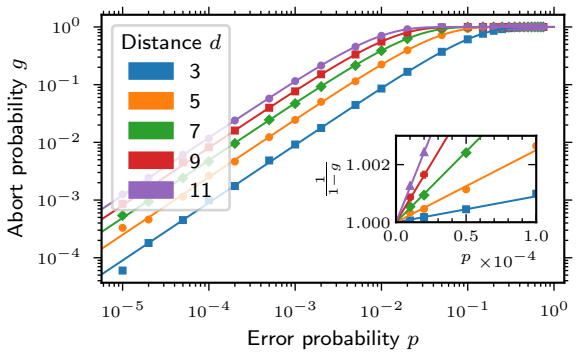


Fig. 3.9 Code-capacity abort probabilities for the zero-tolerance decoder. The decoder is used with the rotated surface code with $n = d^2$ qubits, and the noise model is depolarizing noise of strength p. Data points are markers, and the solid line is the ansatz Eq. (3.18). (inset) A zoomed in view of the inverse of the acceptance probability $(1 - g_0)^{-1}$. This sets the time-overhead of the exclusive decoder, since a computation may need to be repeated until the decoder accepts.

3.D Supporting figures

This section does not contain any additional results not explored in the main text. This section provides a more in-depth detailing of some of the data analysis and fitting procedures that support the main text figures. This section contains the following figures.

- Critical exponent plots of logical threshold values
- Goodness of fit of exponential decay of failure probabilities below threshold
- Plot of offset $C(p)$ for failure probabilities below threshold.
- Critical exponents plots of abort threshold values
- Goodness of fit of exponential decay of aborts below threshold
- Plot of offset $\tilde{C}(p)$ for abort probability below threshold
- Goodness of fit of exponential decay of acceptance probability above the abort threshold.
- Plot of offset $\tilde{C}_h(p)$ for acceptance probability above the abort threshold.

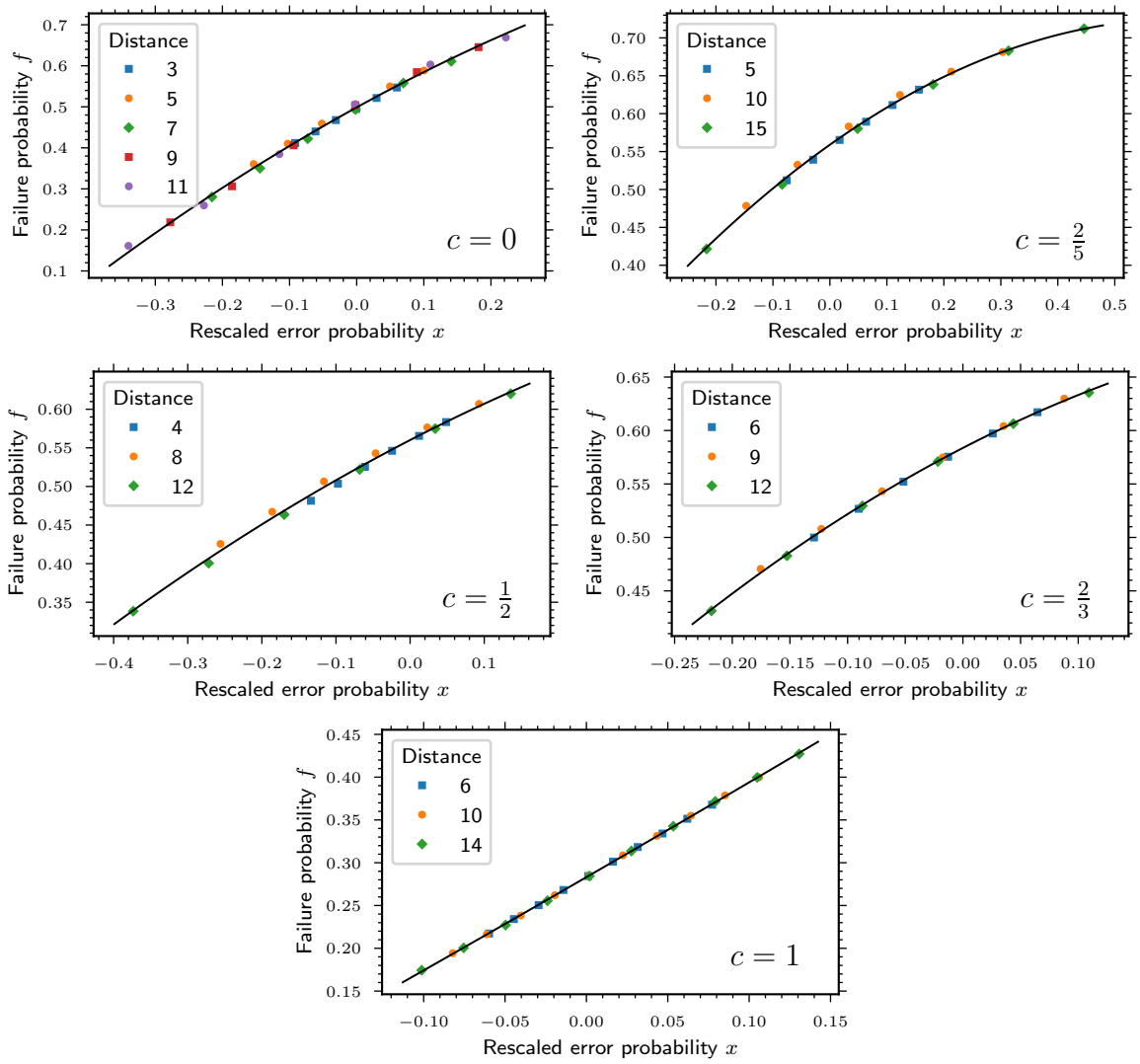


Fig. 3.10 Critical exponent plots for threshold values of failure. For each tolerance, we fit an ansatz near threshold $f = Ax^2 + Bx + C$, where $x = (p - p_{\text{th}})d^{-\nu}$ is the re-scaled error probability, and $A, B, C, p_{\text{th}}, \nu$ are all extracted from fits. Markers are data points, plotted as a function of the re-scaled error probability, and black lines are the ansatz. The fitted parameter p_{th} is the abort threshold that we report in the main text.

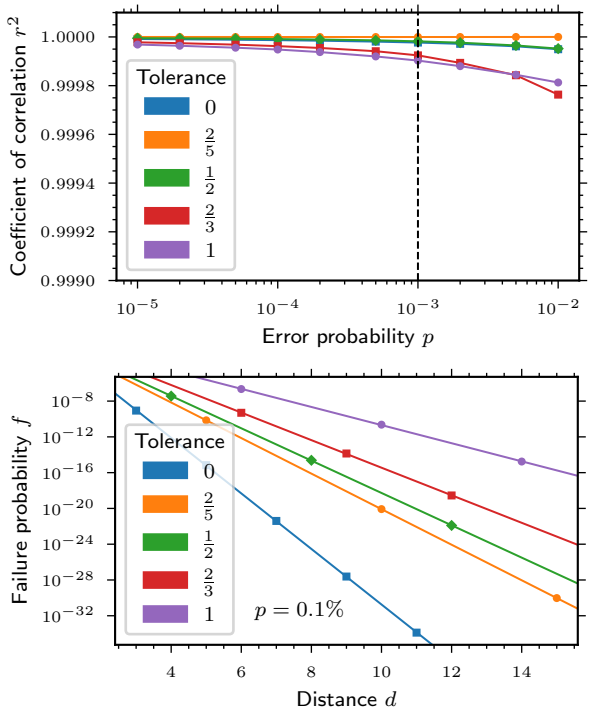


Fig. 3.11 Goodness of fit of a model describing an exponential decay of failure probabilities below the failure threshold, ie, the ansatz $\log f = d \log \Lambda(p) + \log C(p)$. (top) For each error probability, we plot the coefficient of correlation resulting from a linear fit $\log f$ against d. (bottom) We plot the failure probability f against the code distance d at a subthreshold error probability $p = 10^{-3}$. Markers are data points and the solid lines are the ansatz $\log f = d \log \Lambda + \log C$, for Λ and C extracted by fitting.

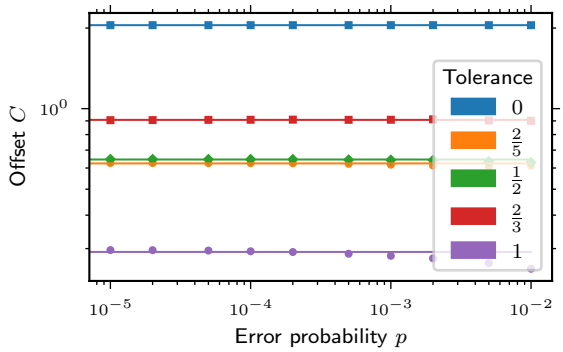


Fig. 3.12 The offset function $C(p)$ is plotted, which is extracted from a linear fit of form $\log f = d \log \Lambda(p) + \log C(p)$, where f is failure probabilities of the exclusive decoder, with the same code and error model as studied in the main text. Markers denote fitted values. We remark that low-error probability ansatz of Eq. (3.2) assumes that $C(p) \equiv C$ is a constant, and we plot this low-error probability ansatz using solid lines.

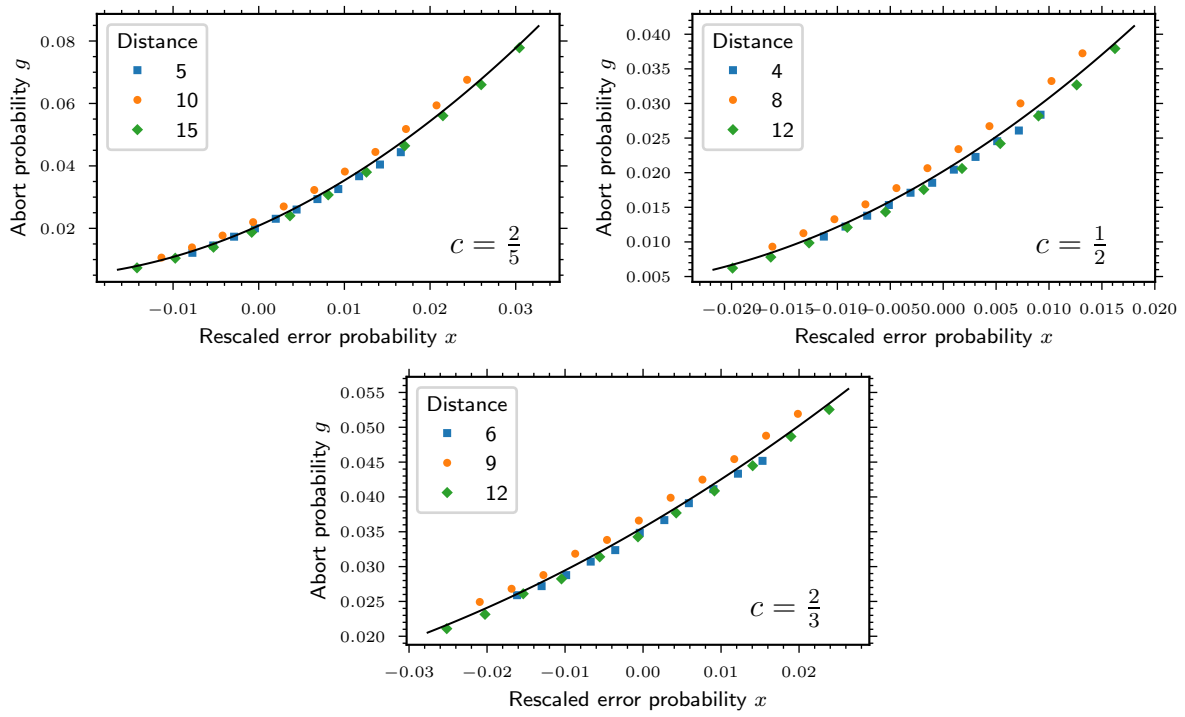


Fig. 3.13 Critical exponent plots for threshold values of abort. For each tolerance, we fit an ansatz near threshold $g = \tilde{A}x^2 + \tilde{B}x + \tilde{C}$, where $x = (p - \tilde{p}_{\text{th}})d^{-\tilde{\nu}}$ is the re-scaled error probability, and $\tilde{A}, \tilde{B}, \tilde{C}, \tilde{p}_{\text{th}}, \tilde{\nu}$ are all extracted from fits. Markers are data points, plotted as a function of the re-scaled error probability, and black lines are the ansatz. The fitted parameter $\tilde{p}_{\text{th}}$ is the abort threshold that we report in the main text.

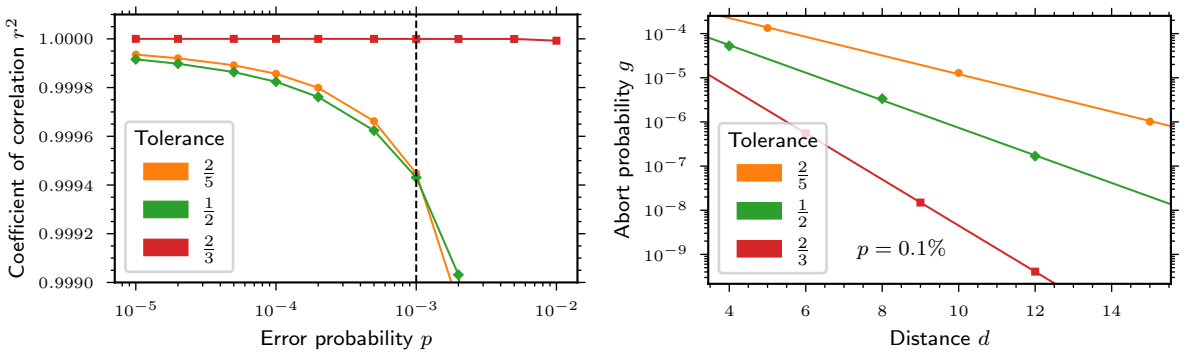


Fig. 3.14 Goodness of fit of a model describing an exponential decay of abort probabilities below the abort threshold, ie, the ansatz $\log g = d \log \tilde{\Lambda}(p) + \log \tilde{C}(p)$. (top) For each error probability, we plot the coefficient of correlation resulting from a linear fit $\log g$ against d. (bottom) We plot the failure probability f against the code distance d at a subthreshold error probability $p = 10^{-3}$. Markers are data points and the solid lines are the ansatz $\log g = d \log \tilde{\Lambda} + \log \tilde{C}$, for $\tilde{\Lambda}$ and $\tilde{C}$ extracted by fitting.

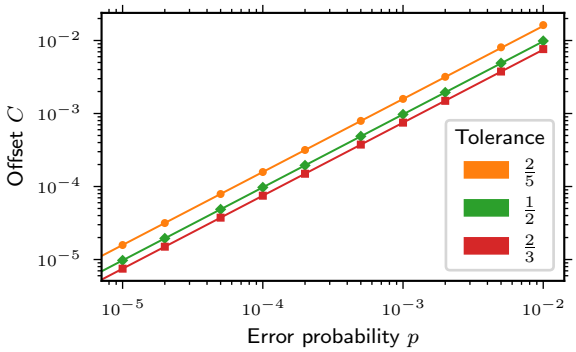


Fig. 3.15 The offset function $\tilde{C}(p)$ is plotted, which is extracted from a linear fit of form $\log g = d \log \tilde{\Lambda}(p) + \log \tilde{C}(p)$, where g is abort probabilities of the exclusive decoder, with the same code and error model as studied in the main text. Markers denote fitted values. We remark that low-error probability ansatz of Eq. (3.4) assumes that $\tilde{C}(p) = \tilde{C}p$ is proportional to the error probability, and we plot this low-error probability ansatz using solid lines.

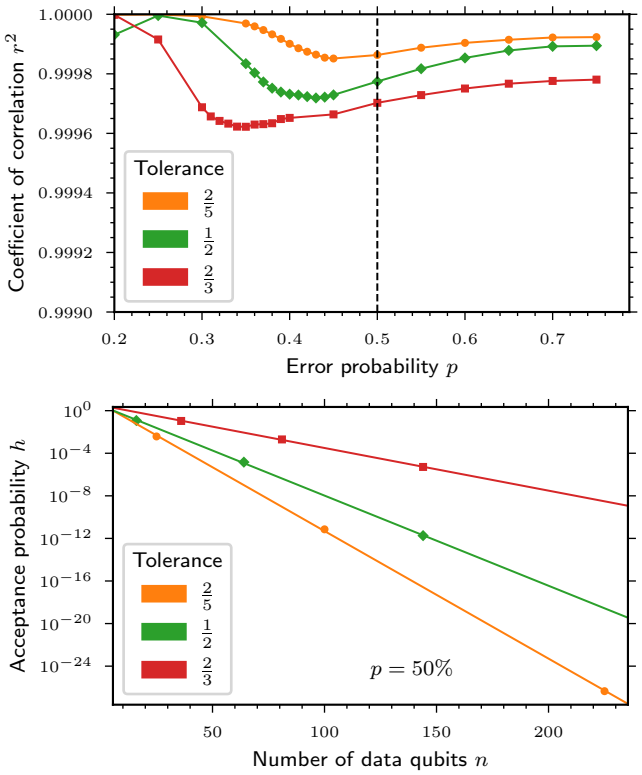


Fig. 3.16 Goodness of fit of a model describing an exponential decay of acceptance probabilities above the abort threshold, ie, the ansatz $\log h = n \log \tilde{\Lambda}_h(p) + \log \tilde{C}_h(p)$, where n is the number of data qubits with $n = d^2$ for the rotated surface code. (top) For each error probability, we plot the coefficient of correlation resulting from a linear fit $\log g$ against n. (bottom) We plot the acceptance probability h against the number of qubits n at an error probability $p = 50\%$. Markers are data points and the solid lines are the ansatz $\log g = n \log \tilde{\Lambda} + \log \tilde{C}$, for $\tilde{\Lambda}$ and $\tilde{C}$ extracted by fitting.

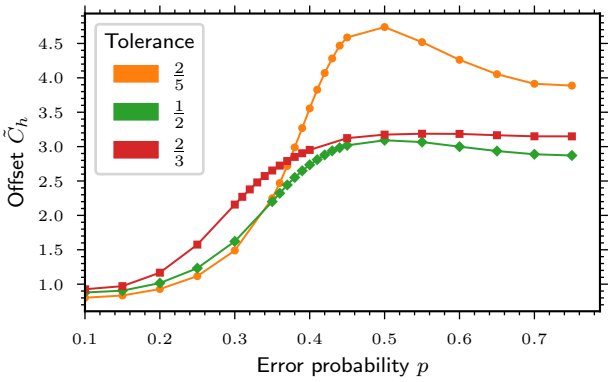


Fig. 3.17 The offset function $\tilde{C}_h(p)$ is plotted, which is extracted from a linear fit of form $\log h = n \log \tilde{\Lambda})_h(p) + \log \tilde{C}_h(p)$, where h is acceptance probabilities of the exclusive decoder, with the same code and error model as studied in the main text. Markers denote fitted values.

3.E Exclusive Union-Find

The notion of a exclusive decoder is not restricted to MWPM-based decoding, and is sufficiently general to be applied to a range of underlying standard decoders from QEC. Here, we define a exclusive variant of the union-find (UF) decoder [44].

The first stage of the standard union-find algorithm is syndrome validation, whereby an initial error consisting of a Pauli part E and erasure part σ is mapped to a final erasure σ'. If the Pauli part E is contained in the final erasure, and the final erasure does not include a logical operator, then a correction can be assured.

To generalize this, we define the survived distance d_{surv}, which will measure the amount of uncertainty in a correction. The survived code distance is defined with reference to the final erasure σ'. If we take a logical operator $\bar{L}$, then it can be decomposed into a part with weight w that shares no support with the final erasure, and a part with weight w' that is entirely supported on the final erasure, so that $w + w' \geq d$. The surviving distance is then the minimum weight w taken over all logical operators $\bar{L}$. We propose a decoder that aborts whenever

$$1 - \frac{d_{\text{surv}}}{d} > c\,. \tag{3.19}$$

We show now that this decoder can either successfully correct, or abort, all errors consisting of a weight s Pauli part E, and a weight t erasure part σ, provided $(2s + t)/2 < kd$, with k given

$$k = 1 - \frac{c}{2}\,. \tag{3.20}$$

The UF decoder breaks the initial erasure σ up into connected components, called clusters, and iteratively updates each cluster. Let l denote the sum of the diameters of the clusters at some point in the algorithm, and let w' denote the weight of E supported on the clusters at that point. The growth of a cluster is always accompanied by the discovery of at least one new qubit in the support of E, and its inclusion into the cluster. This leads to a lower bound $w' \geq (l - t)/2$. Further, if (E, σ) results in failure, then we must also have the lower bound $w \geq d_{\text{surv}}$, where w is the weight of E not supported on the final clusters. Writing $s = w + w'$, we can write

$$2s + t \geq 2d_{\text{surv}} + l \geq d_{\text{surv}} + l \geq 2d\left(1 - \frac{c}{2}\right)\,. \tag{3.21}$$

The first inequality above is the application of the bounds discussed in the previous paragraph, which assume that the result of the UF algorithm is a non-trivial logical operator. The second inequality uses the bound $d_{\text{surv}} + l \geq d$, which is true since the minimization in the definition of d_{surv} can be carried out without loss of generality by considering only logicals that do not

enter or exit each final cluster more than once. The last inequality follows from the assumption that the decoder does not abort, and is the negation of Eq. (3.19).

4

Designing time-optimal syndrome extraction circuits using Latin squares

Abstract

During a fault-tolerant computation, a quantum computer will constantly and repeatedly be executing rounds of circuits that are designed to detect and control the spread of errors. Here, we present a methodology for mapping a *schedule tensor*, which represents an error-correcting circuit, onto a *Latin square*. This allows us to take advantage of existing datasets and knowledge about the structure of Latin squares to quickly generate large numbers of circuits, expanding the search space over which error-correcting circuits can be optimized. Applying our methods to color code syndrome extraction on the honeycomb lattice, we identify approximately 105000 depth-7 circuits that extract one syndrome bit for every stabilizer (compared to a few hundred previously known circuits of this type). On the 4.8.8 lattice, we find circuits that use 8 gate cycles to extract 2 syndrome bits for every weight-4 stabilizer, and 1 syndrome bit for every weight-8 stabilizer, which is more efficient than previously known circuits of this type. We conclude by discussing certain heuristics and approaches that could be used to search through this enlarged space to find circuits with good performance at all error rates.

4.1 Introduction

In a large-scale error-corrected quantum computation, most of the physical quantum circuitry is dedicated to the purpose of extracting syndrome information that allows errors to be tracked and corrected. This offers a degree of freedom in designing the layout of this circuitry, and optimizing over the large space of possible circuit implementations can yield significant performance improvements in terms of logical accuracy, as well as the hardware demands such as the physical noise threshold and connectivity requirements.

In this work, we develop a methodology that can be used to quickly generate a large number of physically valid circuits for error-correction. We also define search criteria that are designed to reduce spacetime overheads and minimize sources of circuit error. In particular, our method assists in designing fast translationally invariant circuits that generate entanglement between two subsystems in a bi-partite system of qubits. This is well-suited, for example, for extracting syndrome information in a topological code, where the two subsystems correspond to data qubits and ancilla qubits. However, we expect that our methods may also be useful for designing more general error-correcting circuits and codes [147–152].

A key insight underpinning our work is that physically valid entangling circuits can often be represented in terms of *Latin squares*. Latin squares are a very well-studied construction in the field of combinatorics, partly out of mathematical curiosity and partly due to their usefulness to experimental design, statistics and scheduling problems [153]. A Latin square is an $n \times n$ matrix with n entries, such that no row or column contains the same entry more than once. Roughly speaking, by interpreting the row indices as specifying a qubit from one subsystem, and the column indices as specifying a qubit from another subsystem, then the Latin square structure naturally describes physically valid circuits that entangle two subsystem so that no single qubit is involved in more than one gate operation at any one time. This mapping allows us to leverage existing datasets and knowledge about the structure of Latin squares in order to quickly generate a large number of physically valid circuits.

In this work, we choose to apply our methods to the concrete task of optimizing syndrome extraction circuits for the 2-dimensional color code [154]. The color code is closely related to the surface code [41, 155], in particular inheriting its local stabilizer checks, which are implementable on a 2-dimensional chip with nearest-neighbour connectivity. However, the color code has a richer anyon structure [156] and consequentially admits a wider range of efficient and fault-tolerant logical operations, with reduced overheads for universal computation compared to the surface code [157–160].

The main drawback is that practical color code implementations have poorer logical performance than the surface code [92], in terms of threshold and the logical failure probabilities below threshold. Recent and ongoing optimization to the color code [71, 73, 161–164, 164–170],

especially in terms of its decoding [71, 73, 161–164] and syndrome extraction circuits [164–168], have brought its logical performance closer to that of the surface code, but further optimization is required to close the gap.

Syndrome extraction on the color code is difficult primarily due to its relatively high-weight stabilizers, compared to those of the surface code. There are consequentially many ways that a mid-circuit fault in the ancillary subsystem can propagate dangerously back onto the data qubits, leading to a damaging hook error [42] that diminishes the effective code distance. It is possible to use additional fault-tolerant gadgetry to catch hook errors and also to reduce connectivity requirements [103, 160, 171–174]. However, more lightweight circuit implementations have fewer locations for possible errors to occur in the first place, which results in high thresholds and good performance especially for realistic near-term error rates, although optimizing these circuits becomes important [164, 166, 175].

Using our new framework, we are able to search through a much larger space of lightweight color code circuits than has previously been possible. In particular, our enlarged search space allows exploration of syndrome extraction circuit that are allowed to vary on plaquettes of different color, which we conjecture will be an important ingredient in protecting against hook errors without requiring additional fault-tolerant gadgetry [175].

We set up and perform three searches in this work, each search looking for circuits that extract syndrome information on the color code using only a single ancilla qubit for every stabilizer in the code. In our first two searches, we consider the color code on a honeycomb lattice that has weight-6 stabilizers. Our first search yields circuits that extract a syndrome using 6 gate cycles that entangle data qubits with ancilla qubits, followed by a single gate cycle to disentangle ancilla qubits prior to measurement. Our second search yields circuits that extract a syndrome using 7 gate cycles that entangle data qubits with ancilla qubits, and do not require any disentangling of the ancilla qubits. Finally, in our third search, we consider the color code on the 4.8.8 lattice, that has weight-4 stabilizers and weight-8 stabilizers. We find circuits that use 8 gate cycles to entangle ancilla qubits with data qubits, and during that time extract 1 bit of syndrome data for every weight-8 stabilizer in the code, and 2 bits of syndrome data for every weight-4 stabilizer in the code, without requiring disentangling of the ancilla qubits prior to measurement.

The structure of this chapter is as follows. In Section 4.2, we introduce preliminary concepts such as the color code, as well as a more detailed description of Latin squares. In Section 4.3, we develop our framework as it applies to syndrome extraction circuits for topological codes. We introduce the schedule tensor, which is a key ingredient in our method, and provide general remarks that relate the schedule tensor to Latin squares, for defining a search. In Section 4.4, we apply our methods to the honeycomb lattice in Subsection 4.4.1 and Subsection 4.4.2, and to the 4.8.8 lattice in Subsection 4.4.3. Finally, we conclude in Section 4.5 and discuss directions

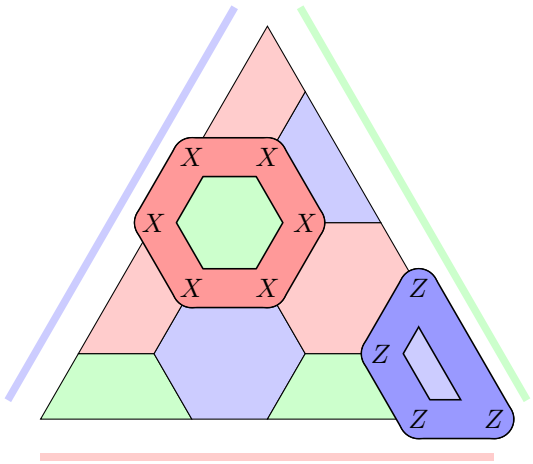


Fig. 4.1 The color code with boundaries, on the honeycomb lattice. A qubit exists for every vertex in the lattice. An X-type stabilizer is shown in the bulk, and a Z-type stabilizer is shown on the boundary. Stabilizers commute since all plaquttes share an even number of vertices. Boundaries inherit the color of the plaquette that they do not touch.

for future work. In Appendix. 4.A, we provide a picture of a valid color code circuit returned by each of our three searches, allowing an at-a-glance understanding of each of our key results.

4.2 Preliminaries

4.2.1 The color code

The 2-dimensional color code is a CSS stabilizer code, defined over a lattice with qubits located on vertices of the lattice. For every plaquette p in the lattice, there is an X-type stabilizer that we denote A_p, and a Z-type stabilizer that we denote B_p, that can be written

$$A_p = \prod_{v \in \partial p} X_v, \quad B_p = \prod_{v \in \partial_p} Z_v, \tag{4.1}$$

where ∂p denotes the set of vertex included in plaquette p. Importantly, the lattice is tri-valent and three-colorable, meaning that every vertex is contained in exactly three edges, and every plaquette can be given one of three color labels (conventionally red, green and blue), with no two plaquettes of the same color touching. In practice, the color code is often given colored boundaries, that are used to condense defects of their own color, although in this work we do not consider boundaries in any detail. A picture of the color code with colored boundaries is shown in Fig. 4.1.

We make one motivating remark about the structure and geometry of defects on the color code. An error process that separates two red (say) defects parallel to a red boundary is

Order	# reduced Latin squares	# Latin squares
1	1	1
2	1	2
3	1	12
4	4	576
5	56	161280
6	9408	812851200 ($\approx 10^9$)
7	16942080 ($\approx 10^7$)	61479419904000 ($\approx 10^{14}$)
8	535281401856 ($\approx 10^{12}$)	108776032459082956800 ($\approx 10^{20}$)

Table 4.1 Number of reduced and unreduced Latin squares.

relatively benign, since the defects can subsequently be condensed onto the red boundary using a low-weight operator to leave behind a stabilizer. On the other hand, an error process that separates a red defect orthogonally from the red boundary implements part of a least-weight logical operator, which is very damaging. The most damaging spatial orientation of a hook error, therefore, is color-dependent. We conjecture that by allowing syndrome extraction circuits to vary on plaquettes of different color, it will be possible to orient hook errors in a more benign way, and an optimal circuit will take advantage of this property.

4.2.2 Latin squares

Latin squares are a well-studied construction in the field of combinatorics, and play a key role in this work. An order-n Latin square is an $n \times n$ matrix, filled with n symbols (usually the integers $0, 1 \ldots n-1$), such that no row or column contains the same symbol more than once. An example of an order-6 Latin square is shown in Figure. 4.2.

A Latin square is said to be in *reduced form* if the first column and the first row are ordered. A Latin square can be placed into reduced form by first permuting its rows to sort the first column of the square, and then permuting the last $n-1$ columns to sort the first row of the square. Since these permutations are uniquely determined, the set of order-n Latin squares can be grouped into well-defined equivalence classes, with class representatives given by the reduced squares. Given a reduced Latin square L, there are exactly $n!(n-1)!$ equivalent squares that we collectively refer to as the *unreductions* of L.

The number of reduced Latin squares for the first few orders can be counted by formulas such as Ref. [176], and is available as sequence A000315 in the OEIS [177], reproduced here in Table. 4.1. We also make use of a complete enumeration of all reduced Latin squares of orders up to $n \leq 7$, available online at Ref. [178]. For the interested reader, we remark that this list of reduced Latin squares can be constructed, for example, by randomly sampling Latin squares

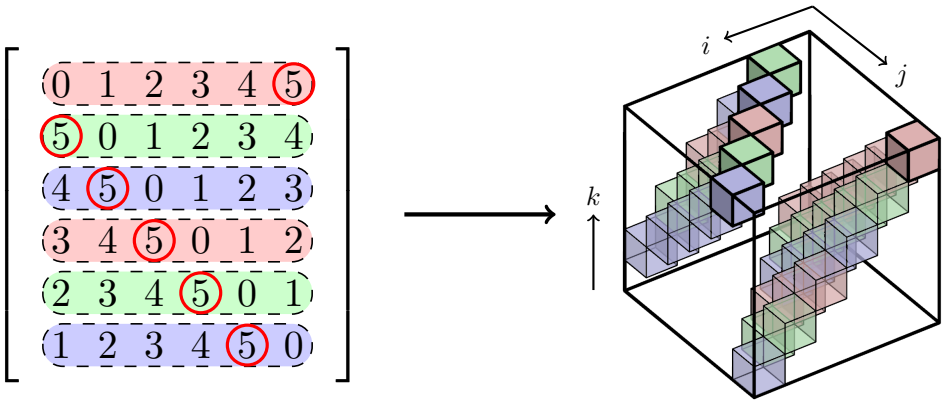


Fig. 4.2 An example of a Latin square (left), along with its binary representation (right). In the binary representation, colored blocks indicate non-zero entries. As a visual aid, colors are included to match matrix elements in the Latin square to corresponding non-zero entries in the binary representation. As an additional visual aid, the matrix elements $L_{ij} = 5$ are circled in the Latin square, and the corresponding non-zero entries are emphasized in the binary representation with increased opacity.

using an algorithm due to Jacobson and Matthews [179], until all the reduced squares have been observed at least once.

Latin squares can be represented in a few different ways, and one representation that we will make use of is the *binary representation*. The binary representation of an order-n Latin square L is an $(n \times n \times n)$ binary array. To construct the binary representation, we set the (i, j, k)-entry of the binary array to one whenever the (i, j)-entry of the Latin square is equal to k, and all other entries are set to zero. The defining property of a Latin square, that each row and each column has unique entries, translates into the property that every 1-dimensional slice in the 3-dimensional binary representation has exactly one non-zero entry.

4.3 Single-ancilla Syndrome Extraction Circuits

In this section, we provide an overview of our approach to search through syndrome extraction circuits. Certain aspects of our approach involves design choices that are made, for example, when setting up the search space. We do not attempt to provide a completely general framework to describe how these choices are made, we only provide a high-level description of the general process.

4.3.1 Structure of a SASEC

The basic building block of a SASEC is a circuit that entangles a single ancilla qubit with a stabilizer of the code. Then, the outcome resulting from the measurement of the ancilla qubit is correlated with the stabilizer eigenvalue. To measure a Z-type stabilizer, for example, we introduce an ancilla qubit in the $|0\rangle$ state, and for every data qubit in the stabilizer, we perform a CNOT gate controlled by the data qubit and targeted on the ancilla qubit. This correlates the stabilizer with the ancilla-qubit Z operator, and a measurement of the ancilla qubit in the $(|0\rangle / |1\rangle)$-basis yields the stabilizer eigenvalue. This is shown in Fig. 4.3. A building block for X-type syndrome extraction is attained by initializing in $|+\rangle$, swapping the direction of the CNOT gates, and performing measurement in the $(|+\rangle / |-\rangle)$-basis.

The *entangling part* of a SASEC can be constructed by placing down one of these building blocks for every stabilizer in the code. However, these building blocks do not exist in isolation from one another, since in general, a single data qubit will be involved in the extraction of a syndrome for more than just one stabilizer. There are two considerations that arise here. First, no single data qubit can be involved in two CNOT gates at the same time. We refer to these constraints as *physical constraints*. Second, if the circuit is not designed carefully then a Z-type ancilla can become entangled with an X-type ancilla, and vice versa, as shown in Fig. 4.3. We refer to these as *entanglement constraints*. If a SASEC does not satisfy some entanglement constraint, then we can rectify the problem by appending a *disentangling part* to the circuit immediately prior to measurement, that acts only on the ancilla qubits. However, this is not ideal, since it takes more time, increases the sources of error in the circuit, and requires connectivity between ancilla qubits.

The CNOT gates that constitute the entangling part of the circuit are grouped into *gate cycles* (sometimes called *timesteps*). Indeed, the entangling part of a SASEC is completely defined by a *schedule*, that provides a group label for every CNOT gate in the circuit. This grouping must satisfy the physical constraints, and the total number of gate cycles is called the *circuit depth*. If a SASEC has a disentangling part, then the disentangling gates can also be grouped into gate cycles, and we add the depth of the disentangling part to the circuit depth. For instance, a "depth-(X+Y)" circuit is a circuit with a depth-X entangling part, and a depth-Y disentangling part.

4.3.2 The schedule tensor

A translationally invariant SASEC over a topological code can be represented concretely in by specifying the schedule on a *primitive cell*. A primitive cell is a set of data qubits and ancilla qubits that tile to generate the full lattice. In Fig. 4.4, we show the primitive cells that we use for the honeycomb and 4.8.8 lattice.

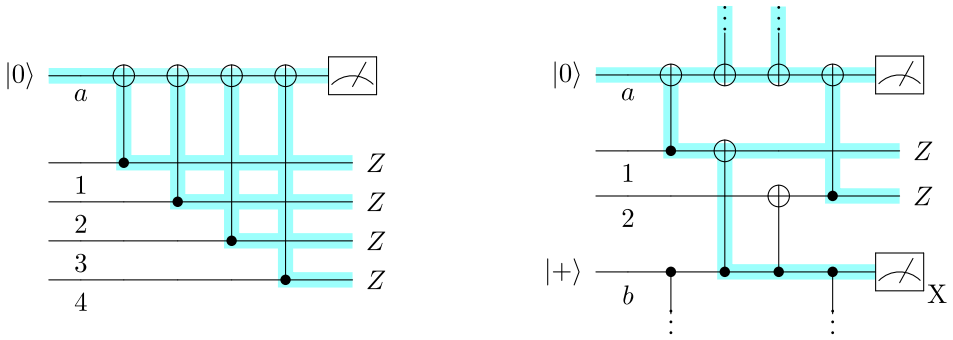


Fig. 4.3 (left) A basic building block for a syndrome extraction circuit. Ancilla qubit a is used to measure the code stabilizer $(Z_1Z_2Z_3Z_4)$. The action is $Z_a \mapsto Z_a(Z_1Z_2Z_3Z_4) \sim 1$, highlighted in blue, and a subsequent measurement of Z_a yields the stabilizer eigenvalue. (right) Interleaved syndrome extraction circuit that requires disentangling. Ancilla qubit a is used to measure code stabilizer $(Z_1Z_2\ldots)$, and ancilla qubit b is used to measure code stabilizer $(X_1X_2\ldots)$. The single-qubit operator Z_a propagates first onto data qubit 1, and then onto ancilla qubit b. It cannot do the same through data qubit 2 due to the time-ordering of the CNOT gates. The action of the circuit is $Z_a \mapsto Z_aZ_b(Z_1Z_2\ldots)$, and ancilla qubit a and ancilla qubit b end up entangled, resulting in random measurement outcomes.

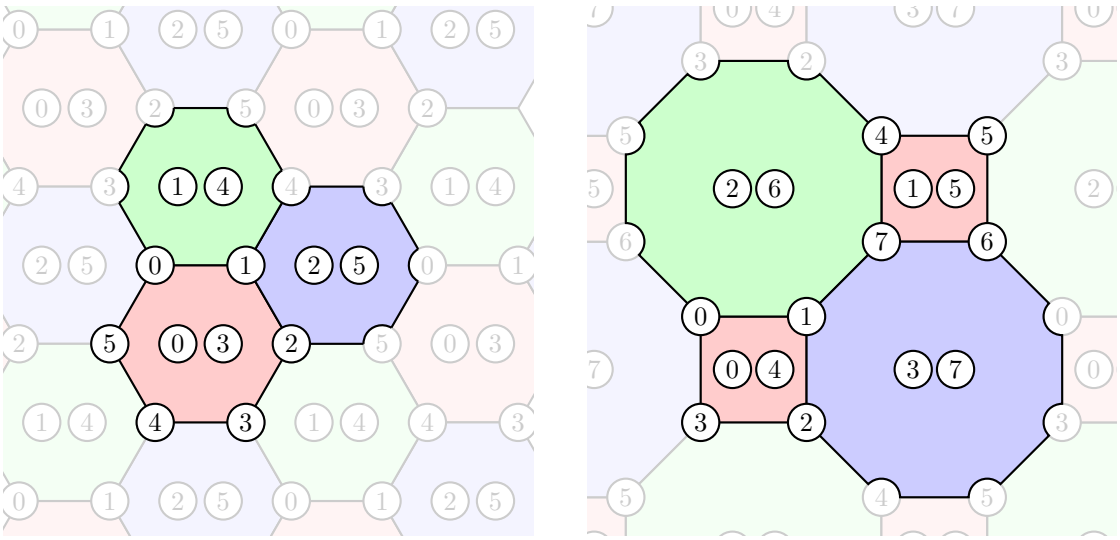


Fig. 4.4 (left) A primitive cell for the honeycomb lattice, containing six data qubits and six ancilla qubits. (right) A primitive cell for the 4.8.8 lattice, containing eight data qubits and eight ancilla qubits. For both lattices, a single primitive cell is shown in full color, and the rest of the lattice is shown faded in the background. On each plaquette, there is one ancilla for the X-type stabilizer, and one ancilla for the Z-type stabilizer. The X-type ancilla qubits are pictured off-center to the left, and the Z-type ancilla qubits are pictured off-center to the right.

A flexible way to represent the entangling part of a SASEC on a primitive cell is by using a *schedule tensor*. A schedule tensor is a $(n \times n \times T)$ array of binary indicator variables, where n is the number of ancilla qubits included in the primitive cell (equal to the number of data qubits included in the primitive cell for stabilizer codes), and T is the depth of the entangling part of the syndrome extraction circuit. The indicator variables themselves represent CNOT gates between qubits, and we have:

$$x_{ijk} = \begin{cases} 1 & \text{if ancilla qubit } i \text{ interacts with data qubit } j \text{ at time } k, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

Importantly, we remark that "ancilla qubit i" and "data qubit j" in Eq. 4.2 are *not* necessarily contained in the same primitive cell. Instead, the value $x_{ijk} = 1$ indicates that there is a set of parallel CNOT gates applied between every ancilla qubit i and data qubit j that are connected, and this occurs at timestep k. An example of a schedule tensor and the corresponding syndrome extraction circuit is given in Fig. 4.5.

One may be concerned whether it is sufficient to satisfy the physical constraints by working in primitive cells. There are two conditions that are sufficient to satisfy the physical constraints. The first condition applies to the primitive cell. We require that for any fixed data qubit or ancilla qubit, the set of neighbouring qubits are given unique labels. This is the case for the primitive cells chosen in Fig. 4.4, and can also be satisfied in general by coarse-graining the lattice into primitive cells that are sufficiently large. The second condition applies to the schedule tensor. We require that every 1-dimensional slice attained by fixing the time and data qubit index and allowing the ancilla qubit index to vary, or by fixing the time and ancilla qubit index and allowing the data qubit index to vary, must have at most one non-zero entry. Together, these conditions imply that no single qubit will be involved in more than one CNOT gate at any one time.

4.3.3 Defining a search

A search is defined by two things:

1. A mapping that allows a schedule tensor to be specified in terms of a Latin square.
2. A list of constraints on the Latin square that determines the disentangling circuit that will be required, if any is required.

The output of the search is the set of *valid squares* that satisfy all constraints in step 2. There is freedom over how these two steps are defined, and we only provide here a few remarks and general notation. First, we remark that the mapping in step 1 can be quite straightforward,

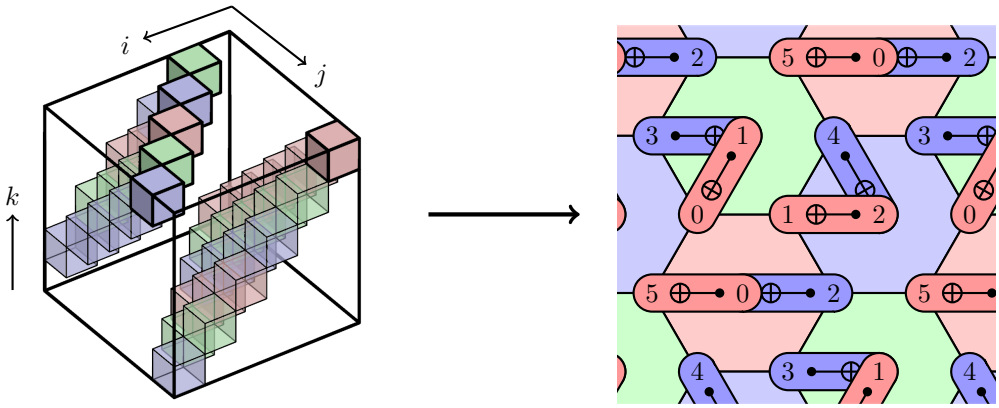


Fig. 4.5 (left) A schedule tensor for a depth-6 syndrome extraction circuit for the honeycomb lattice color code, which also happens to be a binary representation of a Latin square. (right) A depiction of the circuit represented by the schedule tensor at the $k = 5$ timestep. Behind each (i)-labelled qubit that measures an X-type stabilizer is a $(i + 3)$-labelled qubit not shown that measures a Z-type stabilizer. Note that the circuit described here will also require a second stage that disentangles the ancilla qubits prior to measurement.

as in Subsection 4.4.1 and Subsection 4.4.2, but can also be more flexible and involved as in Subsection 4.4.3. In general, this choice of mapping will determine the size of the search space, as well as how efficiently it can be searched through.

In regards to step 2, we introduce a notation for specifying an entanglement constraint on a Latin square. Take two row indices i and j, and a set of column indices $k_1, k_2, \ldots, k_m$, over a Latin square L. Then, we introduce the following notation for a constraint:

$$\{ij\}\{k_1 k_2 \ldots k_m\} \quad \text{means} \quad (L_{ik_1} - L_{jk_1})(L_{ik_2} - L_{jk_2}) \ldots (L_{ik_3} - L_{jk_3}) > 0 \tag{4.3}$$

At a physical level, an entanglement constraint is specified by listing a pair of ancilla qubits of different type, as well as their shared nearest-neighbour data qubits. Then, the constraint is on the parity of time-orderings of CNOT gates between the ancilla qubits and the data qubits, as in Ref. [166]. The map between this physical picture and the Latin square constraints in Eq. 4.3 is provided by step 1 above.

We remark that we will often find it useful to group entanglement constraints into different types in order to tailor the search, depending on what we are looking for. For example, in Subsection 4.4.1, we find circuits that violate some entanglement constraints that are particularly easy to clean up with a disentangling circuit prior to measurement, and in Subsection 4.4.3, we speed up our search significantly by checking certain entanglement constraints early, using a shortcutting method.

4.4 Results

In this section, we provide details of three searches that we performed. In regards to the honeycomb code, we report depth-(6+1) honeycomb circuits in Subsection. 4.4.1, and depth-7 honeycomb circuits in Subsection 4.4.2. In regards to the 4.8.8 lattice, we report time-optimal circuits in Subsection 4.4.3.

4.4.1 Depth-(6+1) honeycomb circuits

In this subsection, we search for circuits with depth-6 entangling parts, but we allow these circuits to leave entanglement between ancilla qubits that are located on the same plaquette. Then, the leftover entanglement structure is very simple: the ancillary subsystem is described by a product state over Bell pairs (ignoring entanglement with the data qubits), requiring only a single additional gate cycle to disentangle.

Search definition The depth-6 search over the honeycomb code is the simplest search to describe. This is because the schedule tensor has dimension $6 \times 6 \times 6$, and since every data qubit interacts with every ancilla qubit exactly once, then the schedule tensor is already the binary representation of a Latin square. In particular, the $(i,\ j)$-th entry of a Latin square simply gives the time at which ancilla qubit i interacts with data qubit j, or equivalently gives the location of the non-zero entry along the (i, j)-slice in the schedule tensor.

The set of entanglement constraints is also straightforward to write down in this case, by inspection of the primitive cell in Fig. 4.4. We identify two types of constraints. Type-I constraints relate ancilla qubits on adjacent plaquettes. For example, pick an ancilla qubit 0, and an ancilla qubit 4 directly above it. These two qubits are of opposite Pauli type, but share connections to data qubits 0 and 1. Therefore, $\{04\}\{01\}$ is a type-I entanglement constraint. Type-II entanglement constraints relate ancilla qubits on the same plaquette. For example, $\{03\}\{012345\}$ is a type-II entanglement constraint. The full set of entanglement constraints is provided in Table 4.2.

The search is over all order-6 Latin squares, and the search criteria is for valid squares that satisfy all the type-I entanglement constraints. We allow circuits that may violate type-II entangling constraints.

Valid circuits In reporting our results, we consider that two circuits may be related by a simple lattice transformation, such as a translation, rotation, reflection or by interchanging the role of X-type and Z-type ancilla qubits. Here, and for the remainder of this work, we report

Type I constraints		
{31}{01}	{32}{12}	{31}{23}
{32}{34}	{31}{45}	{32}{50}
{42}{25}	{40}{54}	{42}{41}
{40}{10}	{42}{03}	{40}{32}
{50}{43}	{51}{30}	{50}{05}
{51}{52}	{50}{21}	{51}{14}
Type II constraints		
{03}{012345}	{25}{012345}	{14}{012345}

Table 4.2 Honeycomb constraints

all valid circuits returned by our search, and in parenthesis we report only valid circuits that are distinct up to simple lattice symmetries.

We searched over close to a billion Latin squares, and found that there were 6444 (162) circuits that left behind a Bell pair on all three colors of plaquettes, there were 0 (0) circuits that left behind a Bell pair on two of the three colors of plaquettes, and there were 288 (4) circuits that left behind a single Bell pair on only one out of the three colors of plaquettes. Unfortunately, there were no circuits included in our search that left behind no entanglement on the ancillary subsystem. An example of a circuit leaving behind a Bell pair only on green plaquettes, and satisfying all other entanglement constraints, is shown in Fig. 4.9.

All valid circuits returned by our search require a single disentangling step to be applied before the ancilla qubits can be measured. During the disentangling part of the circuit, the data qubits are idling and accruing another round of errors as they idle. Alternatively, we could imagine that we hot-swap another ancilla qubit in to replace any ancilla qubit that needs to be disentangled. Then, while ancilla qubits from one round of syndrome extraction are being disentangled and measured, the entangling part of the circuit for the next round of syndrome extraction can begin. This requires more physical qubits, and more connectivity, but may result in faster circuits and higher thresholds. Finally, we note that it may be possible to make use of the leftover entanglement, or to design circuits that intentionally generate entanglement between ancilla qubits, in order to ease connectivity requirements for Pauli measurement [168].

4.4.2 Depth-7 honeycomb circuits

In this subsection, we allow an additional timestep into the honeycomb circuits that we search over, so that we are searching over depth-7 SASECs rather than the time-optimal depth-6 SASECs. The trade-off is that there are many more depth-7 SASECs than depth-6 SASECs. This allows us to keep the search criteria strict, and the circuits that we find require no disentangling part.

Search definition A depth-7 circuit on the honeycomb lattice is described by a $(6 \times 6 \times 7)$ schedule tensor. A way to construct such a schedule tensor is to start with a binary representation of an order-7 Latin square, which is a $(7 \times 7 \times 7)$-tensor, and take a $(6 \times 6 \times 7)$ sub-block. Equivalently, in terms of Latin squares, we start with a 7×7 Latin square, and look only at the restriction to the first six rows and columns. The row indices of the restricted square correspond to ancilla qubits, the column indices correspond to data qubits, and the entanglement constraints in Table 4.2 are applied without any modification to the restricted square to define a valid SASEC.

We have two remarks about the size of the search space. Our first remark is that the search space does not include all depth-7 physically valid SASECs. As a counter-example, consider a depth-7 circuit that is really a depth-6 circuit, followed by idling gates. This circuit is represented by a 6×6 Latin square, which can not be found as the restriction of a 7×7 Latin square, since the last row and column of the (7×7)-square would have to be filled with just one repeated symbol. Therefore, the search that we perform here is a limited search.

Our second remark is that the size of even this limited search space is large—it is equal to the number of 7×7 Latin squares, which from Table 4.1 is approximately 10^{14}. To see this, observe that there is a one-to-one correspondence between restrictions of (7×7)-squares, and (7×7)-squares themselves. This is a result of the fact that the restriction has a well-defined inverse operation, since the first six entries in any row or column of a (7×7)-square uniquely determines the last entry.

Valid circuits We do not attempt to search through every restricted square, since the search space is very large. Instead, we sampled 20 random unreduced squares for every reduced square, sampling around 300 million squares. In total, we found 42 valid circuits. An example of one of these circuits is shown in in Fig. 4.10, in Appendix 4.A. Extrapolating from this result, we estimate that the total search space contains approximately 7 million (105000) valid circuits, and valid circuits are sufficiently dense in the search space that they can be generated at a reasonable rate.

To contextualize our result, we remark that the circuits reported here do not provide a speed improvement on existing circuits, since depth-7 circuits have been found previously in Ref. [166]. However, previously only a few hundred[1] depth-7 circuits were known, and they were limited in the sense that the schedule was not allowed to vary on plaquettes of different color. We therefore consider our result to be of practical significance. Since our circuits can very on plaquettes of different color, we consider it likely that an optimization over this larger space will find circuits that orient hook errors in a more benign way, yielding a further improvement

[1]There is a minor discrepancy on the exact number in the literature, with 292 reported in Ref. [164], and 234 reported in Ref. [166]

L_1	L_2	L_3	L_4
{17}{12}	{52}{56}	{43}{12}	{06}{56}
{52}{32}	{17}{76}	{06}{32}	{43}{76}
{17}{03}	{52}{47}	{43}{03}	{06}{47}
{52}{01}	{17}{45}	{06}{01}	{43}{45}
{15}{0123}	{15}{4567}	{04}{0123}	{04}{4567}
Global constraints			
{07}{12}	{42}{32}	{07}{03}	{42}{01}
{53}{12}	{16}{32}	{53}{03}	{16}{01}
{16}{56}	{53}{76}	{16}{47}	{53}{45}
{42}{56}	{07}{76}	{42}{47}	{07}{45}
{27}{24}	{63}{24}	{27}{17}	{63}{17}
{27}{60}	{63}{60}	{27}{53}	{63}{53}
{26}{01234567}	{37}{01234567}		

Table 4.3 4.8.8 constraints

in terms of higher thresholds and smaller subthreshold failure probabilities for the color code implemented on the honeycomb lattice.

Finally, we remark that it may be possible to use advanced methods from computational group theory to speed up the search in this case. This is because the physical lattice symmetries can be made to fix the first row and column of a Latin square, since a dummy row and column is part of the construction. Then the lattice symmetries are a subgroup of the group of permutations that reduce a Latin square, which can be identified one-to-one with the unreductions of a given reduced square. The problem of finding valid Latin squares corresponds to finding the transversal of the symmetry group in the set of unreduced squares, which is a well-studied problem [180]. We did not attempt a more advanced search, but we expect a speed up of up to the order of the size of the symmetry group ($72\times$) may be possible.

4.4.3 Time-optimal 4.8.8 circuits

In this subsection, we search over SASECs for the color code defined on the 4.8.8 lattice. The 4.8.8 lattice poses an interesting challenge, since it has stabilizers of differing weight. A time-optimal SASEC will extract syndrome measurement data associated with red plaquettes every four gate cycles, and will extract syndrome measurement data associated with green and blue plaquettes every eight gate cycles. We successfully adapt out methods to this setting and find depth-8 SASECs that achieve this time-optimality, providing a demonstration of the flexibility of our Latin-square methodology.

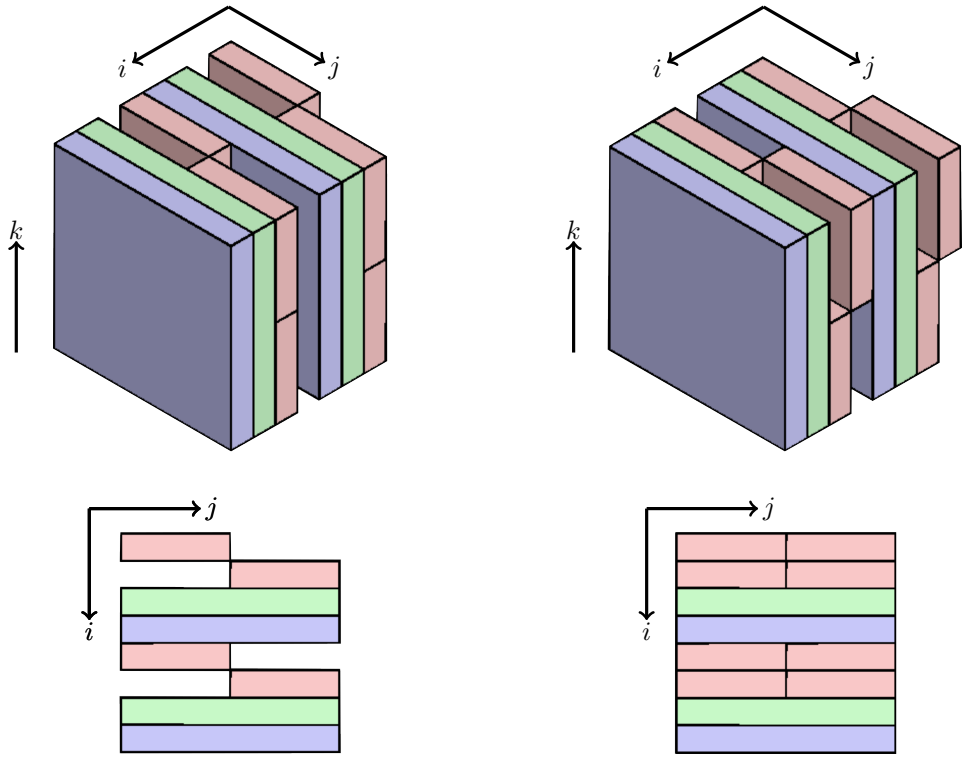


Fig. 4.6 (left) A view of a schedule tensor x_{ijk} that represents a syndrome extraction circuit for the 4.8.8 color code. Each colored block represents a part of the circuit that extracts exactly one syndrome bit. There are two red blocks for every red plaquette, because syndrome data associated with red plaquettes is extracted twice every cycle, once in the first four gate cycles and once in the second four gate cycles. Within each colored block, there is exactly one non-zero entry along every 1-dimensional slice. (right) Views of the schedule tensor after being transformed according to Eq. (4.4). The entire 3-dimensional binary array is now a representation of a Latin square, since every 1-dimensional slice has exactly one non-zero entry. The red blocks that are not visible in the isometric projection are fixed by the transformation.

Search definition The schedule tensor for a depth-8 SASEC on the 4.8.8 lattice does not immediately have the structure of a Latin square, since the Tanner graph over the primitive cell defined in Fig. 4.4 is not fully connected. Assuming that red plaquettes are measured twice in a syndrome extraction cycle, then in the schedule tensor this means that a 1-dimensional slice obtained by fixing a red ancilla qubit i, and a data qubit j, and allowing the time index to vary, will either have no non-zero entries if i is not connected to j, or two non-zero entries if i is connected to j.

Fortunately, there is a simple transformation on the schedule tensor that can turn it into a representation of a Latin square. The transformation shuffles around sub-blocks of the schedule tensor. We introduce the permutation σ_{488} on $\{0, 1, \ldots, 7\}$ that swaps 0 with 1 and also swaps 4 with 5. In cycle notation, this is written $\sigma_{488} = (01)\,(45)$. Then, the transformed schedule

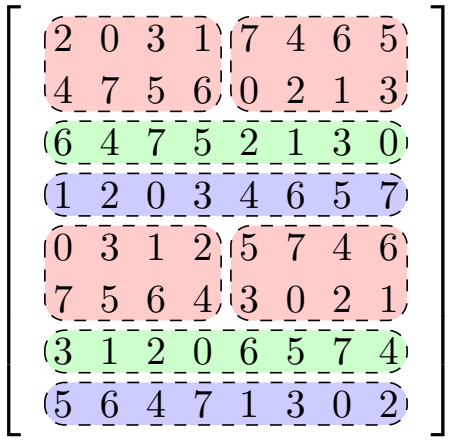


Fig. 4.7 A Latin square representing a schedule of a syndrome extraction circuit for the 4.8.8 color code. Column indices denote data qubits, but row indices do not denote ancilla qubits. Instead, each colored block can be associated with an ancilla qubit. The top left red block contains interactions involving the 0-ancilla qubit. The top right red block contains interactions involving the 1-ancilla qubit. The bottom left red block contains interactions involving the 4-ancilla qubit. Finally, the bottom right red block contains interactions involving the 5-ancilla qubit. A green or blue block at row index i contains interactions involving the i-ancilla qubit.

tensor is defined below:

$$\tilde{x}_{ijk} = \begin{cases} x_{ijk} & k < 4, \\ x_{\sigma_{488}(i)jk} & k \geq 4. \end{cases} \tag{4.4}$$

We direct the reader towards Fig. 4.6 to convince themselves that this does indeed transform x_{ijk} into a representation of a Latin square.

Finally, we can write down the Latin square corresponding to the transformed schedule, however we have to make sure that we interpret the entries of this Latin square correctly. Since the transformation from Eq. (4.4) permutes row indices over part of the schedule, it is no longer the case that row indices of the Latin square will correspond to ancilla qubits. Instead, we refer the reader to Fig. 4.7 to properly interpret all entries of the Latin square representation of a 4.8.8 schedule.

We remark that not all 8×8 Latin squares correspond to SASECs on the 4.8.8 lattice, since we want to be able to measure the red plaquettes halfway through the circuit. For example, this means that in row 0, the first four matrix elements must contain the integers $0, 1, 2$, and 3, since red plaquettes will be measured after the fourth timestep. The same constraint also applies to the first four entries of row 4, to the last four entries of row 1, and to the last four entries of row 5, where we note that row indexing starts from 0. We refer to these constraints collectively as *mid-cycle measurement* constraints.

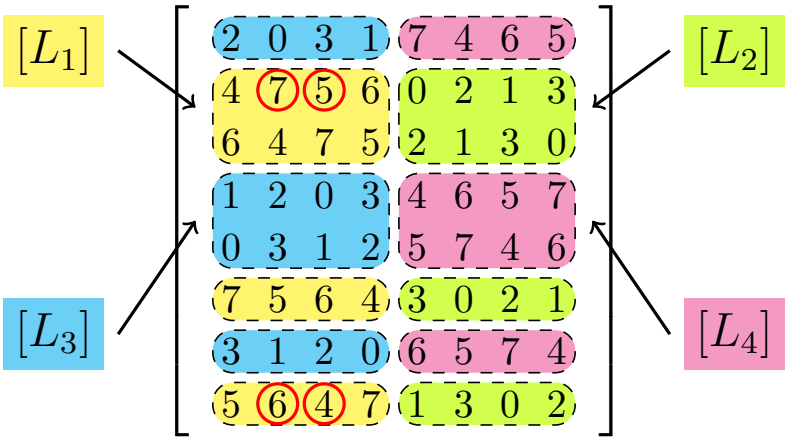


Fig. 4.8 An 8×8 Latin square is constructed from four 4×4 Latin squares. L_2 (green) and L_3 (blue) contain the integers $0, 1, 2, 3$, and L_1 (yellow) and L_4 (pink) contain the integers $4, 5, 6, 7$. The matrix elements involved in the constraint $\{17\}\{12\}$ are circled in red, and can be checked by looking only at L_1.

Shortcut search Rather than initiate a search through all 8×8 Latin squares, of which there are approximately 10^{20}, we restrict the search space by considering only candidate 8×8 Latin square that are constructed from four 4×4 Latin squares. The details of this construction are provided in Fig. 4.8. This limited search space still includes a total of $576^4 \approx 10^{12}$ squares, however we can speed up our search dramatically by shortcutting the checking of entanglement constraints. The entanglement constraints on the 4.8.8 lattice can be grouped into constraints that are supported only on one of the (4×4)-squares shown in Fig. 4.8, and global constraints that have to be checked on the full (8×8) square. For example, from Fig. 4.8, it can be seen that the constraint $\{17\}\{12\}$ involves matrix elements that all come from L_1, whereas the constraint $\{07\}\{12\}$ involves some matrix elements from L_1, and some matrix elements from L_3. A list of all entanglement constraints and their groupings is given in Table 4.3. With these groupings, we can shortcut the search by rejecting a (4×4)-square immediately if it fails a constraint. We find that there are only 16 valid configurations for each (4×4)-square, leaving only $16^4 = 65536$ (8×8)-squares that need to be explicitly checked, which can be searched through in a matter of seconds.

We remark that there are many ways to construct an 8×8 Latin square from four 4×4 Latin squares. The construction in Fig. 4.8 was carefully chosen such that a resulting (8×8)-square could feasibly satisfy all the entanglement constraints from Table 4.3. An additional nice property of the construction in Fig. 4.8 is that it guarantees that the resulting (8×8)-squares will satisfy the mid-circuit measurement constraints, as well as certain global constraints such as $\{07\}\{12\}$, for free.

Valid circuits We searched through a total of $576^4 \approx 10^{12}$ squares using our shortcutting method and found 2304 circuits that satisfied all of the entanglement constraints in Table. 4.3. After taking symmetries into account, we discovered some new circuits, bringing the total up to 5184 (162) circuits. This is because the action of the lattice symmetries does not respect the process that we use to construct squares in our limited search space. As a result, the number of squares that we have effectively searched through, ie squares for which we have checked a symmetry-equivalent square, is larger than $576^4 \approx 10^{12}$. An example of a valid circuit is shown in two pages in Appendix 4.A, with the first four timesteps shown in Fig. 4.11, and the second four timesteps shown in Fig. 4.12.

These circuits are able to extract syndrome information on the red plaquettes at twice the rate as standard 4.8.8 circuits found in the literature. In theory, this allows red defects to be tracked and corrected more accurately than blue defects or green defects. It would be interesting to consider how a decoder can be designed to use this additional syndrome information most effectively, or whether there are fault-tolerant gate constructions that can take advantage of the more reliable data for red defects. Finally, we remark that recent work has explored measuring stabilizers at differing frequencies for local implementations of LDPC codes [181].

4.5 Discussion

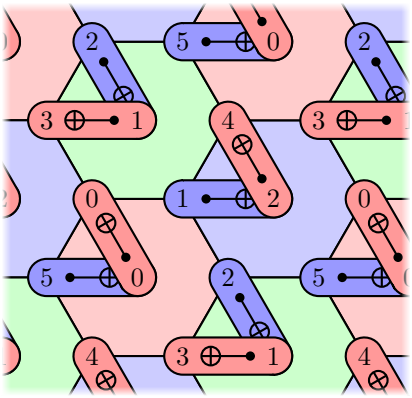
We have developed a methodology for generating a large number of physically valid, translationally invariant stabilizer circuits that are defined on bi-partite connectivity graphs. We applied our methodology to the two-dimensional color code, and generated many instances of space and time-optimized syndrome extraction circuits for both the honeycomb lattice and the 4.8.8 lattice. In particular, we filtered these SASECs by minimizing the depth of the disentangling part of the circuit to be applied prior to measurement, with a particular focus on enumerating SASECs that did not require a disentangling part, where possible. We posit that by minimizing the number of ancilla qubits and timesteps used, our circuits will perform well in the regime of realistic near-term error rates, since with fewer gates there are also fewer locations where errors can occur.

A very immediate and natural follow-up question, still focusing on color code optimization, is what kinds of properties should we look for with regards to the entangling part of the circuit? Put another way, it remains to benchmark the many circuits that we found to further optimize for logical performance at all error rates. In the regime where errors are very unlikely to occur, we expect that performance is going to be dominated to first order by the geometry of hook errors, since the effective code distance sets the sub-threshold scaling of logical failure probabilities with regards to the code distance. In practice our circuits could be filtered, for example, by performing fault-tolerant simulations of a sample of the circuits that we report

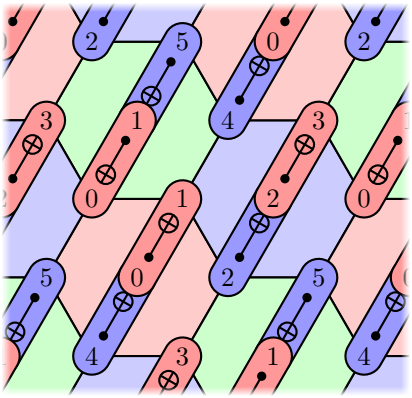
here, and applying a supervised learning technique to extract features with good predictive power for quantities of interest, such as failure probabilities far below threshold. It is worth remarking, however, that an optimal technique may not be a single fixed circuit. For example, it may be the case that by randomly varying the syndrome extraction circuit round to round, then there will be no single direction that hook errors can compound together harmfully, and instead hook errors will propagate defects in a diffusive, random-walk fashion. Finding color code circuits optimized for logical performance at all error rates is an active and ongoing area of research, which will have a large impact on the practicality of color code based quantum computing.

Moving away from color codes, we remark that the bi-partite system of qubits described by our circuits does not necessarily need to correspond to a set of data qubits and a set of ancilla qubits in a syndrome extraction circuit for a stabilizer code. For example, the circuits based on time-dynamics such as Ref. [150], or the middle-out circuit of Ref. [167], do not use ancilla qubits, but are still defined over a bi-partite connectivity graph (the hex-grid) and could therefore be described by our methods. A recent shift in perspective is to consider error-correcting circuits (also called space-time codes [147]) as a fundamental construction, as opposed to starting with an error-correcting code and designing a circuit as an implementation detail. In the past, brute-force searching over translationally invariant stabilizer codes has yielded codes with interesting properties [61]—an intriguing possibility is to use our methods to allow a brute-force search over translationally invariant stabilizer circuits, to hopefully find space-time codes with interesting properties. A key question that will progress this line of inquiry is what fault-tolerant criteria should be used to filter the most robust stabilizer circuits?

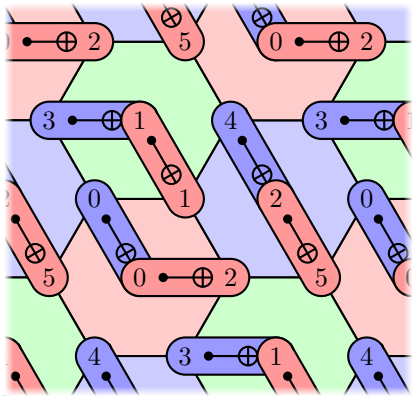
4.A Examples of circuits



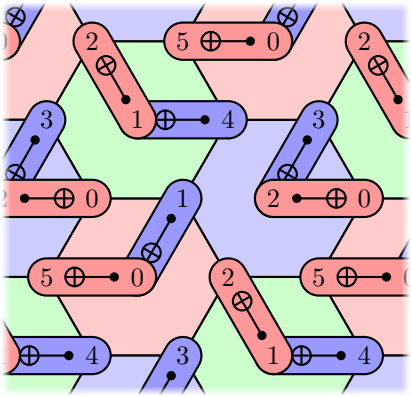
$t = 0$



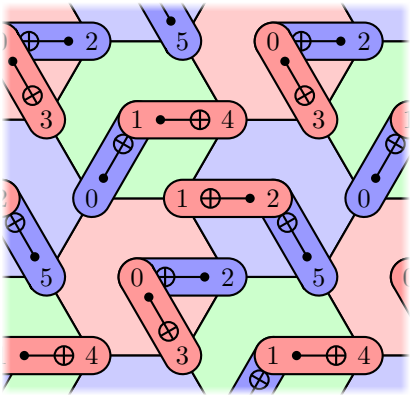
$t = 1$



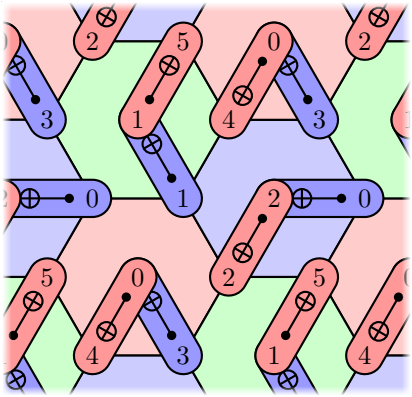
$t = 2$



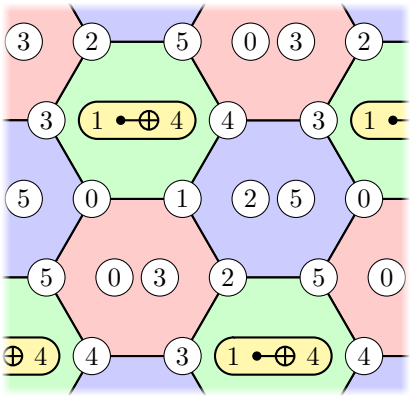
$t = 3$



$t = 4$

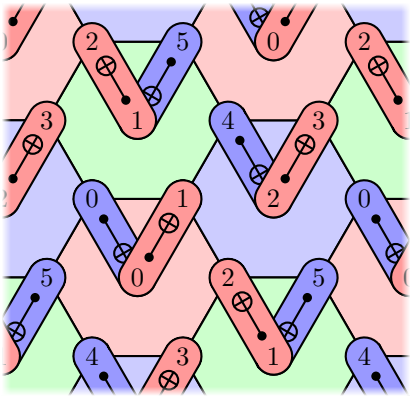


$t = 5$

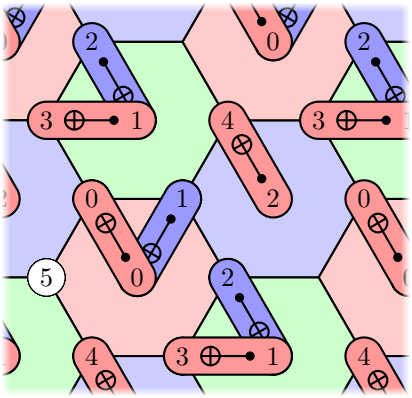


$t = 6$

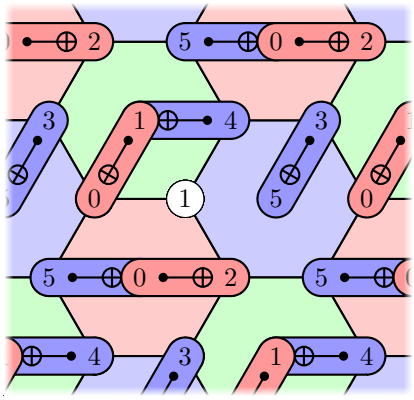
Fig. 4.9 A depth-(6+1) circuit for syndrome extraction on the honeycomb lattice color code. On each plaquette, there are two ancilla qubits, drawn on top of each other, with the X-type ancilla qubit labels covering the Z-type ancilla qubit labels. In timestep 7, all ancilla qubit labels are shown.



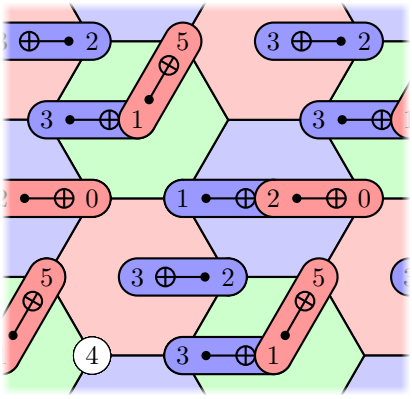
$t = 0$



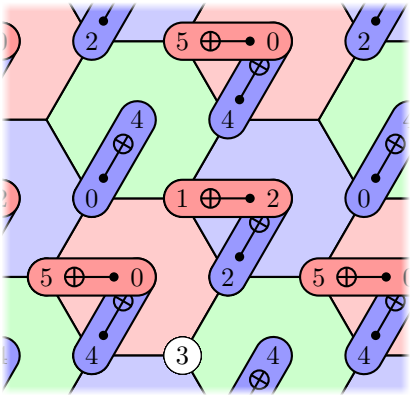
$t = 1$



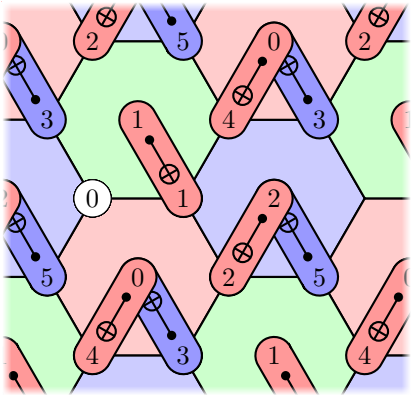
$t = 2$



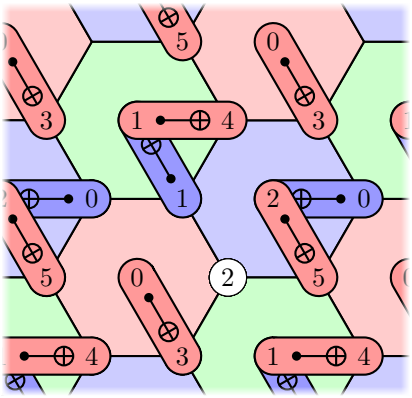
$t = 3$



$t = 4$

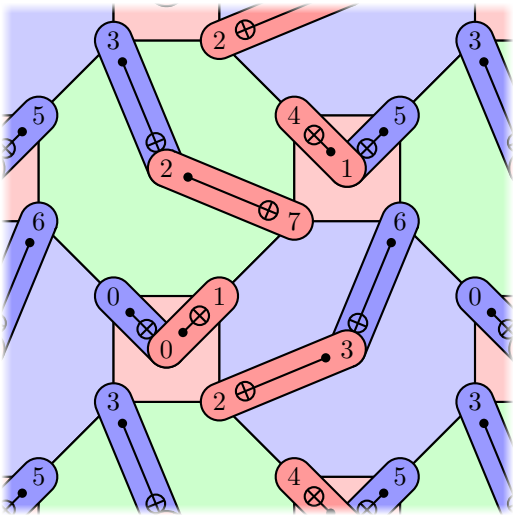


$t = 5$

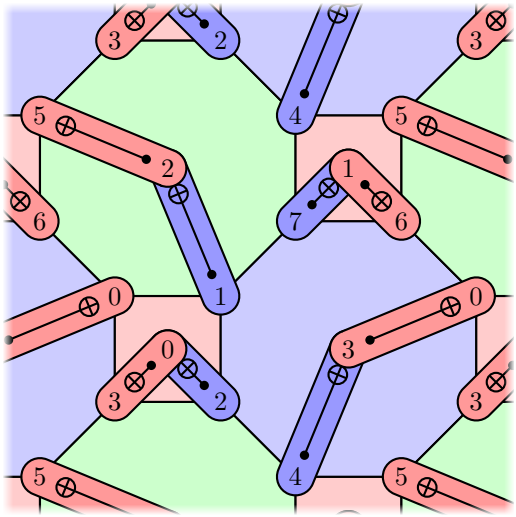


$t = 6$

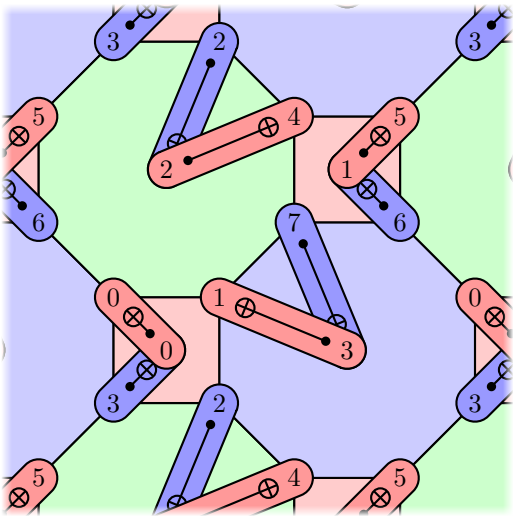
Fig. 4.10 A depth-7 circuit for syndrome extraction on the honeycomb lattice color code. On each plaquette, there are two ancilla qubits, drawn on top of each other, with the X-type ancilla qubit labels covering the Z-type ancilla qubit labels.



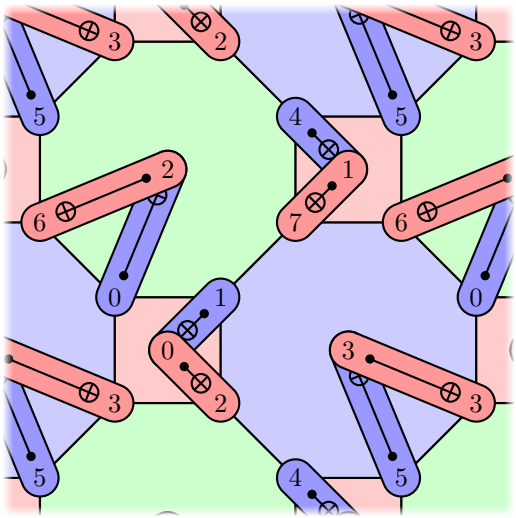
$t = 0$



$t = 1$

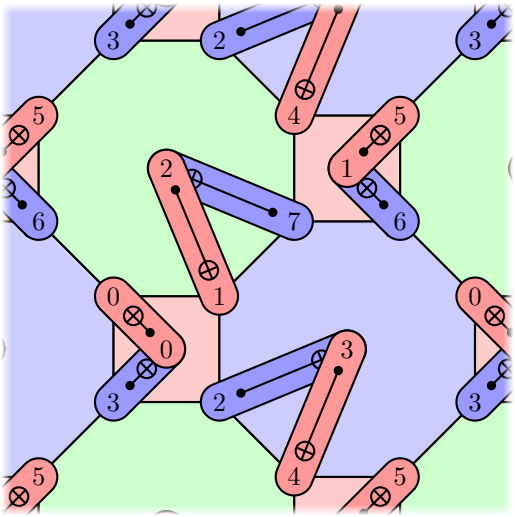


$t = 2$

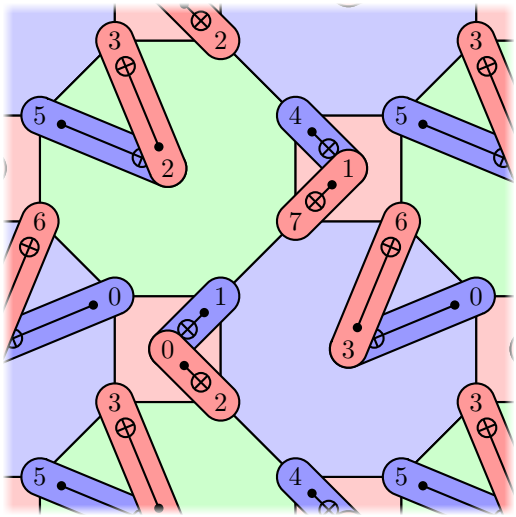


$t = 3$

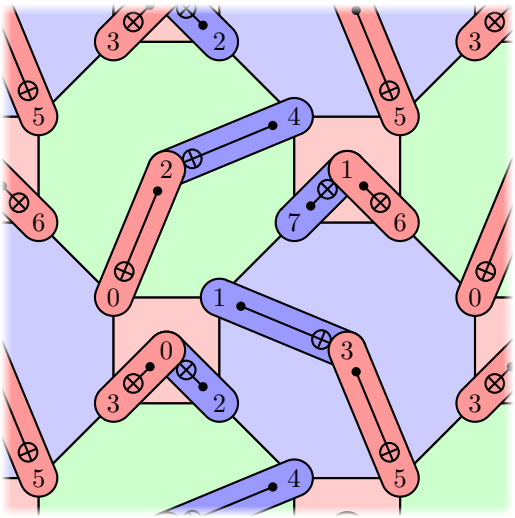
Fig. 4.11 Timesteps $0, 1, 2, 3$ of a depth-8 circuit for syndrome extraction on the 4.8.8 lattice color code, extracting two syndrome bits for every red plaquette, and one syndrome bit for every green and blue plaquette. Timesteps $4, 5, 6, 7$ are shown on the next page. On each plaquette here and on the next page, there are two ancilla qubits, drawn on top of each other. The X-type ancilla qubit labels are covering the Z-type ancilla qubit labels. All ancilla qubits on red plaquettes are measured and reinitialized after the $t = 3$ timestep.



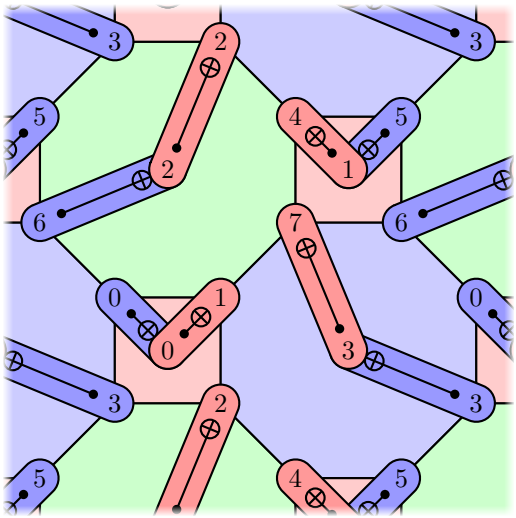
$t = 4$



$t = 5$



$t = 6$



$t = 7$

Fig. 4.12 Timesteps $4, 5, 6, 7$ of a depth-8 circuit for syndrome extraction on the 4.8.8 lattice color code, extracting two syndrome bits for every red plaquette, and one syndrome bit for every green and blue plaquette. Timesteps $0, 1, 2, 3$ are shown on the previous page. All ancilla qubits are measured and re-initialized after the $t = 7$ timestep.

5

Conclusion

In this thesis, we have investigated challenges that must be overcome and techniques that can be useful when implementing the theory of quantum error correction in practice. In particular, we are motivated by recent advances in experimental quantum error correction, especially by the prospect of current-day physical systems continuing to scale up, becoming capable of larger and more complex fault-tolerant computations as they do so. Our thesis highlights a dichotomy that exists within the development of fault-tolerance theory as it applies to practical questions: (i) how can the theory of quantum error correction be optimized and compactified down to fit within the constraints of a near-term device with limited components, and (ii), in doing so, how can the techniques and frameworks that are developed be scaled back up to inform or renew our understanding of the theory of fault-tolerance into the future?

In Chapter 2, we addressed the question of whether classical decoding calculations, which are required to track and correct errors, can be performed in real-time at a sufficient pace to keep up with the clock cycle of a quantum computer. We proposed a solution based on distributing and pipelining the processing of syndrome data. Specifically, we used a greedy predecoder that reduced the complexity of the decoding problem to be sent to a more sophisticated main stage decoder. We took care to consider practical questions, such as where the processing will be carried out and how much communication bandwidth will be required, but we also designed our approach to scale, since the predecoding part of our solution was completely local with minimal computational resource requirements. We found substantial improvements in the runtime of the

global decoder and the communication bandwidth by using the predecoder. For instance, To achieve a logical failure probability of $f = 10^{-15}$ using qubits with physical error rate $p = 10^{-3}$ and a distance $d = 22$ code, we found that the bandwidth cost was reduced by a factor of 1000, and the time taken by a matching decoder was sped up by a factor of 200. To achieve this target failure probability, the predecoding approach required a 50% increase in the qubit count compared with the optimal decoder. We remark that in this chapter we re-examined an assumption that has often been made in the development of QEC—that classical decoding and communication can be performed instantaneously. After dropping this assumption, careful considerations such as ours of how decoding will be implemented in practice has led to the field of real-time decoding, which is related to the more general field of micro-architecture design [74, 182], with the understanding that the entire classical-quantum interface, including specification of where and when classical processing will be done, will need to scale up with the quantum computer. This multidisciplinary field is especially interesting and active at the moment, due to the diversity of promising hardware modalities that are currently building infrastructure to transition from research-scale to industry-scale quantum computing.

In Chapter 3, we investigated post-selection. Post-selection is a powerful tool in experimental QEC [22–27, 29, 32, 35] used for boosting logical performance when hardware components are limited, but it is sometimes considered "cheating" under the presupposition that it will not scale. We posed the question: can post-selection have a role to play in the future of fault-tolerant quantum computing at scale, and if so, what is the nature of that role? Focusing on the surface code, we introduced a parameterized family of exclusive decoders, which could abort on decoding instances that were deemed too difficult. We developed new numerical sampling methods specifically designed to simulate exclusive-style decoders with post-selection. We saw significant improvements to post-selected logical performance, and observed a threshold of 50% under depolarizing noise for the surface code (or 32(1)% for the fault-tolerant case with phenomenological measurement errors), and up to a quadratic improvement in logical failure rates below threshold. A key and novel result, however, was the discovery of a low-error rate regime where we observed that the exclusion rate decayed with code distance, providing a pathway for scalable and time-efficient quantum computing with post-selection. In relation to the first half of our question (can post-selection have a role to play?), these results provided strong evidence in the affirmative. In relation to the second half of our question (what is the nature of that role?), we provided several proposal applications. Most notably, we extrapolated from analytic and numerical models to provide resource estimates for medium-scale circuits, such as the 15-to-1 magic state distillation circuit, and reported that exclusive decoding resulted in a 75% reduction in the number of physical qubits required, and a 60% reduction in the total spacetime volume required. The reduction to qubit footprint is especially relevant to near-term medium-scale QEC demonstrations with limited components, and the reduction to spacetime volume will continue to be relevant in the future to large-scale general error-corrected

quantum computers. We remark that magic-state distillation is an architectural element, and understanding the role of post-selection is closely coupled to the broader question of how to lay out designs for large-scale quantum computing architectures, which is a fertile and largely unexplored area of research. We identify two future lines of research (i) to explore whether better use of the syndrome data, in combination with exclusive decoding, can open up new regimes of applicability, and (ii) can quantum computing architectures be designed to exploit this? We also point out related work published concurrently [124] also explored similar exclusive-style decoders, but focused on the orthogonal application of exclusive decoding for code-concatenation. We believe that the technique of exclusive decoding, especially including our specialized numerical sampling methods that we developed, will continue to be useful in progressing inquiry into uses of post-selection for quantum computing at scale.

In Chapter 4, we developed a methodology for generating many instances of stabilizer circuits, and for searching through these circuits filtering by their error-correcting properties. From one perspective, circuit design can be viewed as a compilation problem, whereby an abstract error-correcting code is compiled into a sequence of machine-level instructions that can be executed on hardware, sensitive to device-level considerations such as connectivity, available gate sets and component-level noise specifications. To this end, we showcased our methodology by generating syndrome extraction circuits for the 2-dimensional color code, on both the honeycomb lattice and on the 4.8.8 lattice. Our search yielded many instances of compact circuits that were optimized to have a small space-time overhead, and also to perform well against physically realistic near-term error rates, making them well-suited for use in current-day experimental demonstrations of fault-tolerant computation with the color code [29, 31, 35, 183]. An open direction left for future work is to benchmark our circuits, to further optimize for logical performance at all error rates.

An alternative perspective of circuit design, however, is that stabilizer circuits should be viewed as a central object in the theory of fault-tolerance, capable of describing a more general variety of fault-tolerant constructions. It is entirely possible that the architecture that will eventually be used in large-scale fault-tolerant quantum computers is yet to be discovered. In relation to the color code, our tools allowed a search through a larger space of compact syndrome extraction circuits than has previously been possible. We hope that our methodology will also be useful in the exploration of fault-tolerant stabilizer circuits to design better fault-tolerant architectures for quantum computing at scale.

Bibliography

[1] R. P. Feynman, International Journal of Theoretical Physics **21**, 467 (1982).

[2] P. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994) pp. 124–134.

[3] O. Regev, An efficient quantum factoring algorithm (2024), arXiv:2308.06572 [quant-ph] .

[4] R. Landauer, M. E. Welland, and J. K. Gimzewski, Philosophical Transactions of the Royal Society of London. Series A: Physical and Engineering Sciences **353**, 367 (1995), https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.1995.0106 .

[5] R. Landauer, Physics Letters A **217**, 188 (1996).

[6] S. C. Kak, Foundations of Physics **26**, 127 (1996).

[7] W. G. Unruh, Phys. Rev. A **51**, 992 (1995).

[8] D. Aharonov and M. Ben-Or, in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '97 (Association for Computing Machinery, New York, NY, USA, 1997) p. 176–188.

[9] P. W. Shor, Phys. Rev. A **52**, R2493 (1995).

[10] A. M. Steane, Phys. Rev. Lett. **77**, 793 (1996).

[11] A. R. Calderbank and P. W. Shor, Phys. Rev. A **54**, 1098 (1996).

[12] P. W. Shor, in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96 (IEEE Computer Society, USA, 1996) p. 56.

[13] J. Preskill, Quantum **2**, 79 (2018).

[14] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, Nature **574**, 505 (2019).

[15] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and A. Kandala, Nature **618**, 500 (2023).

[16] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, Nature Communications **5**, 4213 (2014).

[17] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth, and J. Tennyson, Physics Reports **986**, 1 (2022), the Variational Quantum Eigensolver: a review of methods and best practices.

[18] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm (2014), arXiv:1411.4028 .

[19] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, Physics Reports **1068**, 1 (2024), a review on Quantum Approximate Optimization Algorithm and its variants.

[20] Z. Cai, R. Babbush, S. C. Benjamin, S. Endo, W. J. Huggins, Y. Li, J. R. McClean, and T. E. O'Brien, Rev. Mod. Phys. **95**, 045005 (2023).

[21] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff, Nature **605**, 669 (2022).

[22] N. Sundaresan, T. J. Yoder, Y. Kim, M. Li, E. H. Chen, G. Harper, T. Thorbeck, A. W. Cross, A. D. Córcoles, and M. Takita, Nature Communications **14**, 2852 (2023).

[23] Y. Ye, T. He, H.-L. Huang, Z. Wei, Y. Zhang, Y. Zhao, D. Wu, Q. Zhu, H. Guan, S. Cao, F. Chen, T.-H. Chung, H. Deng, D. Fan, M. Gong, C. Guo, S. Guo, L. Han, N. Li, S. Li, Y. Li, F. Liang, J. Lin, H. Qian, H. Rong, H. Su, S. Wang, Y. Wu, Y. Xu, C. Ying, J. Yu, C. Zha, K. Zhang, Y.-H. Huo, C.-Y. Lu, C.-Z. Peng, X. Zhu, and J.-W. Pan, Phys. Rev. Lett. **131**, 210603 (2023).

[24] R. S. Gupta, N. Sundaresan, T. Alexander, C. J. Wood, S. T. Merkel, M. B. Healy, M. Hillenbrand, T. Jochym-O'Connor, J. R. Wootton, T. J. Yoder, A. W. Cross, M. Takita, and B. J. Brown, Nature **625**, 259 (2024).

[25] R. Harper and S. T. Flammia, Phys. Rev. Lett. **122**, 080504 (2019).

[26] E. H. Chen, T. J. Yoder, Y. Kim, N. Sundaresan, S. Srinivasan, M. Li, A. D. Córcoles, A. W. Cross, and M. Takita, Phys. Rev. Lett. **128**, 110504 (2022).

[27] B. Hetényi and J. R. Wootton, Creating entangled logical qubits in the heavy-hex lattice with topological codes (2024), arXiv:2404.15989 [quant-ph] .

[28] R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, J. Atalaya, R. Babbush, D. Bacon, J. C. Bardin, J. Basso, A. Bengtsson, S. Boixo, G. Bortoli, A. Bourassa, J. Bovaird, L. Brill, M. Broughton, B. B. Buckley, D. A. Buell, T. Burger, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, J. Cogan, R. Collins, P. Conner, W. Courtney, A. L. Crook, B. Curtin, D. M. Debroy, A. Del Toro Barba, S. Demura, A. Dunsworth, D. Eppens, C. Erickson, L. Faoro, E. Farhi, R. Fatemi, L. Flores Burgos, E. Forati, A. G. Fowler, B. Foxen, W. Giang, C. Gidney, D. Gilboa, M. Giustina, A. Grajales Dau, J. A. Gross, S. Habegger, M. C. Hamilton, M. P. Harrigan, S. D. Harrington, O. Higgott, J. Hilton, M. Hoffmann, S. Hong, T. Huang, A. Huff, W. J. Huggins, L. B. Ioffe, S. V. Isakov, J. Iveland, E. Jeffrey, Z. Jiang, C. Jones, P. Juhas, D. Kafri, K. Kechedzhi, J. Kelly, T. Khattar, M. Khezri, M. Kieferová, S. Kim, A. Kitaev, P. V. Klimov, A. R. Klots, A. N. Korotkov, F. Kostritsa, J. M. Kreikebaum, D. Landhuis, P. Laptev, K.-M. Lau, L. Laws, J. Lee, K. Lee, B. J. Lester, A. Lill,

W. Liu, A. Locharla, E. Lucero, F. D. Malone, J. Marshall, O. Martin, J. R. McClean, T. McCourt, M. McEwen, A. Megrant, B. Meurer Costa, X. Mi, K. C. Miao, M. Mohseni, S. Montazeri, A. Morvan, E. Mount, W. Mruczkiewicz, O. Naaman, M. Neeley, C. Neill, A. Nersisyan, H. Neven, M. Newman, J. H. Ng, A. Nguyen, M. Nguyen, M. Y. Niu, T. E. O'Brien, A. Opremcak, J. Platt, A. Petukhov, R. Potter, L. P. Pryadko, C. Quintana, P. Roushan, N. C. Rubin, N. Saei, D. Sank, K. Sankaragomathi, K. J. Satzinger, H. F. Schurkus, C. Schuster, M. J. Shearn, A. Shorter, V. Shvarts, J. Skruzny, V. Smelyanskiy, W. C. Smith, G. Sterling, D. Strain, M. Szalay, A. Torres, G. Vidal, B. Villalonga, C. Vollgraff Heidweiller, T. White, C. Xing, Z. J. Yao, P. Yeh, J. Yoo, G. Young, A. Zalcman, Y. Zhang, N. Zhu, and G. Q. AI, Nature **614**, 676 (2023).

[29] L. Postler, S. Heußen, I. Pogorelov, M. Rispler, T. Feldker, M. Meth, C. D. Marciniak, R. Stricker, M. Ringbauer, R. Blatt, P. Schindler, M. Müller, and T. Monz, Nature **605**, 675 (2022).

[30] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz, Phys. Rev. X **11**, 041058 (2021).

[31] C. Ryan-Anderson, N. C. Brown, M. S. Allman, B. Arkin, G. Asa-Attuah, C. Baldwin, J. Berg, J. G. Bohnet, S. Braxton, N. Burdick, J. P. Campora, A. Chernoguzov, J. Esposito, B. Evans, D. Francois, J. P. Gaebler, T. M. Gatterman, J. Gerber, K. Gilmore, D. Gresh, A. Hall, A. Hankin, J. Hostetter, D. Lucchetti, K. Mayer, J. Myers, B. Neyenhuis, J. Santiago, J. Sedlacek, T. Skripka, A. Slattery, R. P. Stutz, J. Tait, R. Tobey, G. Vittorini, J. Walker, and D. Hayes, Implementing fault-tolerant entangling gates on the five-qubit code and the color code (2022), arXiv:2208.01863 [quant-ph] .

[32] M. P. da Silva, C. Ryan-Anderson, J. M. Bello-Rivas, A. Chernoguzov, J. M. Dreiling, C. Foltz, F. Frachon, J. P. Gaebler, T. M. Gatterman, L. Grans-Samuelsson, D. Hayes, N. Hewitt, J. Johansen, D. Lucchetti, M. Mills, S. A. Moses, B. Neyenhuis, A. Paz, J. Pino, P. Siegfried, J. Strabley, A. Sundaram, D. Tom, S. J. Wernli, M. Zanner, R. P. Stutz, and K. M. Svore, Demonstration of logical qubits and repeated error correction with better-than-physical error rates (2024), arXiv:2404.02280 [quant-ph] .

[33] L. Egan, D. M. Debroy, C. Noel, A. Risinger, D. Zhu, D. Biswas, M. Newman, M. Li, K. R. Brown, M. Cetina, and C. Monroe, Fault-tolerant operation of a quantum error-correction code (2021), arXiv:2009.11482 [quant-ph] .

[34] M. H. Abobeih, Y. Wang, J. Randall, S. J. H. Loenen, C. E. Bradley, M. Markham, D. J. Twitchen, B. M. Terhal, and T. H. Taminiau, Nature **606**, 884 (2022).

[35] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, Nature **626**, 58 (2024).

[36] D. P. DiVincenzo, Fortschritte der Physik **48**, 771 (2000).

[37] R. Acharya, I. Aleiner, R. Allen, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, J. Atalaya, R. Babbush, D. Bacon, J. C. Bardin, J. Basso, A. Bengtsson, S. Boixo, G. Bortoli, A. Bourassa, J. Bovaird, L. Brill, M. Broughton, B. B. Buckley, D. A. Buell, T. Burger, B. Burkett, N. Bushnell, Y. Chen, Z. Chen, B. Chiaro, J. Cogan, R. Collins, P. Conner, W. Courtney, A. L. Crook, B. Curtin, D. M. Debroy, A. Del Toro Barba, S. Demura, A. Dunsworth, D. Eppens, C. Erickson, L. Faoro, E. Farhi,

R. Fatemi, L. Flores Burgos, E. Forati, A. G. Fowler, B. Foxen, W. Giang, C. Gidney, D. Gilboa, M. Giustina, A. Grajales Dau, J. A. Gross, S. Habegger, M. C. Hamilton, M. P. Harrigan, S. D. Harrington, O. Higgott, J. Hilton, M. Hoffmann, S. Hong, T. Huang, A. Huff, W. J. Huggins, L. B. Ioffe, S. V. Isakov, J. Iveland, E. Jeffrey, Z. Jiang, C. Jones, P. Juhas, D. Kafri, K. Kechedzhi, J. Kelly, T. Khattar, M. Khezri, M. Kieferová, S. Kim, A. Kitaev, P. V. Klimov, A. R. Klots, A. N. Korotkov, F. Kostritsa, J. M. Kreikebaum, D. Landhuis, P. Laptev, K.-M. Lau, L. Laws, J. Lee, K. Lee, B. J. Lester, A. Lill, W. Liu, A. Locharla, E. Lucero, F. D. Malone, J. Marshall, O. Martin, J. R. McClean, T. McCourt, M. McEwen, A. Megrant, B. Meurer Costa, X. Mi, K. C. Miao, M. Mohseni, S. Montazeri, A. Morvan, E. Mount, W. Mruczkiewicz, O. Naaman, M. Neeley, C. Neill, A. Nersisyan, H. Neven, M. Newman, J. H. Ng, A. Nguyen, M. Nguyen, M. Y. Niu, T. E. O'Brien, A. Opremcak, J. Platt, A. Petukhov, R. Potter, L. P. Pryadko, C. Quintana, P. Roushan, N. C. Rubin, N. Saei, D. Sank, K. Sankaragomathi, K. J. Satzinger, H. F. Schurkus, C. Schuster, M. J. Shearn, A. Shorter, V. Shvarts, J. Skruzny, V. Smelyanskiy, W. C. Smith, G. Sterling, D. Strain, M. Szalay, A. Torres, G. Vidal, B. Villalonga, C. Vollgraff Heidweiller, T. White, C. Xing, Z. J. Yao, P. Yeh, J. Yoo, G. Young, A. Zalcman, Y. Zhang, N. Zhu, and G. Q. AI, Nature **614**, 676 (2023).

[38] J. Haah, What is your logical qubit? (2024).

[39] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[40] J. Preskill, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**, 385 (1998), https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1998.0167 .

[41] A. Kitaev, Annals of Physics **303**, 2 (2003).

[42] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Journal of Mathematical Physics **43**, 4452 (2002), https://pubs.aip.org/aip/jmp/article-pdf/43/9/4452/19183135/4452_1_online.pdf .

[43] B. J. Brown, IEEE BITS the Information Theory Magazine **2**, 5 (2022).

[44] N. Delfosse and N. H. Nickerson, Quantum **5**, 595 (2021).

[45] B. M. Terhal, Rev. Mod. Phys. **87**, 307 (2015).

[46] D. Ristè, L. C. G. Govia, B. Donovan, S. D. Fallek, W. D. Kalfus, M. Brink, N. T. Bronn, and T. A. Ohki, npj Quantum Information **6**, 71 (2020).

[47] C. K. Andersen, A. Remm, S. Lazar, S. Krinner, N. Lacroix, G. J. Norris, M. Gabureac, C. Eichler, and A. Wallraff, Nature Physics **16**, 875 (2020).

[48] M. Herold, E. T. Campbell, J. Eisert, and M. J. Kastoryano, npj Quantum Information **1**, 15010 (2015).

[49] M. Herold, M. J. Kastoryano, E. T. Campbell, and J. Eisert, New Journal of Physics **19**, 063012 (2017).

[50] P. Gács, Journal of Statistical Physics **103**, 45 (2001).

[51] J. W. Harrington, *Analysis of quantum error-correcting codes: symplectic lattice codes and toric codes*, Ph.D. thesis, California Institute of Technology (2004).

[52] G. Dauphinais and D. Poulin, Communications in Mathematical Physics **355**, 519 (2017).

[53] J. F. S. Miguel, D. J. Williamson, and B. J. Brown, Quantum **7**, 940 (2023).

[54] A. Kubica and J. Preskill, Phys. Rev. Lett. **123**, 020501 (2019).

[55] M. Vasmer, D. E. Browne, and A. Kubica, Scientific Reports **11**, 2027 (2021).

[56] N. P. Breuckmann, K. Duivenvoorden, D. Michels, and B. M. Terhal, Quantum Info. Comput. **17**, 181–208 (2017).

[57] C. Wang, J. Harrington, and J. Preskill, Annals of Physics **303**, 31 (2003).

[58] H. Anwar, B. J. Brown, E. T. Campbell, and D. E. Browne, New Journal of Physics **16**, 063038 (2014).

[59] A. G. Fowler, Quantum Info. Comput. **15**, 145–158 (2015).

[60] J. Wootton, Entropy **17**, 1946 (2015).

[61] S. Bravyi and J. Haah, Phys. Rev. Lett. **111**, 200501 (2013).

[62] A. Hutter, D. Loss, and J. R. Wootton, New Journal of Physics **17**, 035017 (2015).

[63] K. Hammar, A. Orekhov, P. W. Hybelius, A. K. Wisakanto, B. Srivastava, A. F. Kockum, and M. Granath, Phys. Rev. A **105**, 042616 (2022).

[64] Y.-H. Liu and D. Poulin, Phys. Rev. Lett. **122**, 200501 (2019).

[65] G. Torlai and R. G. Melko, Phys. Rev. Lett. **119**, 030501 (2017).

[66] S. Varsamopoulos, B. Criger, and K. Bertels, Quantum Science and Technology **3**, 015004 (2017).

[67] A. S. Darmawan and D. Poulin, Phys. Rev. E **97**, 051302 (2018).

[68] G. Duclos-Cianci and D. Poulin, Phys. Rev. Lett. **104**, 050504 (2010).

[69] J. R. Wootton and D. Loss, Phys. Rev. Lett. **109**, 160503 (2012).

[70] S. Bravyi, M. Suchara, and A. Vargo, Phys. Rev. A **90**, 032326 (2014).

[71] N. Delfosse, Phys. Rev. A **89**, 012317 (2014).

[72] A. Kubica and N. Delfosse, Quantum **7**, 929 (2023).

[73] K. Sahay and B. J. Brown, PRX Quantum **3**, 010310 (2022).

[74] S. S. Tannu, Z. A. Myers, P. J. Nair, D. M. Carmean, and M. K. Qureshi, in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17 (Association for Computing Machinery, New York, NY, USA, 2017) p. 679–691.

[75] P. Das, C. A. Pattison, S. Manne, D. M. Carmean, K. M. Svore, M. Qureshi, and N. Delfosse, in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (2022) pp. 259–273.

[76] C. Chamberland, L. Goncalves, P. Sivarajah, E. Peterson, and S. Grimberg, Quantum Science and Technology **8**, 045011 (2023).

[77] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, Neo-qec: Neural network enhanced online superconducting decoder for surface codes (2022), arXiv:2208.05758 [quant-ph] .

[78] S. Gicev, L. C. L. Hollenberg, and M. Usman, Quantum **7**, 1058 (2023).

[79] K. Meinerz, C.-Y. Park, and S. Trebst, Phys. Rev. Lett. **128**, 080505 (2022).

[80] N. Delfosse, Hierarchical decoding to reduce hardware requirements for quantum computing (2020), arXiv:2001.11427 [quant-ph] .

[81] G. S. Ravi, J. M. Baker, A. Fayyazi, S. F. Lin, A. Javadi-Abhari, M. Pedram, and F. T. Chong, in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS 2023 (Association for Computing Machinery, New York, NY, USA, 2023) p. 88–102.

[82] P. Das, A. Locharla, and C. Jones, in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22 (Association for Computing Machinery, New York, NY, USA, 2022) p. 541–553.

[83] A. Paler and A. G. Fowler, Quantum **7**, 1205 (2023).

[84] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, in *2021 58th ACM/IEEE Design Automation Conference (DAC)* (2021) pp. 451–456.

[85] A. Holmes, M. R. Jokar, G. Pasandi, Y. Ding, M. Pedram, and F. T. Chong, in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)* (2020) pp. 556–569.

[86] Y. Ueno, M. Kondo, M. Tanaka, Y. Suzuki, and Y. Tabuchi, in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (2022) pp. 274–287.

[87] M. E. Beverland, B. J. Brown, M. J. Kastoryano, and Q. Marolleau, Journal of Statistical Mechanics: Theory and Experiment **2019**, 073404 (2019).

[88] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, Phys. Rev. Lett. **108**, 180501 (2012).

[89] J. Edmonds, Journal of research of the National Bureau of Standards B **69**, 55 (1965).

[90] J. Edmonds, Canadian Journal of Mathematics **17**, 449–467 (1965).

[91] V. Kolmogorov, Mathematical Programming Computation **1**, 43 (2009).

[92] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).

[93] D. K. Tuckett, *Tailoring surface codes: Improvements in quantum error correction with biased noise*, Ph.D. thesis, University of Sydney (2020), (qecsim: https://github.com/qecsim/qecsim).

[94] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, Nature Methods **17**, 261 (2020).

[95] S. Bravyi and A. Vargo, Phys. Rev. A **88**, 062308 (2013).

[96] C. H. Bennett, Journal of Computational Physics **22**, 245 (1976).

[97] O. Higgott, ACM Transactions on Quantum Computing **3**, 16 (2022).

[98] A. G. Fowler, D. S. Wang, and L. C. L. Hollenberg, Quant. Info. Comput. **11**, 0008 (2011).

[99] D. K. Tuckett, S. D. Bartlett, and S. T. Flammia, Phys. Rev. Lett. **120**, 050505 (2018).

[100] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, Nature Communications **12**, 2172 (2021).

[101] A. S. Darmawan, B. J. Brown, A. L. Grimsmo, D. K. Tuckett, and S. Puri, PRX Quantum **2**, 030345 (2021).

[102] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, Phys. Rev. X **10**, 011022 (2020).

[103] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, New Journal of Physics **22**, 023019 (2020).

[104] E. Knill, Nature **434**, 39 (2005).

[105] Y. Li, New Journal of Physics **17**, 023037 (2015).

[106] C. Chamberland and A. W. Cross, Quantum **3**, 143 (2019).

[107] C. Gidney and M. Ekerå, Quantum **5**, 433 (2021).

[108] S. Singh, A. S. Darmawan, B. J. Brown, and S. Puri, Phys. Rev. A **105**, 052410 (2022).

[109] H. Bombín, M. Pant, S. Roberts, and K. I. Seetharam, PRX Quantum **5**, 010302 (2024).

[110] S. T. Spitz, B. Tarasinski, C. W. J. Beenakker, and T. E. O'Brien, Advanced Quantum Technologies **1**, 1800012 (2018).

[111] T. Wagner, H. Kampermann, D. Bruß, and M. Kliesch, Quantum **6**, 809 (2022).

[112] C. Gidney, M. Newman, P. Brooks, and C. Jones, Yoked surface codes (2023), arXiv:2312.04522 [quant-ph] .

[113] D. Bruß, D. P. DiVincenzo, A. Ekert, C. A. Fuchs, C. Macchiavello, and J. A. Smolin, Phys. Rev. A **57**, 2368 (1998).

[114] N. J. Cerf, Physical Review Letters **84**, 4497 (2000).

[115] T. S. Cubitt, M. B. Ruskai, and G. Smith, Journal of Mathematical Physics **49**, 102104 (2008).

[116] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, Nature Communications **14**, 7040 (2023).

[117] X. Tan, F. Zhang, R. Chao, Y. Shi, and J. Chen, PRX Quantum **4**, 040344 (2023).

[118] S. Bravyi and A. Kitaev, Phys. Rev. A **71**, 022316 (2005).

[119] D. Litinski, Quantum **3**, 205 (2019).

[120] J. Son, M. Gluza, R. Takagi, and N. H. Y. Ng, Quantum dynamic programming (2024), arXiv:2403.09187 [quant-ph] .

[121] W. J. Huggins and J. R. McClean, Quantum **8**, 1264 (2024).

[122] I. Marvian and S. Lloyd, Universal quantum emulator (2024), arXiv:1606.02734 [quant-ph] .

[123] C. A. Pattison, A. Krishna, and J. Preskill, Hierarchical memories: Simulating quantum LDPC codes with local gates (2023), arXiv:2303.04798 [quant-ph] .

[124] N. Meister, C. A. Pattison, and J. Preskill, Efficient soft-output decoders for the surface code (2024), arXiv:2405.07433 .

[125] H. Levine, A. Haim, J. Hung, N. Alidoust, M. Kalaee, L. DeLorenzo, E. Wollack, P. Arrangoiz-Arriola, A. Khalajhedayati, R. Sanil, H. Moradinejad, Y. Vaknin, A. Kubica, D. Hover, S. Aghaeimeibodi, J. Alcid, C. Baek, J. Barnett, K. Bawdekar, P. Bienias, H. Carson, C. Chen, L. Chen, H. Chinkezian, E. Chisholm, A. Clifford, R. Cosmic, N. Crisosto, A. Dalzell, E. Davis, J. D'Ewart, S. Diez, N. D'Souza, P. Dumitrescu, E. Elkhouly, M. Fang, Y. Fang, S. Flammia, M. Fling, G. Garcia, M. Gharzai, A. Gorshkov, M. Gray, S. Grimberg, A. Grimsmo, C. Hann, Y. He, S. Heidel, S. Howell, M. Hunt, J. Iverson, I. Jarrige, L. Jiang, W. Jones, R. Karabalin, P. Karalekas, A. Keller, D. Lasi, M. Lee, V. Ly, G. MacCabe, N. Mahuli, G. Marcaud, M. Matheny, S. McArdle, G. McCabe, G. Merton, C. Miles, A. Milsted, A. Mishra, L. Moncelsi, M. Naghiloo, K. Noh, E. Oblepias, G. Ortuno, J. Owens, J. Pagdilao, A. Panduro, J.-P. Paquette, R. Patel, G. Peairs, D. Perello, E. Peterson, S. Ponte, H. Putterman, G. Refael, P. Reinhold, R. Resnick, O. Reyna, R. Rodriguez, J. Rose, A. Rubin, M. Runyan, C. Ryan, A. Sahmoud, T. Scaffidi, B. Shah, S. Siavoshi, P. Sivarajah, T. Skogland, C.-J. Su, L. Swenson, J. Sylvia, S. Teo, A. Tomada, G. Torlai, M. Wistrom, K. Zhang, I. Zuk, A. Clerk, F. Brandão, A. Retzker, and O. Painter, Physical Review X **14**, 011051 (2024).

[126] A. Kubica, A. Haim, Y. Vaknin, H. Levine, F. Brandão, and A. Retzker, Phys. Rev. X **13**, 041022 (2023).

[127] S. Gu, A. Retzker, and A. Kubica, Fault-tolerant quantum architectures based on erasure qubits (2023), arXiv:2312.14060 [quant-ph] .

[128] Z. Li, I. Kim, and P. Hayden, Quantum **7**, 1089 (2023).

[129] C. H. Bennett, D. P. DiVincenzo, and J. A. Smolin, Physical Review Letters **78**, 3217 (1997).

[130] N. Delfosse and G. Zémor, Physical Review Research **2**, 033042 (2020).

[131] N. Delfosse, V. Londe, and M. E. Beverland, IEEE Transactions on Information Theory **68**, 3187 (2022).

[132] N. Connolly, V. Londe, A. Leverrier, and N. Delfosse, Fast erasure decoder for a class of quantum LDPC codes (2023), arXiv:2208.01002 [quant-ph] .

[133] E. Arikan, IEEE Transactions on Information Theory **55**, 3051 (2009).

[134] S. Kudekar, T. J. Richardson, and R. L. Urbanke, IEEE Transactions on Information Theory **57**, 803 (2011).

[135] S. Kudekar, S. Kumar, M. Mondelli, H. D. Pfister, E. Şaşoğlu, and R. Urbanke, in *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '16 (Association for Computing Machinery, New York, NY, USA, 2016) p. 658–669.

[136] S. C. Smith, B. J. Brown, and S. D. Bartlett, Physical Review Applied **19**, 034050 (2023).

[137] F. Battistel, C. Chamberland, K. Johar, R. W. J. Overwater, F. Sebastiano, L. Skoric, Y. Ueno, and M. Usman, Nano Futures **7**, 032003 (2023).

[138] A. Holevo, Quantum Electronics **50**, 440 (2020).

[139] L. Gyongyosi, S. Imre, and H. V. Nguyen, IEEE Communications Surveys & Tutorials **20**, 1149 (2018).

[140] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, Physical Review A **54**, 3824 (1996).

[141] A. W. Leung, Physical Review A **77**, 012322 (2008).

[142] H. Aschauer, *Quantum communication in noisy environments*, Ph.D. thesis, Ludwig-Maximilians-Universität München (2005).

[143] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, Physical Review Letters **76**, 722 (1996).

[144] M. Christandl and A. Müller-Hermes, IEEE Transactions on Information Theory **70**, 282 (2024).

[145] P. Belzig, M. Christandl, and A. Müller-Hermes, IEEE Transactions on Information Theory **70**, 2655 (2024).

[146] A. Hutter, J. R. Wootton, and D. Loss, Phys. Rev. A **89**, 022326 (2014).

[147] N. Delfosse and A. Paetznick, Spacetime codes of clifford circuits (2023), arXiv:2304.05943 .

[148] D. Gottesman, Opportunities and challenges in fault-tolerant quantum computation (2022), arXiv:2210.15844 .

[149] D. Bacon, S. T. Flammia, A. W. Harrow, and J. Shi, IEEE Transactions on Information Theory **63**, 2464 (2017).

[150] M. McEwen, D. Bacon, and C. Gidney, Quantum **7**, 1172 (2023).

[151] M. E. Beverland, S. Huang, and V. Kliuchnikov, Fault tolerance of stabilizer channels (2024), arXiv:2401.12017 .

[152] M. B. Hastings and J. Haah, Quantum **5**, 564 (2021).

[153] A. Keedwell and J. Dénes, *Latin Squares and Their Applications: Latin Squares and Their Applications* (Elsevier Science, 2015).

[154] H. Bombin and M. A. Martin-Delgado, Phys. Rev. Lett. **97**, 180501 (2006).

[155] A. Kubica, B. Yoshida, and F. Pastawski, New Journal of Physics **17**, 083026 (2015).

[156] M. S. Kesselring, J. C. Magdalena de la Fuente, F. Thomsen, J. Eisert, S. D. Bartlett, and B. J. Brown, PRX Quantum **5**, 010342 (2024).

[157] F. Thomsen, M. S. Kesselring, S. D. Bartlett, and B. J. Brown, Low-overhead quantum computing with the color code (2024), arXiv:2201.07806 .

[158] A. J. Landahl and C. Ryan-Anderson, Quantum computing by color-code lattice surgery (2014), arXiv:1407.5103 [quant-ph] .

[159] M. S. Kesselring, F. Pastawski, J. Eisert, and B. J. Brown, Quantum **2**, 101 (2018).

[160] A. G. Fowler, Physical Review A **83**, 042310 (2011).

[161] A. M. Stephens, Efficient fault-tolerant decoding of topological color codes (2014), arXiv:1402.3037 [quant-ph] .

[162] P. Baireuther, M. D. Caio, B. Criger, C. W. J. Beenakker, and T. E. O'Brien, New Journal of Physics **21**, 013003 (2019).

[163] A. Kubica and N. Delfosse, Quantum **7**, 929 (2023).

[164] S.-H. Lee, A. Li, and S. D. Bartlett, Color code decoder with improved scaling for correcting circuit-level noise (2024), arXiv:2404.07482 [quant-ph] .

[165] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes (2011), arXiv:1108.5738 [quant-ph] .

[166] M. E. Beverland, A. Kubica, and K. M. Svore, PRX Quantum **2**, 020341 (2021).

[167] C. Gidney and C. Jones, New circuits and an open source decoder for the color code (2023), arXiv:2312.08813 .

[168] G. P. Gehér, O. Crawford, and E. T. Campbell, PRX Quantum **5**, 010348 (2024).

[169] K. Tiurev, A. Pesah, P.-J. H. S. Derks, J. Roffe, J. Eisert, M. S. Kesselring, and J.-M. Reiner, The domain wall color code (2024), arXiv:2307.00054 .

[170] J. Zhang, Y.-C. Wu, and G.-P. Guo, Facilitating practical fault-tolerant quantum computing based on color codes (2024), arXiv:2309.05222 .

[171] B. Pato, T. Tansuwannont, S. Huang, and K. R. Brown, PRX Quantum **5**, 020336 (2024).

[172] C. Chamberland and M. E. Beverland, Quantum **2**, 53 (2018).

[173] T. J. Yoder and I. H. Kim, Quantum **1**, 2 (2017).

[174] T. Tansuwannont, C. Chamberland, and D. Leung, Physical Review A **101**, 012342 (2020).

[175] Y. Tomita and K. M. Svore, Phys. Rev. A **90**, 062320 (2014).

[176] J. yu Shao and W. di Wei, Discrete Mathematics **110**, 293 (1992).

[177] OEIS Foundation Inc., The On-Line Encyclopedia of Integer Sequences (2024), published electronically at http://oeis.org.

[178] B. McKay, Latin squares (2024), published electronically at https://users.cecs.anu.edu.au/~bdm/data/latin.html.

[179] M. T. Jacobson and P. Matthews, Journal of Combinatorial Designs **4**, 405 (1996).

[180] D. Holt, B. Eick, and E. O'Brien, *Handbook of Computational Group Theory*, Discrete Mathematics and Its Applications (CRC Press, 2005).

[181] N. Berthusen, D. Devulapalli, E. Schoute, A. M. Childs, M. J. Gullans, A. V. Gorshkov, and D. Gottesman, Toward a 2d local implementation of quantum ldpc codes (2024), arXiv:2404.17676 [quant-ph] .

[182] X. Fu, M. A. Rol, C. C. Bultink, J. van Someren, N. Khammassi, I. Ashraf, R. F. L. Vermeulen, J. C. de Sterke, W. J. Vlothuizen, R. N. Schouten, C. G. Almudever, L. DiCarlo, and K. Bertels, in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17 (Association for Computing Machinery, New York, NY, USA, 2017) p. 813–825.

[183] L. Postler, F. Butt, I. Pogorelov, C. D. Marciniak, S. Heußen, R. Blatt, P. Schindler, M. Rispler, M. Müller, and T. Monz, Demonstration of fault-tolerant steane quantum error correction (2023), arXiv:2312.09745 [quant-ph] .