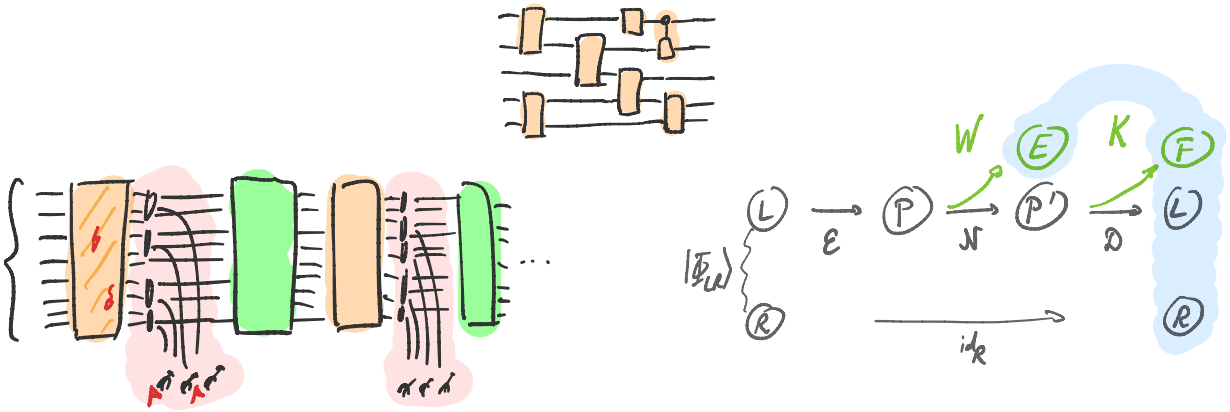
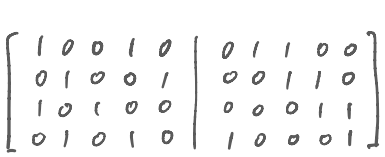
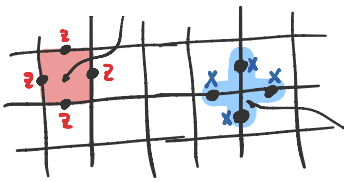






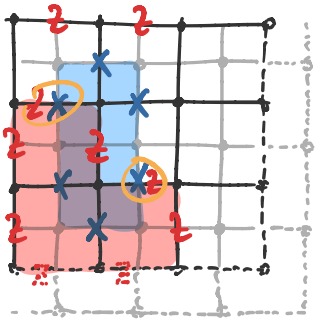
Quantum Error Correction

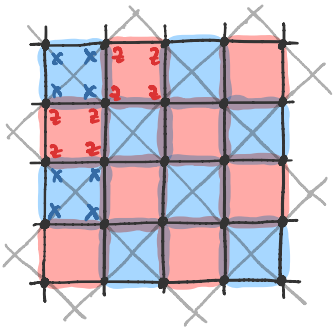
LECTURE NOTES

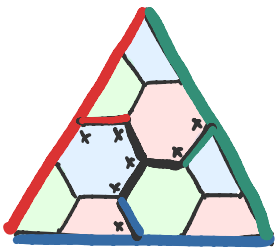
BY

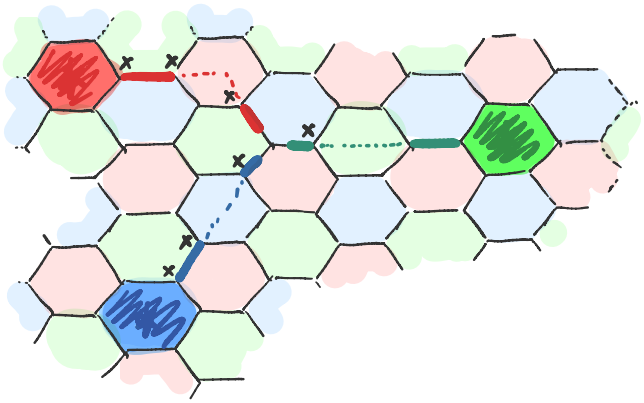
Philippe Faist

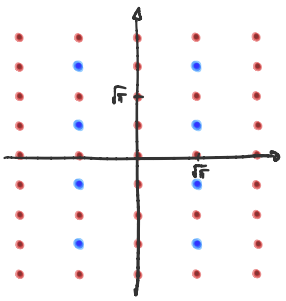
Dec 11, 2024











Contents

Preface

Lecture notes based on a lecture given at the Freie Universität Berlin in 2024.

The goal of these lecture notes is to bring a reader with elementary background in quantum information theory to understand both the main concepts of quantum error correction as well as how a fully fault-tolerant computation might be implemented on quantum hardware in the future.

What we'll cover here

The bulk of the course begins with the fundamental theory behind quantum error correction. If an environment corrupts quantum information stored on a quantum system, we'll understand the quantum-information-theoretic principles that dictate when the information can be recovered. We'll then focus on a powerful framework to construct and analyze quantum error-correcting codes on n qubits, known as the *stabilizer formalism*. We'll then study a serious candidate for building reliable quantum computers: the surface code. We'll then cover more carefully some important concepts that are well illustrated by the surface code, including the general idea of topological codes as well as the relation between stabilizer codes and the mathematical field of *homology*.

Moving beyond the bare construction of interesting codes, we'll study their ability to protect information in a more realistic setting where the exposure to noise is persistent and can also affect the operations involved in the error-correcting procedure itself. Specifically, we'll understand the idea of a *code threshold*; we'll prove that the surface code has a threshold, and draw some deep connections between decoding the surface code and phases of matter of statistical models in condensed matter physics.

Beyond the storage of information, we'd like to run a quantum computation on encoded states. We'll cover the main important topics of fault-tolerant quantum computation, including transversal gates and implementing gates with magic states. Putting together these elements, we'll overview a proposal for fault-tolerant quantum computation based on the surface code.

An additional chapter presents an additional topic that is also central to the field of quantum error correction, but is not immediately necessary in the above story line. A lecture on quantum error correction wouldn't feel complete if it failed to include an introduction to *bosonic codes*, which exploit the structure of continuous-variable quantum systems to offer very different schemes for protecting information against noise.

I initially had the ambition to go deeper in some more advanced topics of qubit codes and introduce subsystem codes, modern decoding techniques, and some recent constructions that are at the core of some high-performance qubit codes known as *qLDPC* codes. I didn't have the chance to include these topics, and I'll most likely add them in a later version of these notes.

Other topics that are not included in these notes, and might make their way into them in the future, include a formal framework for proving the *threshold theorem*, constructions of *codes for qudits*, *holographic codes* and their application to quantum gravity, as well as deeper connections between topological error correction and condensed matter physics. See Recommended literature for more exhaustive references that cover these topics.

Recommended literature

The field of quantum error correction is too vast to cover in a single course; I'll focus on a selection of topics that I feel most significantly contribute to the story line of how to run a quantum error-corrected, fault-tolerant computation on a quantum computer. Several other resources provide deeper information about quantum error correction and fault tolerance or one of its more specific sub-topics. I'll select a few important resources here.

First, Daniel Gottesman has recently made public a draft of his upcoming book *Surviving as a quantum computer in a classical world* [1]. Gottesman's book is an in-depth textbook that covers a broad range of topics in quantum error correction, with a focus on qubit stabilizer codes. I'm going to try to stick to a large extent to Gottesman's definitions and conventions; I highly recommend readers to look up additional details and proofs in this book. Gottesman's book is, in any case, a great reference for readers who would like to dig further into the subject.

Second, the pedagogical aspect of these lecture notes can be complemented by looking up information in the Error Correction Zoo [2], which collects many details about error-correcting codes with a survey of the literature in the area that is as comprehensive as possible. The Error Correction Zoo is a project founded and led by Victor V Albert and which I've co-founded with Victor. We point readers to this resource in particular if they would like to look up properties of specific codes, to find papers related to a specific code, or to look up lists of codes that have a certain property.[1]

Other precious references have influenced the writing of these notes and are warmly recommended to any interested reader; some are mentioned below. Furthermore, each chapter in these notes also contains a section "Further reading" in which I collect a few selected pointers for interested readers who would like to start delving into the literature about related topics. The references provided here and in each "Further reading" section are suggestions of mine which typically include the references that most influenced the writing of these notes. They are not meant to be comprehensive.

- John Preskill's lecture notes [3] on quantum computation are a wonderful resource to learn about quantum error correction. They cover the important concepts in quantum error correction while providing many code constructions and insightful explanations. The lecture notes are available here;[2] see especially Chapter 7 "Quantum Error Correction" as well as handwritten notes on fault tolerance.[3]

[1] See https://errorcorrectionzoo.org/lists

[2] http://theory.caltech.edu/~preskill/ph219/

[3] http://theory.caltech.edu/~preskill/ph219/fault-tolerance-2011.pdf

- Nielsen and Chuang's great textbook [4] on quantum computation and quantum information contains a useful explanation of the important concepts of quantum error correction as well as of the stabilizer formalism.
- Several review or tutorial papers cover a broad range of topics in quantum error correction, including by Keisuke Fujii [5], by Daniel Gottesman [6], and by Joschka Roffe [7]. Several Ph.D. theses have great introductions to multiple topics in quantum error correction, such as Daniel Gottesman's [8], Nikolas Breuckmann's [9], Aleksander Kubica's [10], and Tomas Jochym-O'Connor's [11].
- Helpful lecture notes are available by Steven Girvin [12], Dan Browne [13], Kishor Bharti,[4] Andrew Cleland [14], as well as Ulysse Chabaud and Francesco Arzani.[5]
- The book on *Quantum Error Correction* edited by Lidar and Brun appeared in 2013 and contains several chapters by significant researchers in the quantum error correction community.
- Recorded lectures and tutorials at Boulder Summer Schools include presentations by Victor V Albert, Liang Jiang, and Aleksander Kubica.[6] Recorded presentations at QIP conferences include tutorials by Naomi Nickerson[7] and Barbara Terhal.[8].
- Recordings of the course taught by Beni Yoshida and Daniel Gottesman in 2018 are available on the PIRSA platform.[9] Recordings from the course taught by Gottesman in 2007 are also available.[10]
- A recorded lecture series by Joe Renes at ETH Zurich covers many of the topics included in these notes.[11]
- A collection of further useful are listed in a Twitter thread.[12]
- A Discord server also serves to connect the error correction community (simply ask Victor for an invite; I'm avoiding to post the invite link publicly to avoid spam).

Acknowledgments

These notes have enormously benefited from discussions with many people from the quantum error correction and the broader quantum information communities. I thank Victor V Albert for his leadership in the zoo and for many discussions. I thank also group members in Berlin for many discussions and for feedback on parts of these notes, including Daniel Miller, Julio Magdalena de la Fuente, Alex Townsend-Teague, Peter-Jan Derks, Jonathan Conrad, Alexander Jahn, and Jens Eisert. I also want to thank the students of my 2024 QEC course for their enthusiasm and for useful feedback. Finally, I am profoundly indebted to Evita and Zoé for their love, care, and joy.

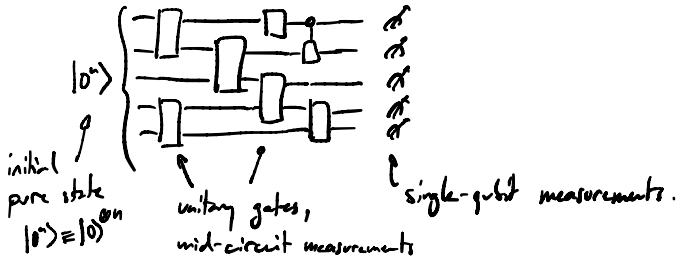
[4] https://sites.google.com/view/contextual-stories/teaching/quantum-error-correction-part-1
[5] https://ulyssechabaud.github.io/QC_course/
[6] https://boulderschool.yale.edu/2023/boulder-school-2023-lecture-notes
[7] https://youtu.be/2v-J95GFSGc
[8] https://youtu.be/Je7sVJGKMgU
[9] https://pirsa.org/c17045
[10] https://pirsa.org/c07001
[11] https://video.ethz.ch/lectures/d-phys/2022/spring/402-0460-00L.html
[12] https://twitter.com/theeczoo/status/1742915540530991146

Chapter 1

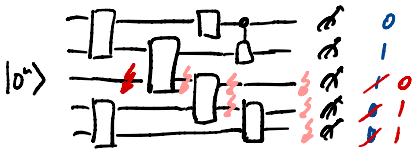
Introduction

Quantum computing operates by controlling individual quantum systems, enabling us to initialize them in some suitable state, engineer their evolution so as to perform a computation, and read out a measurement result from the final state.

In the usual paradigm of quantum computation, we operate on a collection of *qubits*. They can be initialized in a standard state, say $|0\rangle$; we can apply 2-qubit gates on certain pairs of qubits, depending on their hardware connectivity; we can measure individual qubits, collapsing them onto a definite computational basis state:



However, errors happen because the qubits are not perfectly isolated systems. Errors can happen at any stage of the computation, due to noise, to a calibration failure, or one of the many other error sources. Errors can generally corrupt the computation and lead to an incorrect result:



There is a tension between the requirement that good qubits be very well isolated from their environment to avoid errors, and the requirement that the qubits couple strongly to the systems we use to control them (laser pulses, microwave signals, magnetic fields, ...) to drive the computation. This tension leads to errors being inevitable in practice.

1.1 Common types of errors

- *decoherence* or *dephasing noise*: A superposition of quantum states evolves into a mixture, and off-diagonal coefficients of the density matrix written in the energy

eigenbasis decay. Such noise happens if the environment weakly measures the energy of the system;

- *depolarizing noise*: qubits become more random and lose the information stored in them. Their state is gradually replaced by a maximally mixed state. Such noise occurs if the qubit is coupled to a high-temperature environment;
- *photon loss* or *erasure errors*: a subsystem (e.g., a photon) is lost, along with any information it was carrying;
- *amplitude damping* or *relaxation noise* or *spontaneous emission noise*: a subsystem decays to its ground state, inducing the loss of the information it was carrying;
- *displacements* in continuous-variable systems: small shifts in phase space of a bosonic mode;
- discretized qubit errors: *bit-flip (X)* and *phase-flip (Z)* errors. These errors are convenient for the analysis of qubit error-correcting codes. We'll see how we can turn the native hardware error noise dynamics of a system, which is often one of the types of errors above, into discrete X and Z errors.
- ...

1.2 Strategies to counter errors

1. **Do nothing and hope for the best.** We can only run short circuits before our computation is overwhelmed by the noise. This regime is called the *Noisy Intermediate-Scale Quantum (NISQ) regime* [15].

   There was some initial optimism that an advantage over classical computers might already be obtained by operating quantum computers in this regime. As our understanding of this regime progresses, however, it appears less and less likely that such an advantage can materialize. We have not seen any conclusive evidence that running circuits in the NISQ regime can offer any significant practical advantages for useful problems.

2. **Play classical tricks to reduce the effect of the noise.** In particular, *quantum error mitigation* refers to a collection of schemes in which one first characterizes the noise precisely and then uses multiple runs of the circuit to infer, often through some type of extrapolation, the circuit's result without the noise. For instance, we might run computationally equivalent circuits of varying depth, and thus with varying levels of noise, to estimate the expectation value of an observable on the output state for the different levels of noise. Extrapolating this value to zero noise provides an estimate for the output of the noiseless circuit.

   Quantum error mitigation is indeed used in practice to push the limits of how large of a circuit we can run on noisy hardware. However, quantum error mitigation requires a number of runs of the circuit that grows exponentially both in the number of qubits as well as in the circuit depth. This high sampling overhead means the technique is not a scalable solution to remove errors in a large-scale quantum computation.

3. **Avoid the noise by using *decoherence-free subspaces*.** Suppose there is a subspace of the quantum system's Hilbert space that is not acted upon by the noise. We could encode our precious quantum information in that subspace, out of reach of the noise.

However, the noise typically acts in the full state space, meaning that there is no decoherence-free subspace available to store our information. Furthermore, even if the noise acts very weakly in that subspace, there is no mechanism to remove the effect of that noise; errors will accumulate over time, eventually corrupting the computation.

4. **Remove undesirable Hamiltonian terms with dynamical decoupling.** Sometimes, there is a term in the system's Hamiltonian that couples the qubit with an environmental degree of freedom (e.g., a background magnetic field), which we would like to suppress. A set of techniques called *dynamical decoupling*, consist in applying sequences of pulses to the system that are designed to suppress the effect of the undesired coupling. A *spin-echo* scheme, for instance, reduces the effect on a spin qubit of a slowly varying background magnetic field: We flip the qubit state at regular intervals in order to undo the drift caused by the background field. More sophisticated schemes can achieve better accuracy and can be combined with computational gates.

   These techniques are powerful and are actively used in modern quantum hardware. But they can only suppress coherent errors, i.e., an evolution generated by a Hamiltonian term. And again, these techniques are generally unable to remove any residual nonunitary noise that might have found its way into the computational degrees of freedom.

5. **Actively detect the presence of errors and perform corrective operations to remove them: *quantum error correction.*** With quantum error correction, parts of the circuit that we run on the quantum device serve solely to detect and correct errors as they happen, rather than to directly apply logical gates. Suppose we would like to run a logical quantum circuit on k qubits. Techniques from quantum error correction transform this circuit into a larger protocol involving n qubits and including multiple rounds of gates applied to perform logical computation, measurements to detect the presence of errors, running an algorithm on a classical computer to compute any corrections that need to be applied to the qubits (running a *decoder*), and applying any necessary corrections (Fig. 1.1).

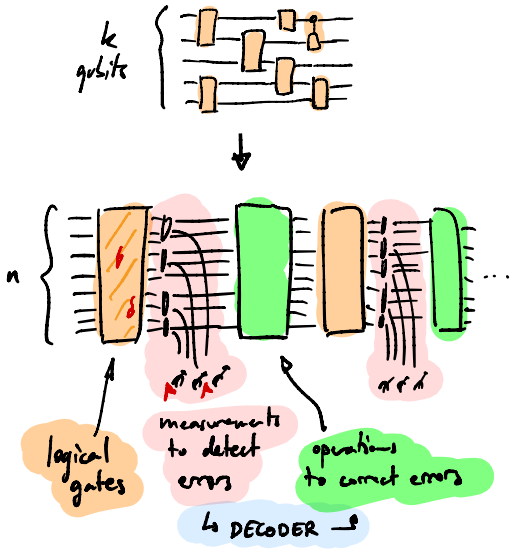


Figure 1.1: Standard paradigm of quantum error correction

A major challenge in implementing fault-tolerant quantum computation with quantum

error correction is the large overheads, both in number of qubits and in computation time, that are often required in schemes that we have found so far. Reducing these overheads is a central topic of current research.

6. **Engineer a system with dissipative dynamics that are specially designed to remove errors: *autonomous quantum error correction* or *self-correcting codes*.** While some promising results exist, few such schemes are known. Such schemes seem to require a high connectivity of the qubits (e.g., a four-dimensional lattice) or some other nonpractical features. Looks hard!

1.3 Designing a quantum error-correcting code

Classical information can be protected by simply repeating it (***classical repetition code***):

$$0 \to 00000; \qquad 1 \to 11111 \ . \tag{1.1}$$

Any errors that corrupt the encoded bits can easily be inferred, as long as the noise is weak enough such that the chance of corrupting more than half of the bits is negligible. For instance, the string 00010 can be identified as a corrupted version of 00000, encoding the logical bit value 0.

By analogy, an appealing idea would be to encode a quantum state $|\psi\rangle$ by repeating it multiple times:

$$|\psi\rangle \to |\psi\rangle \otimes |\psi\rangle \otimes |\psi\rangle \ . \tag{1.2}$$

Unfortunately, such an encoding is **forbidden by the no-cloning theorem.** Recall, the map $\rho \to \rho \otimes \rho$ is not linear in ρ and is therefore not a valid quantum operation.

Another idea, exploiting our intuition from the classical repetition code, would be to encode a qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ as

$$|\psi_{\text{code}}\rangle = \alpha|00\ldots0\rangle + \beta|11\ldots1\rangle \ . \tag{1.3}$$

This is the ***quantum repetition code*** EC Zoo.

Simple exercise: Find an encoding circuit for the quantum repetition code.

Suppose that one of the qubits of the quantum repetition code gets flipped: $|0\rangle \to |1\rangle$, $|1\rangle \to |0\rangle$. This error is called a ***bit-flip error***, or X ***error***, as it results from the application of a Pauli-X matrix $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ on the affected qubit. If, for instance, the second qubit undergoes an X error, the state evolves to

$$|\psi_{\text{qrep}}^{X_2}\rangle = \alpha|010\ldots0\rangle + \beta|101\ldots1\rangle \ . \tag{1.4}$$

We can detect the presence of such an error *without destroying the superposition* by measuring the following 2-body operator on the first and second qubits:

$$Z \otimes Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & -1 & & \\ & & -1 & \\ & & & 1 \end{pmatrix} \ . \tag{1.5}$$

The operator $Z \otimes Z$ has two eigenvalues, ± 1. We obtain the measurement outcome $+1$ if the two qubits are projected into the subspace spanned by $|00\rangle, |11\rangle$ and -1 if the state is projected into the subspace spanned by $|01\rangle, |10\rangle$. Measuring $Z \otimes Z$ on the two first qubits therefore tells us if the two qubits are in the same computational basis state (outcome $+1$) or not (outcome -1), without revealing the values of the qubits.

Our $|\psi_{\text{qrep}}\rangle$ is an eigenvector of $Z_1 \otimes Z_2 \equiv Z \otimes Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes \mathbb{1}$, associated with the eigenvalue $+1$:

$$Z_1 Z_2 |\psi_{\text{qrep}}\rangle = |\psi_{\text{qrep}}\rangle \ . \tag{1.6}$$

On the other hand, $|\psi_{\text{qrep}}^{X_2\,\text{↯}}\rangle$ is an eigenstate with eigenvalue -1, indicating the presence of an error:

$$Z_1 Z_2 |\psi_{\text{qrep}}^{X_2\,\text{↯}}\rangle = -|\psi_{\text{qrep}}^{X_2\,\text{↯}}\rangle \ . \tag{1.7}$$

Similarly, measuring Z_2Z_3, Z_3Z_4, and Z_4Z_5 (which all commute and can therefore be measured simultaneously) on $|\psi_{\text{qrep}}^{X_2\,\text{↯}}\rangle$ gives -1, $+1$, and $+1$, indicating that a bit-flip must have occurred on the second qubit.

Therefore, by measuring Z_1Z_2, Z_2Z_3, Z_3Z_4, and Z_4Z_5, we can tell if a qubit suffered a bit-flip error, and furthermore identify which one, without destroying the superposition between the two logical computational basis states. Indeed, we can restore $|\psi_{\text{qrep}}\rangle$ by flipping the affected qubit that we've identified from the measurement outcomes.

⚠ **The measurements must be implemented as genuine two-body measurements, or else the encoded state is destroyed.** Measuring Z_1 and Z_2 separately and multiplying the outcomes would result in the same outcome as measuring the two-body operator Z_1Z_2; but such a measurement would destroy the superpositions in our encoded state and we wouldn't be able to restore $|\psi_{\text{qrep}}\rangle$ without prior knowledge of α and β. (We don't want to assume prior knowledge of α and β because the point of an error-correcting code is to store unknown information; if we already know what state we are currently storing, we wouldn't need to store it in the first place.)

What if an error causes a qubit to pick up a (-1) relative phase between its $|0\rangle$ and $|1\rangle$ states? In this case, the qubit undergoes the evolution $|0\rangle \to |0\rangle$, $|1\rangle \to -|1\rangle$. This error is called a ***phase-flip error*** or Z ***error***. In this case, the encoded state evolves to

$$|\psi_{\text{qrep}}^{Z_i\,\text{↯}}\rangle = \alpha|00000\rangle - \beta|11111\rangle \ . \tag{1.8}$$

Flash exercise: Show that there does not exist any measurement that can detect this error without destroying the superposition.

The problem is that the state $|\psi_{\text{qrep}}^{Z_i\,\text{↯}}\rangle$, which is the encoded version of the state $|\psi\rangle$ on which a Z_i error occurred, coincides exactly with the encoded version of the *different* logical state $|\psi'\rangle = \alpha|0\rangle - \beta|1\rangle$ on which no error occurred. It is therefore impossible to tell whether an error occurred without knowing *a priori* whether we encoded $|\psi\rangle$ or $|\psi'\rangle$; again, if we knew what the encoded state is, we wouldn't need to encode it!

The quantum repetition code therefore cannot detect phase flips.

Recall that the Hadamard gate $H = \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)/\sqrt{2} = H^\dagger$ interchanges the X and Z operators: $HXH = Z$, $HZH = X$. Therefore, if we apply a Hadamard gate onto each physical qubit

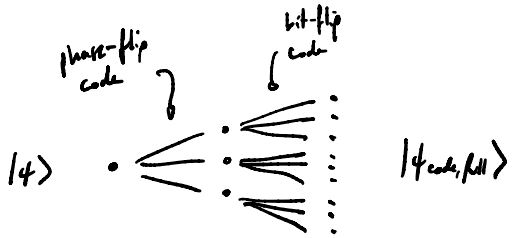
of the quantum repetition code, we obtain a code that can protect against phase-flip errors:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \rightarrow \quad |\psi_{\text{qrep,H}}\rangle = \alpha|+++++\rangle + \beta|-----\rangle \ . \tag{1.9}$$

Since the original encoding was vulnerable to phase-flip errors, the Hadamard-rotated version is vulnerable to bit-flip errors.

Is it possible to design a code that can correct any single-qubit error, regardless of whether it is an X or a Z error?

Consider three copies of the $1 \rightarrow 3$ quantum repetition code: This scheme uses 9 physical qubits to encode 3 qubits that are protected against up to one X error on any triplet of physical qubits. Suppose that we further use those three bit-flip-protected qubits to host a $1 \rightarrow 3$ quantum repetition code in the Hadamard basis. Then a phase flip error acting on any of the nine physical qubits induces a phase flip error on one of the three intermediate qubits, which can be corrected for by the Hadamard version of the repetition code. This code can therefore correct any X or Z single-qubit error on the nine physical qubits:



This code is constructed using the technique of ***code concatenation***, where the output of a code is fed to the input of another code.

In the above code, a qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is encoded as:

$$\begin{aligned}
&|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \\
&\xrightarrow{\text{phase-flip code}} \alpha|+++\rangle + \beta|---\rangle \\
&\xrightarrow{\text{bit-flip code}} \ \frac{\alpha}{2\sqrt{2}}\Big[(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)\Big] \\
&\qquad\qquad + \ \frac{\beta}{2\sqrt{2}}\Big[(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)\Big] \\
&\qquad\qquad = \alpha|\overline{0}\rangle + \beta|\overline{1}\rangle \ ,
\end{aligned} \tag{1.10}$$

where the computational basis logical states (or codewords) are given as

$$|\overline{0}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \ ; \tag{1.11}$$

$$|\overline{1}\rangle = \frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \ . \tag{1.12}$$

This code is known as the ***9-qubit Shor code*** which can correct any single-qubit bit-flip or phase-flip error.

Therefore, quantum error correction is possible! There are still many missing elements before we can actually protect quantum information in quantum hardware, but it is already

encouraging to see that it is in principle possible to protect information against some unknown noise. (Missing steps include: checking that we can correct for hardware-native errors such as dephasing noise as well as developing schemes for performing the relevant error-detecting measurements without risking introducing too many new errors with faulty measurements)

In summary, here are some of the challenges of designing quantum error-correcting codes which we've encountered above:

1) The no-cloning theorem prohibits encoding a state by simply repeating it multiple times. Protection cannot be achieved by naive redundancy.

2) Measurements that detect the presence of errors need to be chosen cleverly such that they do not collapse superpositions in the encoded state. They must typically be genuine multi-body measurements.

3) There are different types of errors. A code designed to protect against specifically one type of error will generically fail to protect against other types of errors.

4) Errors can also occur during the operations necessary for the error-correction protocol. They are generally problematic and can cause an uncontrolled accumulation of errors! We'll deal with this challenge later, when we'll shift the focus from constructing error-correcting codes to studying *fault-tolerant schemes* for quantum computation.

1.4 What's next?

Now that we've identified the main challenges we encounter when attempting to protect quantum information from noise, we'll proceed with an-depth analysis of how to construct schemes for quantum error correction and fault-tolerant quantum computation. In particular, we'll address the following points:

- Establish the fundamental theory of quantum error correction and identify some essential principles;
- Develop a formalism to systematically design and analyze a large class of quantum error-correcting codes;
- Construct the surface code and study its properties. The surface code will serve as our main go-to example of a fully-featured error-correcting code;
- Develop schemes to compute with encoded states and establish some principles of *fault tolerance*;
- Understand the formalism of quantum error correction beyond qubits: bosonic codes;
- ... and more!

1.5 A brief recap of the formalism of quantum information theory.

I'll be assuming that you are familiar with the fundamental concepts of quantum information theory and quantum computation. In particular, I'll assume familiarity with the following concepts, which I'm listing here to clarify some notation and definitions:

- To every quantum system A is associated a complex Hilbert space $\mathscr{H}_A$. For now, we'll assume that the Hilbert space dimension is finite. We'll comment on how to extend the concepts introduced here to infinite-dimensional systems when we introduce bosonic codes, but we'll keep our explanations at the level of the physical concepts, leaving a fully rigorous treatment to further reading.

- A system may be composed of subsystems $A, B, C, \ldots$, in which case the Hilbert space of the composed system is the tensor product of the individual Hilbert spaces of each subsystem:

$$\mathscr{H}_{ABC\ldots} = \mathscr{H}_A \otimes \mathscr{H}_B \otimes \mathscr{H}_C \otimes \cdots \ . \tag{1.13}$$

- The space of linear operators on $\mathscr{H}_A$ is denoted by $L(\mathscr{H}_A)$. It is equipped with the *Hilbert-Schmidt inner product* $\langle A, B\rangle \equiv \mathrm{tr}(A^\dagger B)$.

  An operator $A \in L(\mathscr{H}_A)$ is *Hermitian* if $A^\dagger = A$. An operator $U \in L(\mathscr{H}_A)$ is *unitary* if $U^\dagger U = UU^\dagger = \mathbb{1}_A$. An operator $V : \mathscr{H}_A \to \mathscr{H}_B$ is an *isometry* if $V^\dagger V = \mathbb{1}_A$. An operator $W : \mathscr{H}_A \to \mathscr{H}_B$ is a *partial isometry* if $W^\dagger W$ is a projector.

- A *ket*, or a *pure state* of A, is a vector $|\psi\rangle_A \in \mathscr{H}_A$ such that $\langle\psi|\psi\rangle = 1$.

- A *quantum state*, or simply *state* of A is a positive semidefinite operator $\rho_A \in L(\mathscr{H}_A)$ such that $\mathrm{tr}(\rho_A) = 1$. Such an operator is also called *density operator* or *density matrix*.

  A state ρ_A is *pure* if it can be written as $\rho_A = |\psi\rangle\langle\psi|_A$ for some $|\psi\rangle_A \in \mathscr{H}_A$. The term *pure state* is used both for $|\psi\rangle_A$ and $|\psi\rangle\langle\psi|_A$.

- An *observable* O is a Hermitian operator. A *measurement* of an observable $O \in L(\mathscr{H}_A)$ performed on a system A in the state ρ_A yields an *outcome* $x \in \mathbb{R}$ that is random and distributed according to $\Pr[x] = \mathrm{tr}(\rho_A P^{O,x})$, where $P^{O,x}$ is the projector onto the eigenspace of O corresponding to the eigenvalue x.

- The most general form of a *quantum measurement* is specified by a *positive-operator-valued measure (POVM)* $dE(x)$ with $x \in \Omega$, where Ω is some measurable space representing the possible measurement outcomes. The probability measure $d\mu(x)$ of obtaining the *outcome* $x \in \Omega$ if the system is in the state ρ is

$$d\mu(x) = \mathrm{tr}(dE(x)\,\rho) \ . \tag{1.14}$$

  Usually, Ω is finite, representing a finite collection of possible measurement outcomes. In this case, the POVM is specified by a collection of *POVM effects* $\{E_x\}_{x\in\Omega}$, where E_x is positive semidefinite, where $\sum_{x\in\Omega} E_x = \mathbb{1}$, and where the outcome x occurs with probability

$$\Pr[x] = \mathrm{tr}(E_x\rho) \ . \tag{1.15}$$

- Notation: For two operators $X, Y \in L(\mathscr{H}_A)$, we write $X \leq Y$ if $Y - X$ is positive semidefinite. (For example, a POVM effect E_x necessarily satisfies $0 \leq E_x \leq \mathbb{1}$.)

- The state ρ_A of a *closed system* evolves as $\rho_A \to U_t \rho_A U_t^\dagger$, where U_t is a unitary operator which depends on the time duration t of the evolution.

- The most general physical evolution of input quantum states on $\mathscr{H}_A$ to output quantum states on $\mathscr{H}_B$ is specified by a *completely positive (c.p.), trace-preserving*

(t.p.) map $\mathcal{E}_{A\to B}$, also known as a *quantum channel*, that maps input states to output states:

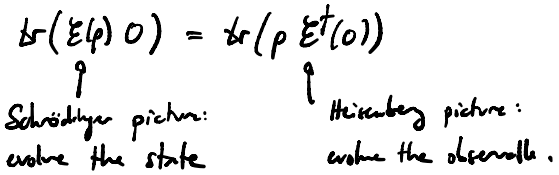
$$\begin{aligned}\mathcal{E}_{A\to B} : L(\mathscr{H}_A) &\to L(\mathscr{H}_B)\\ \rho_A &\mapsto \mathcal{E}_{A\to B}(\rho_A)\ .\end{aligned} \tag{1.16}$$

The map is *c.p.* if $(\mathcal{E}_A \otimes \mathrm{id}_R)(\rho_{AR}) \geq 0$ for any additional system R and for any state ρ_{AR}. The map is *t.p.* if $\mathrm{tr}(\mathcal{E}(\rho_A)) = 1$ for any state ρ_A.

- The *adjoint* of a map $\mathcal{E}$ is the unique map $\mathcal{E}^\dagger$ such that

$$\mathrm{tr}(X\mathcal{E}(Y)) = \mathrm{tr}(\mathcal{E}^\dagger(X)\,Y) \qquad \forall X, Y \in L(\mathscr{H})\ . \tag{1.17}$$

The adjoint map of a c.p., t.p. map gives the Heisenberg picture of time evolution:



Note that a map $\mathcal{E}_{A\to B}$ is t.p. if and only if $\mathcal{E}^\dagger(\mathbb{1}_B) = \mathbb{1}_A$.

- A map $\mathcal{E}_{A\to B}$ is c.p. if and only if it can be written in the *operator-sum representation*

$$\mathcal{E}_{A\to B}(\cdot) = \sum_k E_k\,(\cdot)\,E_k^\dagger\ , \tag{1.18}$$

where the $E_k : \mathscr{H}_A \to \mathscr{H}_B$ are arbitrary linear complex operators. The map $\mathcal{E}_{A\to B}$ is further t.p. if and only if $\sum_k E_k^\dagger E_k = \mathbb{1}_A$.

The operators $\{E_k\}$ are said to be *Kraus operators* of $\mathcal{E}_{A\to B}$.

- Any c.p., t.p. map $\mathcal{E}_{A\to B}$ admits a *Stinespring dilation* in the following form: There exists a system E and an isometry $U_{A\to BE}$ such that

$$\mathcal{E}_{A\to B} = \mathrm{tr}\Big\{U_{A\to BE}\,(\cdot)\,U^\dagger\Big\}\ . \tag{1.19}$$

The Stinespring dilation is unique up to partial isometries on E. The dimension of E need not exceed $\dim(\mathscr{H}_A)\dim(\mathscr{H}_B)$.

The isometry $U_{A\to BE}$ can be replaced by a unitary $U'_{AE_{\mathrm{in}}\to BE_{\mathrm{out}}}$, provided the environment systems E_{in} and E_{out} are chosen of suitable dimension and E_{in} is initialized in a fixed state.

- Notation: Given a real function $f : \mathbb{R} \to \mathbb{R}$, and given any Hermitian matrix A with spectral decomposition $A = \sum aP^a$, we write

$$f(A) = \sum_a f(a)P^a\ . \tag{1.20}$$

- The *trace norm* $\|A\|_1 = \mathrm{tr}\sqrt{A^\dagger A} = \mathrm{tr}|A|$ induces the *trace distance* between states,

$$\delta_{\mathrm{tr}}(\rho,\sigma) = \frac{1}{2}\|\rho - \sigma\|_1\ . \tag{1.21}$$

- The *fidelity* between two states ρ, σ is defined as

$$F(\rho,\sigma) = \operatorname{tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}} = \|\sqrt{\rho}\sqrt{\sigma}\|_1 \ . \tag{1.22}$$

  The fidelity and the trace distance both quantify the distinguishability of quatnum states.

- The *operator norm* or *infinity norm* $\|A\|_\infty$ is the largest singular value of A. If A is Hermitian, it coincides with the largest eigenvalue of A in magnitude.

- The *diamond distance* between two channels $\mathcal{E}^{(1)}_{A\to B}$, $\mathcal{E}^{(2)}_{A\to B}$ is

$$\frac{1}{2}\|\mathcal{E}^{(1)} - \mathcal{E}^{(2)}\|_\diamond = \max_{\sigma_{AR}} \frac{1}{2}\|\mathcal{E}^{(1)}(\sigma) - \mathcal{E}^{(2)}(\sigma)\|_1 \ , \tag{1.23}$$

  where the optimization ranges over all states σ_{AR} over the input system A and a reference system $R \simeq A$. The optimization may be restricted to pure states without changing the optimal value.

- The *von Neumann entropy* of a quantum state ρ_A on $\mathscr{H}_A$ is defined as

$$H(A)_\rho = -\operatorname{tr}(\rho\log\rho) \ . \tag{1.24}$$

  For a bipartite quantum state ρ_{AB} on $\mathscr{H}_A \otimes \mathscr{H}_B$, the *conditional von Neumann entropy* of A conditioned on B is

$$H(A|B)_\rho = H(AB)_\rho - H(A)_\rho \ . \tag{1.25}$$

- The *quantum relative entropy* between two states ρ_A and σ_A is defined as

$$D(\rho_A\|\sigma_A) = \operatorname{tr}\{\rho_A(\log\rho_A - \log\sigma_A)\} \ . \tag{1.26}$$

- The *Choi matrix* of a quantum channel $\mathcal{N}_{A\to B}$ is defined as

$$N_{BR} = \mathcal{N}_{A\to B}(\Phi_{B:R}) \ , \tag{1.27}$$

  where

$$\Phi_{AR} = \sum |i\rangle_A \otimes |i\rangle_R \tag{1.28}$$

  is a maximally entangled ket (or nonnormalized state) between A and a reference system $R \simeq A$ with computational bases $\{|i\rangle_A\}$ and $\{|i\rangle_R\}$, and where $\Phi_{AR} = |\Phi\rangle\langle\Phi|_{AR}$.

- The ket $|\Phi\rangle_{AR}$ defined above has useful properties:

  → $O_A|\Phi\rangle_{AR} = \Phi_{AR}$ if and only if $O_A = \mathbb{1}_A$;

  → $\mathcal{N}_{A\to A}(\Phi_{AR}) = \Phi_{AR}$ if and only if $\mathcal{N}_{A\to A} = \mathrm{id}_A$;

  → $(O_A \otimes \mathbb{1}_R)|\Phi\rangle_{AR} = \mathbb{1}_A \otimes (O^T)_R|\Phi\rangle_{AR}$, where $O^T = \sum\langle i|O|i'\rangle|i'\rangle\langle i|$ denotes the transpose with respect to the $\{|i\rangle\}$ basis.

  → For any pure state $|\rho_{AR}\rangle$, there exists $M \in L(\mathscr{H}_A)$ (a square $d_A \times d_A$ complex matrix) such that $|\rho_{AR}\rangle = (\mathbb{1}\otimes M)|\Phi\rangle_{AR}$, with $MM^\dagger = \rho_R$ and $(M^\dagger M)^T = \rho_A$.

- The *Pauli matrices* are

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} ; \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} ; \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} . \tag{1.29}$$

  We'll use these quite a bit!

1.6 Further reading

The role of the *Noisy Intermediate-Scale Quantum (NISQ)* regime was laid out in [15]. In the NISQ regime, devices operate beyond the regime in which we can compute classically with brute force what the device does. But errors accumulate quickly because no quantum error correction is carried out, limiting the size of circuits that can be run before the calculation becomes overwhelmed by noise. Experimental developments in this regime include demonstrations on superconducting qubits platforms by Google [16] and IBM [17].

A technique to reduce the effect of noise in the absence of quantum error correction is to use quantum error mitigation [18–20]. These techniques are an appealing and practical means of reducing errors on circuits, as they only involve classical processing and additional repetitions of the experiment. However, they require overheads in the number of additional repetitions of the experiment that scale exponentially with the circuit size, making this scheme impractical for larger circuits [21].

Another measure that can be taken against noise is to make sure that the coherent control of qubits is made as accurate as possible. A careful calibration of experimental parameters can make sure that a Hamiltonian term is turned on for the correct amount of time, for instance. Yet there are often undesired Hamiltonian terms that couple the qubit we wish to control to other qubits or other degrees of freedom on the device. A calibration might be quickly invalidated, for instance, in the presence of slowly varying background fields that are hard to control. In such cases, techniques based on dynamical decoupling [22] and modern techniques of quantum control and pulse engineering [23] can help cancel out the effect of unwanted Hamiltonian couplings.

But once these options are exhausted, and to run large or long circuits, we need a way of making sure errors are *removed* from the system at a rate at least as high as the rate at which they occur. This process can be achieved using techniques of quantum error correction and fault tolerance.

Chapter 2

Fundamental Theory of Quantum Error Correction I: Protecting information against noise

In this chapter, we'll develop the fundamental theoretical tools needed to understand the principles of quantum error correction. We'll define the task of quantum error correction and we'll find mathematical characterizations for when this task is achievable.

The fundamental theory of quantum error correction is split into two parts. In this chapter, we'll cover the information-theoretic principles of error correction and the fundamental conditions under which error correction is possible. In the following chapter, we'll establish some fundamental principles that apply to encodings involving multiple qubits, or more generally, multiple subsystems.

2.1 Encoding logical information on a physical system

The ***logical space*** $\mathscr{H}_L$ is a Hilbert space representing the state space of the quantum information (the *logical information*) we seek to protect from the noise.

The ***physical space*** $\mathscr{H}_P$ is a Hilbert space associated with the physical quantum system(s) that we use to encode the logical information.

Suppose we want to run an algorithm that requires 10 qubits with very low error rates, by using an error-correcting code hosted on a physical platform consisting of 200 noisy physical superconducting qubits. Then $\mathscr{H}_L = \mathbb{C}^{2^{10}}$ is the state space of the 10 qubits and $\mathscr{H}_P = \mathbb{C}^{2^{200}}$ is the state space of the 200 superconducting qubits.

An ***encoding map*** $\mathcal{E}_{L\to P}$ is a completely positive, trace-preserving map (i.e., a quantum channel) from L to P, such that there exists a quantum channel $\mathcal{R}_{P\to L}$ that satisfies $\mathcal{R}_{P\to L} \circ \mathcal{E}_{L\to P} = \mathrm{id}_L$ (i.e., we have $\mathcal{R}_{P\to L}(\mathcal{E}_{L\to P}(\rho)) = \rho$ for all states ρ on L).

The existence of $\mathcal{R}_{P\to L}$ ensures that the encoding procedure itself doesn't lose information, even in the absence of noise.

Side exercise: Show that for all ρ, σ: $D(\mathcal{E}(\rho)\|\mathcal{E}(\sigma)) = D(\rho\|\sigma)$, where $D(\rho\|\sigma) = \mathrm{tr}\big[\rho(\log\rho - \log\sigma)\big]$ is the quantum relative entropy.

An encoding map $\mathcal{E}_{L\to P}$ is an ***isometric encoding*** if $\mathcal{E}_{L\to P}(\cdot) = V_{L\to P}(\cdot)V^\dagger$ for some isometry $V_{L\to P}$.

Recall that an isometry $V_{L\to P}$ is defined as an operator $\mathscr{H}_L \to \mathscr{H}_P$ such that $V^\dagger V = \mathbb{1}_L$. Most encodings we'll study are isometric.

An isometry $V_{L\to P}$ can be thought of as an embedding of states on $\mathscr{H}_L$ into the larger Hilbert space $\mathscr{H}_P$. It can be specified by how the basis states of $\mathscr{H}_L$ are mapped to states in $\mathscr{H}_P$:

$$\begin{aligned} |0\rangle_L &\to V_{L\to P}|0\rangle_L =: |\psi_0\rangle_P \\ |1\rangle_L &\to V_{L\to P}|1\rangle_L =: |\psi_1\rangle_P \\ &\vdots \end{aligned} \tag{2.1}$$

For any isometry $V_{L\to P}$, the map $\mathcal{E}(\cdot) = V(\cdot)V^\dagger$ is a valid encoding map.

Flash exercise: Find a channel $\mathcal{R}$ for an isometric encoding $\mathcal{E}(\cdot) = V(\cdot)V^\dagger$ such that $\mathcal{R}\circ\mathcal{E} = \mathrm{id}$.

Let $\mathcal{E}_{L\to P}$ be an isometric encoding map, $\mathcal{E}_{L\to P}(\cdot) = V_{L\to P}(\cdot)V^\dagger$. Then the subspace of $\mathscr{H}_P$ that is the image (or range) of $V_{L\to P}$ is called the ***code space*** $\mathcal{C}$.

We often denote the projector onto the code space $\mathcal{C}$ of an isometric encoding $\mathcal{E}(\cdot) = V(\cdot)V^\dagger$ by Π (***code space projector***):

$$\Pi = VV^\dagger \ . \tag{2.2}$$

Any ket $|\psi\rangle \in \mathcal{C}$ that lies in the code space $\mathcal{C}$ of an isometric encoding is called a ***code word***. (Sometimes, the term "code word" is used to designate specifically the encoded states $\{|\psi_j\rangle_P\}$ of a canonical basis $\{|j\rangle_L\}$ of $\mathscr{H}_L$: $|\psi_j\rangle_P = V_{L\to P}|j\rangle_L$.)

For an isometric encoding with code space projector Π, a ***logical operator*** is an operator O on $\mathscr{H}_P$ that satisfies

$$\Pi O \Pi = O\Pi \ . \tag{2.3}$$

If O is Hermitian, or if O is unitary, then O is a logical operator if and only if $[O,\Pi] = 0$.

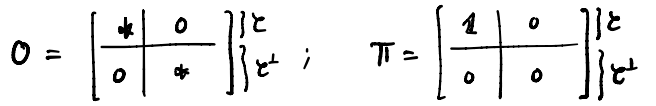
Simple exercise: Prove the last statement.

The logical operator is a ***nontrivial logical operator*** if its action on the code space is not proportional to $\mathbb{1}_L$, i.e., $\Pi O \Pi \not\propto \Pi$. Equivalently, $V^\dagger O V \not\propto \mathbb{1}_L$, where V is the encoding isometry.

Logical operators are precisely those operators that map code words into code words:

$$O \text{ is a logical operator} \quad\Longleftrightarrow\quad O|\psi\rangle \in \mathcal{C} \quad \forall\, |\psi\rangle \in \mathcal{C} \ . \tag{2.4}$$

A Hermitian or unitary logical operator is block-diagonal in a basis of $\mathcal{C}$, complemented by a basis of $\mathcal{C}^\perp$:



Example: The encoding of one qubit into four qubits given by

$$\begin{aligned} |0\rangle_L \to |\overline{0}\rangle_P &= \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \ ; \\ |1\rangle_L \to |\overline{1}\rangle_P &= \frac{1}{\sqrt{2}}(|0011\rangle + |1100\rangle) \ , \end{aligned} \tag{2.5}$$

is isometric, with encoding isometry $V_{L\to P} = |\overline{0}\rangle_P\langle 0|_L + |\overline{1}\rangle_P\langle 1|_L$. Its code space is spanned by the code words $\{|\overline{0}\rangle_P, |\overline{1}\rangle_P\}$.

Simple exercise: Show that $X \otimes X \otimes \mathbb{1} \otimes \mathbb{1}$ is a nontrivial logical operator.

Some encoding maps are not isometric. Such maps may encode a pure logical state into a physical state that is mixed. Yet often, many encoding maps can still be thought of as isometric in some way, despite not being strictly speaking isometric. For instance:

- *Encoding with a "dummy mixed state:"* Consider an encoding map of the form

$$\mathcal{E}_{L\to P}(\cdot) = U((\cdot)\otimes\tau)U^\dagger \ , \tag{2.6}$$

  where U is some unitary operator and where τ is some fixed state. The map is clearly an encoding map, with a corresponding recovery operation $\mathcal{R}$ defined by applying the inverse unitary and tracing out the system containing the fixed state τ.

  While $\mathcal{E}_{L\to P}$ is not an isometric encoding in general, it becomes an isometric encoding if τ is chosen to be a pure state. (We'll see that certain types of codes, called *subsystem codes*, can be viewed as having encoding maps of this type.)

- *Encoding with a random, but known, isometric encoding:* Given a collection of isometries $\{V^{(j)}_{L\to P_0}\}_j$, and a probability distribution $\{p_j\}$, we can define the following encoding map:

$$\mathcal{E}_{L\to P_0 X}(\cdot) = \sum_j p_j [V^{(j)}(\cdot)V^{(j)\,\dagger}] \otimes |j\rangle\langle j|_X \ , \tag{2.7}$$

  where X is a classical register and where we formally identify $P \simeq P_0 X$ as the physical space. The encoding corresponds to randomly choosing an isometric encoding $V^{(j)}$ with probability p_j, encoding the logical information with $V^{(j)}$ from L to P_0, and storing the random choice j on a classical register. This map is a valid encoding map, with associated channel $\mathcal{R}$ given by measuring the classical register and applying the corresponding inverse isometry.

  While $\mathcal{E}_{L\to P}$ is not an isometric encoding, it can be viewed as an isometric encoding if we condition on the value of X, i.e., on the choice j of which isometric code is employed.

The second example above can be understood as a special case of the first example; indeed, choose τ as a diagonal encoding of the distribution p_j and U as executing one of the $V^{(j)}$ conditioned on the value in X.

In fact, *any* encoding map must be essentially of the form of the first example (this structure follows from algebraic arguments; see further reading). This is partly the reason why we usually don't worry too much about nonisometric encodings in quantum error correction.

2.2 Protecting against noise

We model the noise as any possible quantum evolution that can involve an environment. Such evolutions are specified by completely positive maps.

We model ***noise*** by a completely positive, trace-preserving map (quantum channel) $\mathcal{N}_{P\to P'}$.

For example, the *depolarizing noise channel* on a single qubit ($P' = P =$ one qubit) with depolarizing rate $0 \leq p \leq 3/4$ is

$$\mathcal{N}^{(p)}_{\text{depol.}}(\rho) = (1-p)\rho + \frac{p}{3}X\rho X + \frac{p}{3}Y\rho Y + \frac{p}{3}Z\rho Z \tag{2.8}$$

$$= \left(1 - \frac{4}{3}p\right)\rho \;+\; \frac{4}{3}p\,\frac{\mathbb{1}_2}{2}\;. \tag{2.9}$$

The noise channel is formally defined with an output space P' that can differ from the input space P. While we usually have $P' = P$, it is occasionally convenient to formally include into P' any additional states or memory registers that contain additional information about the errors that occurred.

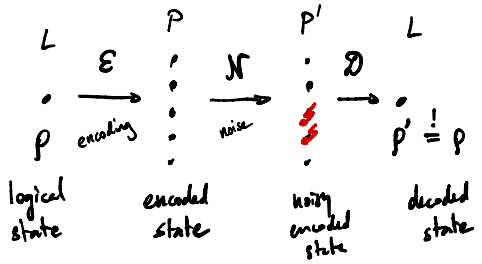
We are now in a position to formally define what it means for a code to protect against a specific type of noise.

An encoding $\mathcal{E}_{L\to P}$ can ***correct***, or ***protect against***, a noise channel $\mathcal{N}_{P\to P'}$ if there exists a quantum channel $\mathcal{D}_{P'\to L}$ such that

$$\mathcal{D}_{P'\to L} \circ \mathcal{N}_{P\to P'} \circ \mathcal{E}_{L\to P} = \mathrm{id}_L\;. \tag{2.10}$$

We also say that $\mathcal{E}_{L\to P}$ is $\mathcal{N}$***-correcting***.

Schematically:



The channel $\mathcal{D}_{P'\to L}$ is the ***decoding channel***. (It is sometimes referred to also as ***recovery map*** and denoted by "$\mathcal{R}$" in some literature works, not to be confused with the "$\mathcal{R}$" in the definition of an encoding map.) This channel includes all operations necessary to undo the effect of the noise, such as performing measurements, running a classical algorithm to infer what error happened, and applying any relevant corrections.

In the interest of defining general-purpose error-correcting codes that can correct more than a very specific error channel $\mathcal{N}$, we formally extend the definition above to identify the ability of an encoding to protect against an entire family of noise channels. We'll see

that the requirement that an encoding protects against an entire family of noise channels need not be particularly restrictive compared to protecting against a single noise channel, ensuring the existence of useful general-purpose error-correcting codes.

An encoding $\mathcal{E}_{L\to P}$ can ***protect against a family*** $\{\mathcal{N}^{(\alpha)}_{P\to P'}\}_\alpha$ of noise channels if there exists a single quantum channel $\mathcal{D}_{P'\to L}$ such that for all α,

$$\mathcal{D}_{P'\to L}\circ\mathcal{N}^{(\alpha)}_{P\to P'}\circ\mathcal{E}_{L\to P}=\mathrm{id}_L \qquad (\forall\ \alpha)\ . \tag{2.11}$$

Note that $\mathcal{D}_{P'\to L}$ is not allowed to depend on α. Therefore, the ability of $\mathcal{E}_{L\to P}$ to correct a family $\{\mathcal{N}^{(\alpha)}_{P\to P'}\}_\alpha$ implies, but is not implied by, the ability to correct each member of the family.

We obtain a more intuitive picture of what noise channels a given encoding can correct by studying the Kraus decomposition of the noise channel:

$$\mathcal{N}_{P\to P'}(\cdot)=\sum_k E_k(\cdot)E_k^\dagger\ , \tag{2.12}$$

with $E_k\in\mathbb{C}^{d_{P'}\times d_P}$ and $\sum_k E_k^\dagger E_k=\mathbb{1}_P$.

We can think of the $\{E_k\}$ as "errors" happening on the system: We can interpret the noise channel as one of the operators $\{E_k\}$ being applied on the system, yet without knowledge of which E_k was applied.

Example: A Kraus representation of the depolarizing noise channel (2.8) can be read off as:

$$\underbrace{E_0=\sqrt{1-p}\,\mathbb{1}}_{\text{no error}}\ ,\quad \underbrace{E_1=\sqrt{\tfrac{p}{3}}X}_{X\text{ error}}\ ,\quad \underbrace{E_2=\sqrt{\tfrac{p}{3}}Y}_{Y\text{ error}}\ ,\quad \underbrace{E_3=\sqrt{\tfrac{p}{3}}Z}_{Z\text{ error}}\ .$$

Noise channels often have Kraus operators of the form $E_0\approx\mathbb{1}$ and $E_k\approx 0$ $(k\geq 1)$, like the depolarizing noise channel. The $\{E_k\}_{k\geq 1}$ can be thought of as nontrivial errors that the system suffers. In fact, if $P'=P$, any channel $\mathcal{N}$ that is close to the identity channel (with $\|\mathcal{N}-\mathrm{id}_P\|_\diamond\approx 0$) has Kraus operators of this form [1, Lemma 1.3].

It turns out that codes can generally protect against a big family of noise channels, thanks to the property of the *linearity of quantum error correction*. This theorem states that a code that can correct a noise channel with a given set of Kraus operators $\{E_k\}$ can automatically correct the entire family of noise channels with Kraus operators that are linear combinations of the $\{E_k\}$.

Theorem 1 (Linearity of quantum error correction)**.** *Let $\mathcal{E}_{L\to P}$ be an isometric encoding map that protects against a noise channel with Kraus representation $\mathcal{N}_{P\to P'}(\cdot)=\sum_k E_k(\cdot)E_k^\dagger$. Then $\mathcal{E}_{L\to P}$ protects against the entire family of noise channels whose Kraus operators lie in the linear span of $\{E_k\}$:*

$$\left\{\mathcal{N}'\ :\ \mathcal{N}'(\cdot)=\sum F_\ell(\cdot)F_\ell^\dagger\ ,\ F_\ell\in\mathrm{span}\{E_k\}\right\}\ . \tag{2.13}$$

Recall that the latter statement means that a single decoding channel $\mathcal{D}_{P'\to L}$ can ensure recovery of the logical information regardless of which noise channel from the family is actually applied.

Proof. Since $\mathcal{E}_{L\to P}$ corrects $\mathcal{N}_{P\to P'}$, there exists $\mathcal{D}_{P'\to L}$ such that

$$\mathcal{D}_{P'\to L}\circ\mathcal{N}_{P\to P'}\circ\mathcal{E}_{L\to P}=\mathrm{id}_L\ . \tag{2.14}$$

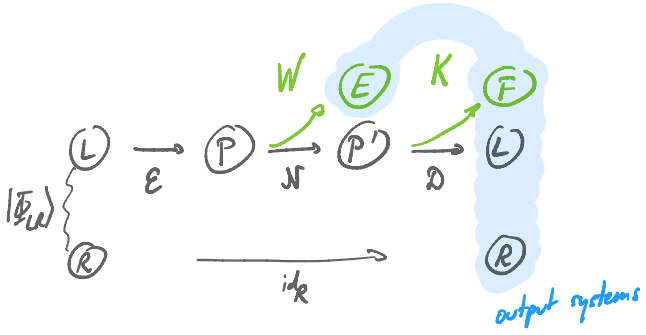
Let $K_{P'\to LF}$ be a Stinespring dilation isometry of $\mathcal{D}_{P'\to L}$, such that

$$\mathcal{D}_{P'\to L}(\cdot)=\mathrm{tr}_F\{K_{P'\to LF}\,(\cdot)\,K^\dagger\}\ , \tag{2.15}$$

and consider the Stinespring dilation of $\mathcal{N}_{P\to P'}$ given by

$$W_{P\to P'E}=\sum_k|k\rangle_E\otimes E_k\ ;\qquad\qquad \mathcal{N}_{P\to P'}(\cdot)=\mathrm{tr}_E\{W\,(\cdot)\,W^\dagger\}\ . \tag{2.16}$$

Let $|\Phi\rangle_{L:R}=\sum_i|i\rangle_L|i\rangle_R$ be a maximally entangled ket (nonnormalized state) between L and a reference system R. The purified process can be visualized as follows:



Because $\mathcal{E}$ corrects $\mathcal{N}$ [i.e., (2.14)], we have

$$\mathrm{tr}_{EF}\{K_{P'\to LF}W_{P\to P'E}V_{L\to P}\,\Phi_{L:R}\,V^\dagger W^\dagger K^\dagger\}=\Phi_{L:R}\ , \tag{2.17}$$

where $\Phi_{L:R}\equiv|\Phi\rangle\langle\Phi|_{L:R}$. Since $\Phi_{L:R}$ is pure, there exists some state $|\chi\rangle_{EF}$ such that

$$K_{P'\to LF}W_{P\to P'E}V_{L\to P}\,|\Phi\rangle_{L:R}=|\Phi\rangle_{L:R}\otimes|\chi\rangle_{EF}\ . \tag{2.18}$$

(If the reduced state of a globally pure quantum state is also pure, the state is necessarily a tensor product of pure states across the system being traced out and the remaining system.)

Plugging in the definition of $W_{P\to P'E}$ [Eq. (2.16)], we find for any k that

$$K_{P'\to LF}\,E_k\,V_{L\to P}\,|\Phi\rangle_{L:R}=|\Phi\rangle_{L:R}\otimes(\langle k|_E)|\chi\rangle_{EF}\ . \tag{2.19}$$

Now consider any quantum channel $\mathcal{N}'(\cdot)=\sum F_\ell(\cdot)F_\ell^\dagger$ such that $F_\ell\in\mathrm{span}\{E_k\}$. By

definition, $F_\ell = \sum \alpha_{k,\ell} E_k$ for some coefficients $\alpha_{k,\ell} \in \mathbb{C}$. Let $W' = \sum_\ell |l\rangle_E \otimes F_\ell$. We find

$$\begin{aligned} K_{P'\to LF} W'_{P\to P'E} V_{L\to P} \, |\Phi\rangle_{L:R} \\ &= \sum_\ell |\ell\rangle_E \otimes \big(K_{P'\to LF} F_\ell V |\Phi\rangle_{L:R}\big) \\ &= \sum_{k,\ell} |\ell\rangle_E \otimes \big(\langle k|_E |\chi\rangle_{EF} \otimes |\Phi\rangle_{L:R}\big) \\ &= |\Phi\rangle_{L:R} \otimes \Big(\sum_{k,\ell} \alpha_{k,\ell} |\ell\rangle\langle k|_E \, |\chi\rangle_{EF}\Big) \\ &=: |\Phi\rangle_{L:R} \otimes |\chi'\rangle_{EF} \ . \end{aligned} \tag{2.20}$$

Therefore,

$$\begin{aligned} \mathcal{D}(\mathcal{N}'(\mathcal{E}(\Phi_{L:R}))) \\ &= \mathrm{tr}_{EF}\big\{K_{P'\to LF} W'_{P\to P'E} V_{L\to P} \, \Phi_{L:R} \, V^\dagger W'^\dagger K^\dagger\big\} \\ &= \Phi_{L:R} \, \langle\chi'|\chi'\rangle = \Phi_{L:R} \ , \end{aligned} \tag{2.21}$$

where the last equality holds because all the maps $\mathcal{E}$, $\mathcal{N}'$, and $\mathcal{D}$ are trace preserving.

Therefore, $\mathcal{D} \circ \mathcal{N}' \circ \mathcal{E} = \mathrm{id}_L$ and $\mathcal{D}$ is also a decoding channel for $\mathcal{N}'$. The encoding $\mathcal{E}$ can thus correct (with the same decoding channel $\mathcal{D}$) the entire family of all channels $\mathcal{N}'$ whose Kraus operators line in $\mathrm{span}\{E_k\}$. ∎

An important consequence of the linearity of quantum error correction is that *rather than designing encodings for a specific noise channel $\mathcal{N}$, we can focus on a basis of error operators that span a space containing all errors that we want to be able to correct.*

Let's define an ***error set*** Err as any set of operators from P to P', subject to the technical condition that there exists some channel $\mathcal{N}'$ whose Kraus operators $\{E_k\}$ satisfy $\mathrm{span}\{E_k\} = \mathrm{span}\,\mathsf{Err}$. (The condition on Err can be viewed as a technicality that prohibits us from including an operator in the set Err that cannot be included in any actual c.p., t.p. noise map whose Kraus operators are linear combinations of elements of Err. The error sets we'll consider naturally satisfy this property.)

We say that an encoding $\mathcal{E}_{L\to P}$ ***corrects an error set*** Err if it corrects the family of all quantum channels whose Kraus operators lie in $\mathrm{span}\,\mathsf{Err}$.

Example: If P consists of n qubits, a basis of all operators on P is the *Pauli basis* formed by all tensor products of $\mathbb{1}$'s and single-qubit Pauli operators (e.g.: $X \otimes \mathbb{1} \otimes Z \otimes Z$). We can define an error set consisting of those Pauli operators that act nontrivially on at most t sites ($0 < t < n$). (The technical condition on Err is satisfied by noting that all these Pauli operators, scaled by an appropriate constant, form the Kraus operators of some channel.) As we'll see, many quantum error-correcting codes are designed to correct precisely this error set.

2.3 Criteria for quantum error correction

When can an encoding $\mathcal{E}_{L\to P}$ correct an error set (and therefore the family of quantum channels with Kraus operators in the set's linear span)? It turns out that there is a general

and elegant set of conditions that determine exactly when an isometric encoding can correct a set of errors. These are the well-known *Knill-Laflamme conditions*, which is the focus of this section.

In this section, we'll focus exclusively on isometric encodings for convenience. More general encodings will be considered in the next section (Section 2.6).

Theorem 2 (Knill-Laflamme conditions)**.** *An isometric encoding $\mathcal{E}_{L\to P}$ with code space projector Π corrects an error set $\mathsf{Err} = \mathrm{span}\{E_k\}$ if and only if for all k, k', there exists $c_{kk'} \in \mathbb{C}$ such that*

$$\Pi E_{k'}^\dagger E_k \Pi = c_{kk'} \Pi \ . \tag{2.22}$$

There are several equivalent formulations of the Knill-Laflamme conditions:

$$\begin{aligned}
\text{Eq. (2.22)} \ \forall\, k,k' \quad &\Leftrightarrow \quad \Pi E_{k'}^\dagger E_k \Pi \propto \Pi \quad \forall\, k,k' \\
&\Leftrightarrow \quad V^\dagger E_{k'}^\dagger E_k V \propto \mathbb{1}_L \quad \forall\, k,k' \\
&\Leftrightarrow \quad \exists\ c_{kk'} \ :\ \forall\, |\psi\rangle \in \mathcal{C} \ :\ \langle\psi|\, E_{k'}^\dagger E_k\, |\psi\rangle = c_{kk'} \\
&\Leftrightarrow \quad \exists\ c_{kk'} \ :\ \forall\, i,j \ :\ \langle\psi_j|\, E_{k'}^\dagger E_k\, |\psi_i\rangle = \delta_{i,j}\, c_{kk'} \ , \\
&\Leftrightarrow \quad \Pi\, E'^\dagger E\, \Pi \propto \Pi \quad \forall\, E, E' \in \mathsf{Err} \\
&\Leftrightarrow \quad \Pi M_{\ell'}^\dagger M_\ell \Pi \propto \Pi \quad \forall\, \ell,\ell' \ ,
\end{aligned} \tag{2.23}$$

where $V_{L\to P}$ is the isometry defining the isometric encoding, $\mathcal{C}$ is the corresponding code space, $|\psi_j\rangle = V_{L\to P}|j\rangle_L$ are the computational basis code words, $\delta_{i,j}$ is the Kronecker delta symbol ($\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ if $i \neq j$), and $\{M_\ell\}$ is any other set of operators with $\mathsf{Err} = \mathrm{span}\{M_\ell\}$.

Exercise: Prove the above equivalences.

The Knill-Laflamme conditions (2.22) imply that a pair of an error operator and the adjoint of another error cannot map one codeword to a state that has overlap with another codeword:

$$\langle\psi_j|\, E_{k'}^\dagger E_k\, |\psi_i\rangle = 0 \quad \forall\, i \neq j,\ \forall\, k,k' \ . \tag{2.24}$$

We can ask why the pair $E_{k'}^\dagger E_k$ appears in (2.24), rather than a single error operator E_k. Why does it not suffice that a single error operator E_k not map a code word $|\psi_i\rangle$ to a state with overlap with a different code word $|\psi_j\rangle$, i.e., $\langle\psi_j|\, E_k\, |\psi_i\rangle = 0 \ \ \forall\, i \neq j,\ \forall\, k$? The answer is that to correct errors, we need a state of the form $E_k|\psi_i\rangle$ to be orthogonal to any other state of the same form $E_{k'}|\psi_j\rangle$ where $j \neq i$, or else we would be unable to tell if the error E_k occurred on the code word $|\psi_i\rangle$ or if $E_{k'}$ occurred on the code word $|\psi_j\rangle$. The overlap of two such states is precisely given by $\langle\psi_j|\, E_{k'}^\dagger E_k\, |\psi_i\rangle$.

The coefficient matrix $[c_{kk'}]_{k,k'}$ is Hermitian, $c_{kk'} = c_{k'k}^*$, as can be seen by taking the adjoint of Equation Eq. (2.22) and reversing the roles of k and k'. As a consequence, we can diagonalize $[c_{kk'}]$ as

$$\sum_{k,k'} u_{k\ell}\, c_{kk'}\, u_{k'\ell'}^* = \lambda_\ell\, \delta_{\ell,\ell'} \ , \tag{2.25}$$

where $[u_{k\ell}]$ is a unitary matrix. Now let

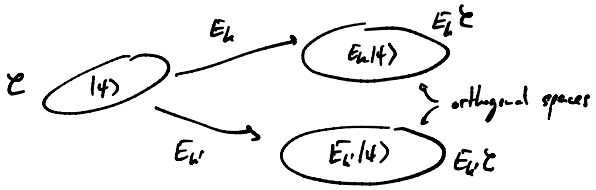
$$E'_\ell = \sum_k u_{k\ell} E_k \ . \tag{2.26}$$

We find

$$\Pi E'^\dagger_{\ell'} E'_\ell \Pi = \sum_{k,k'} u_{k\ell} u^*_{k'\ell'} \underbrace{\Pi E^\dagger_{k'} E_k \Pi}_{c_{kk'}\Pi} = \lambda_\ell \, \delta_{\ell,\ell'} \, \Pi \ . \tag{2.27}$$

Therefore, for any set of generating errors $\{E_k\}$, the transformed set of generating errors (2.26) diagonalizes the $[c_{kk'}]$ coefficients. If $\{E_k\}$ are the Kraus operators of some noise channel $\mathcal{N}$, then recall that the transformation (2.26) yields operators E'_ℓ that are simply another Kraus representation of the *same* noise channel.

Let's choose a basis $\{E_k\}$ of error operators that diagonalize the Knill-Laflamme conditions. We see that the Knill-Laflamme conditions imply that each E_k maps the code space to different, orthogonal subspaces:



We can thus determine which error happened without perturbing the logical state, by measuring which subspace $E_k\mathcal{C}$ the state lies in without resolving the states within each subspace. The correction then amounts to applying a suitable unitary that maps the error space $E_k\mathcal{C}$ back onto $\mathcal{C}$.

If the matrix $[c_{kk'}]$ has maximal rank, the code is said to be ***nondegenerate***. In this case, there is a basis $\{E_k\}$ of the span of the error operators such that the subspaces $\{E_k\mathcal{C}\}$ are all nontrivial and pairwise orthogonal.

Otherwise, the code is said to be ***degenerate***. If the code is degenerate, a basis $\{E_k\}$ that diagonalizes $[c_{kk'}]$ must contain one or more element(s) E_k that annihilate the code space: $E_k\mathcal{C} = 0$. (Indeed, if $[c_{kk'}]$ does not have maximal rank, then one or more of the λ_k's must vanish.) These errors can never happen as long as the state lies in the code space!

Proof of Theorem 2. From the definition of an error set, we can choose without loss of generality the $\{E_k\}$ both to span the space span Err as well as to be the Kraus operators of some channel $\mathcal{N}_{P\to P'}$. (We've already seen that the Knill-Laflamme conditions do not depend on the elements chosen to span the space span Err.) The proof strategy is to show that the Knill-Laflamme conditions (2.22) are both necessary and sufficient for correcting the given set of errors.

The Knill-Laflamme conditions are necessary: By assumption, there exists a quantum channel $\mathcal{D}_{P'\to L}$ such that $\mathcal{D}_{P'\to L} \circ \mathcal{N}_{P\to P'} \circ \mathcal{E}_{L\to P} = \mathrm{id}_L$. Denote by $\{R_\ell\}$ a set of Kraus operators of $\mathcal{D}_{P'\to L}$, noting that $\sum_\ell R^\dagger_\ell R_\ell = \mathbb{1}_{P'}$. Let $K_{P'\to LF} = \sum_\ell |\ell\rangle_F \otimes R_\ell$, and observe that $K_{P'\to LF}$ is a Stinespring dilation isometry of $\mathcal{D}_{P'\to L}$, with

$$\mathcal{D}_{P'\to L}(\cdot) = \mathrm{tr}_F\{K_{P'\to LF}\,(\cdot)\,K^\dagger\} \ . \tag{2.28}$$

Let as before

$$W_{P\to P'E} = \sum_k |k\rangle_E \otimes E_k \ ; \qquad\qquad \mathcal{N}(\cdot) = \mathrm{tr}_E\{W\,(\cdot)\,W^\dagger\}\ , \tag{2.29}$$

and $|\Phi\rangle_{L:R} = \sum |i\rangle_L |i\rangle_R$. As we saw earlier (cf. proof of Theorem 1), there exists $|\chi\rangle_{EF}$ such that

$$K_{P'\to LF}\, W_{P\to P'E}\, V_{L\to P}\, |\Phi\rangle_{L:R} = |\Phi\rangle_{L:R} \otimes |\chi\rangle_{EF}\ . \tag{2.30}$$

Multiplying on the left by $(\langle k|_E \otimes \langle \ell|_F)$, we find

$$R_\ell\, E_k\, V\, |\Phi\rangle_{L:R} = \underbrace{(\langle k|_E \otimes \langle \ell|_F)|\chi\rangle_{EF}}_{=:\,\mu_{k\ell}}\, |\Phi\rangle_{L:R}\ . \tag{2.31}$$

Therefore, there exists $\mu_{k\ell} \in \mathbb{C}$ such that $R_\ell E_k V = \mu_{k\ell} \mathbb{1}_L$. Now, for any k, k' we have

$$\begin{aligned} V^\dagger E_{k'}^\dagger E_k V &= V^\dagger E_{k'} \Big(\sum_\ell R_\ell^\dagger R_\ell\Big) E_k V \\ &= \sum_\ell (V^\dagger E_{k'}^\dagger R_\ell^\dagger)(R_\ell E_k V) = \Big(\sum_\ell \mu_{k\ell}^* \mu_{k'\ell}\Big)\mathbb{1}_L =: c_{kk'}\mathbb{1}_L\ . \end{aligned} \tag{2.32}$$

Therefore, the Knill-Laflamme conditions are satisfied.

The Knill-Laflamme conditions are sufficient: We assume that (2.22) hold true, and our goal is to construct a map $\mathcal{D}_{P'\to L}$ such that $\mathcal{D}\circ\mathcal{N}\circ\mathcal{E} = \mathrm{id}_L$. By linearity of quantum error correction, the latter statement imples that $\mathcal{E}$ can correct the error set Err.

We've seen that the operators $\{E_k\}$ can be transformed via (2.26) to a set of operators $\{E'_\ell\}$ which represent the same quantum channel $\mathcal{N}_{P\to P'}$ and which diagonalize the Knill-Laflamme conditions:

$$V^\dagger E_{\ell'}'^\dagger E'_\ell V = \lambda_\ell\, \delta_{\ell,\ell'}\, \mathbb{1}_L \qquad \forall\ \ell, \ell'\ . \tag{2.33}$$

Furthermore, taking the sum of this equation over ℓ with $\ell' = \ell$ gives

$$\Big(\sum_\ell \lambda_\ell\Big)\,\mathbb{1}_L = \sum_\ell V^\dagger E_\ell'^\dagger E'_\ell V = V^\dagger \Big(\sum E_\ell'^\dagger E'_\ell\Big) V = \mathbb{1}_L\ , \tag{2.34}$$

so we see that $\sum \lambda_\ell = 1$.

We can now begin defining Kraus operators for our decoding channel $\mathcal{D}_{P'\to L}$. Let

$$R_\ell = \frac{1}{\sqrt{\lambda_\ell}}\, V^\dagger E_\ell'^\dagger\ . \tag{2.35}$$

We now study the operator

$$Q = \sum_\ell R_\ell^\dagger R_\ell\ . \tag{2.36}$$

To ensure that the operators $\{R_\ell\}$ form the Kraus operators of some quantum channels, we would need $Q = \mathbb{1}$. But it turns out this is not the case yet in general. Indeed, Q is

Hermitian and is a projector:

$$Q^2 = \sum_{\ell,\ell'} \frac{1}{\lambda_\ell \lambda_{\ell'}} E'_{\ell'} V \underbrace{V^\dagger E'^\dagger_{\ell'} E'_\ell V}_{\delta_{\ell,\ell'}\,\lambda_\ell} V^\dagger E^\dagger_\ell = \sum_\ell \frac{1}{\lambda_\ell} E'_\ell V V^\dagger E'^\dagger_\ell = Q \ . \tag{2.37}$$

We see that Q projects onto the subspace of states $\mathrm{span}\{E_k|\psi_j\rangle\}$ reachable by applying an error to a state in the code space.

To define $\mathcal{D}$ as a valid quantum channel, we set:

$$\mathcal{D}_{P'\to L}(\cdot) = \sum_\ell R_\ell\,(\cdot)\,R^\dagger_\ell \ + \ (\mathbb{1} - Q)(\cdot)(\mathbb{1} - Q) \ . \tag{2.38}$$

(We could replace the last term with any other term that acts only on the subspace orthogonal to the support of Q.)

Let's now check that $\mathcal{D}$ successfully recovers the corrupted information. For any $|\phi\rangle_L$, we find

$$\begin{aligned}
\mathcal{D}\circ\mathcal{N}\circ\mathcal{E}(\phi_L) &= \sum_{\ell,\ell'} R_{\ell'} E'_\ell V\,\phi\,V^\dagger E'^\dagger_\ell R^\dagger_{\ell'} \\
&= \sum_{\ell,\ell'} \frac{1}{\lambda_{\ell'}} \underbrace{V^\dagger E'^\dagger_{\ell'} E'_\ell V}_{=\delta_{\ell,\ell'}\lambda_\ell}\ \phi_L\ \underbrace{V^\dagger E'^\dagger_\ell E'_{\ell'} V}_{=\delta_{\ell,\ell'}\lambda_\ell} \\
&= \Big(\sum \lambda_\ell\Big)\,\phi_L = \phi_L \ .
\end{aligned} \tag{2.39}$$

Therefore, $\mathcal{D}\circ\mathcal{N}\circ\mathcal{E} = \mathrm{id}_L$ as desired. ∎

2.4 Error-detecting codes

Sometimes, we might only care about determining whether an error happened, without worrying about correcting it. In such a case we can, perhaps, throw the system away and start again.

To detect errors, we use an isometric encoding $\mathcal{E}_{L\to P}$ to store the logical information $|\phi\rangle_L$ on the physical system as a state $|\psi\rangle_P$. The physical system is then exposed to a noise channel $\mathcal{N}_{P\to P'}$. Rather than attempting to recover the logical information, as in the case of quantum error correction, we demand that the two-outcome measurement with POVM effects $\{\Pi, \mathbb{1} - \Pi\}$ behave as follows:

- The outcome $\mathbb{1} - \Pi$ signifies an error has occurred and our system needs to be thrown away;
- If we obtain the outcome Π, then the state projected by the measurement is again $|\psi\rangle_P$, up to normalization associated with the probability of this outcome.

We say an isometric encoding map $\mathcal{E}_{L\to P}$ can ***detect errors*** from an error set Err if a measurement of the POVM $\{\Pi, \mathbb{1} - \Pi\}$ can identify the presence of an error without corrupting the logical state:

$$\Pi E|\psi\rangle \propto |\psi\rangle \qquad \forall\ |\psi\rangle \in \mathcal{C},\ \forall\ E \in \mathsf{Err} \ . \tag{2.40}$$

It turns out that a set of conditions characterize quantum error-detecting codes in a way reminiscent of the Knill-Laflamme conditions for quantum error correction.

Theorem 3 (Conditions for a quantum error-detecting code)**.** *An isometric encoding $\mathcal{E}_{L\to P}$ can detect errors from $\mathsf{Err} = \mathrm{span}\{E_k\}$ if and only if $\Pi E_k \Pi \propto \Pi$ for all k.*

Exercise: Prove the above statement.

Error-detecting codes are closely connected to an error model that we'll cover in the next chapter, namely erasure errors, that applies to encodings on a physical space consisting of multiple subsystems.

2.5 Nearly correctable errors

What if an error E' happens that is not strictly in $\mathrm{span}\,\mathsf{Err}$, but which might have high overlap with a correctable error $E \in \mathrm{span}\,\mathsf{Err}$? Intuitively, the code should be able to "correct approximately" E'.

We quantify the distinguishability of two quantum channels $\mathcal{N}, \mathcal{N}'$ with the *diamond distance*

$$\frac{1}{2}\|\mathcal{N}' - \mathcal{N}\|_\diamond = \sup_\sigma \frac{1}{2}\|(\mathcal{N}' \otimes \mathrm{id})(\sigma) - (\mathcal{N} \otimes \mathrm{id})(\sigma)\|_1 \ . \tag{2.41}$$

If $\frac{1}{2}\|\mathcal{N}' - \mathcal{N}\|_\diamond \leq \epsilon$, then we cannot distinguish $\mathcal{N}$ from $\mathcal{N}'$ except with probability $\sim \epsilon$ better than a random guess.

If an encoding $\mathcal{E}_{L\to P}$ corrects a noise channel $\mathcal{N}_{P\to P'}$, then applying the same decoding channel $\mathcal{D}_{P'\to L}$ to another noise channel $\mathcal{N}'_{P\to P'}$ with $\frac{1}{2}\|\mathcal{N}' - \mathcal{N}\|_\diamond \leq \epsilon$ recovers the logical state with an error tolerance $\leq \epsilon$ in trace distance: For any reference system R and for any σ_{LR},

$$\begin{aligned}\frac{1}{2}\|\mathcal{D}\mathcal{N}'\mathcal{E}(\sigma_{LR}) - \sigma_{LR}\|_1 &\leq \frac{1}{2}\|\mathcal{D}\mathcal{N}'\mathcal{E} - \mathrm{id}_L\|_\diamond = \frac{1}{2}\|\mathcal{D}\mathcal{N}'\mathcal{E} - \mathcal{D}\mathcal{N}\mathcal{E}\|_\diamond \\ &\leq \frac{1}{2}\|\mathcal{D}(\mathcal{N}' - \mathcal{N})\mathcal{E}\|_\diamond \leq \frac{1}{2}\|\mathcal{N}' - \mathcal{N}\|_\diamond \leq \epsilon \ ,\end{aligned} \tag{2.42}$$

where we recall that the diamond norm is submultiplicative, $\|\mathcal{N}\mathcal{M}\|_\diamond \leq \|\mathcal{N}\|_\diamond \|\mathcal{M}\|_\diamond$ and that any c.p., t.p. map has diamond norm equal to one. Therefore, any noise channel that is close to a correctable noise channel in diamond norm can only weakly confuse the decoding channel in restoring the logical information.

The diamond distance between noise channels might be cumbersome to compute. Fortuntely, we can employ standard matrix analysis techniques and norm inequalities to bound the diamond norm between two noise channels in terms of the norm of pairs of Kraus operators. Suppose $\mathcal{N}'(\cdot) = \sum_{k=1}^m E'_k (\cdot) E'^\dagger_k$ with some Kraus operators $\{E'_k\}_{k=1}^m$, and suppose that there exists a quantum channel $\mathcal{N}(\cdot) = \sum_{k=1}^m E_k (\cdot) E_k^\dagger$ with $E_k \in \mathrm{span}\,\mathsf{Err}$ and such that for all $k = 1, \ldots, m$, we have $\|E_k - E'_k\|_\infty \leq \epsilon$. Then $\frac{1}{2}\|\mathcal{N}' - \mathcal{N}\|_\diamond \leq 3m\epsilon$.

Exercise: Prove the latter inequality.

It is natural to define a notion of *approximate* quantum error correction, where a degree of imperfection is allowed during the recovery of the logical information. A natural definition, for instance, is the following:

Let $\mathcal{E}_{L\to P}$ be an encoding map and let $\mathcal{N}_{P\to P'}$ be any quantum channel. Let $\epsilon \geq 0$. We say the encoding $\mathcal{E}$ can ***ϵ-approximately protect against*** $\mathcal{N}$ if there exists a quantum channel $\mathcal{D}_{P'\to L}$ such that

$$\|\mathcal{D}\mathcal{N}\mathcal{E} - \mathrm{id}_L\|_\diamond \leq \epsilon \ . \tag{2.43}$$

In this lecture, we'll focus on exact quantum error-correcting codes. It turns out that for common noise models, such as when noise acts independently on each qubit of a n-qubit system, we can replace the noise by a similar noise model for which we can develop exact error-correcting codes. Such considerations enable us to avoid the formalities and technicalities of the framework of approximate error correction.

We'll see, however, that the strict framework of exact quantum error correction is insufficient to develop some of the more important results that we'll cover later on in the context of fault tolerance.

2.6 The environment's perspective

It turns out that quantum information theory gives us a powerful technique to analyze quantum error-correcting codes by studying what information leaks to the environment.

This section is not strictly essential to understanding the remainder of this course, but presents an information-theoretic picture that brings additional clarity to the concept of quantum error correction and to the interpretation of the Knill-Laflamme conditions. As a plus, we'll be extending some of the above concepts to encoding maps that are not isometric.

A remarkable property of processes in quantum information theory is that they can always be viewed as a "part" of some unitary process that involves an additional quantum system. The Stinespring dilation indeed guarantees that any quantum channel $\mathcal{M}_{A\to B}$ can be viewed as an isometric embedding of the states of A onto joint states of B and an additional system E (the "environment"), followed by a partial trace on E. (The isometry can be replaced by a unitary, provided the input environment system is chosen of a suitable dimension and is initialized in a fixed pure state.)

This picture tells us that quantum information cannot be destroyed—it always has to go *somewhere*. If we can't find it at the output of a process, it must have flowed, at least in part, to the environment. Conversely, if the environment E obtains any information about the input state of $\mathcal{M}$, then the no-cloning principle dictates that the state cannot be fully recovered from the output of $\mathcal{M}$.

We've seen that a noise channel $\mathcal{N}_{P\to P'}$ with Kraus operators $\{E_k\}$ can be written in Stinespring form as

$$\mathcal{N}(\cdot) = \mathrm{tr}_E\Big\{W_{P\to P'E}\,(\cdot)\,W^\dagger\Big\}\ ; \qquad\qquad W_{P\to P'E} = \sum E_k \otimes |k\rangle_E\ . \tag{2.44}$$

We can interpret the action of $\mathcal{N}$ as follows: One of the $\{E_k\}$'s is applied on the system, without us knowing which E_k occurred; simultaneously, the information about which E_k occurred is supplied to the environment E. The environment can be thought of an adversary who, by gleaning information about the system P, can ruin our ability to recover that information from the channel's output P'.

Let's now study more specifically how any information learned by the environment can affect our ability to recover information encoded in the physical system. For this, we first consider the channel that describes what information the environment learns from the action of a quantum process.

Given any quantum channel $\mathcal{M}_{A\to B}$, we define a ***complementary channel*** $\widehat{\mathcal{M}}_{A\to E}$ to $\mathcal{M}_{A\to B}$ as any channel that can be expressed in the form

$$\widehat{\mathcal{M}}_{A\to E}(\cdot) = \mathrm{tr}_B\Big\{U_{A\to BE}\,(\cdot)\,U^\dagger\Big\}\,, \tag{2.45}$$

where $U_{A\to BE}$ is any Stinespring dilation isometry of $\mathcal{M}$, i.e., an isometry such that $\mathcal{M}_{A\to B}(\cdot) = \mathrm{tr}_E\{U\,(\cdot)\,U^\dagger\}$.

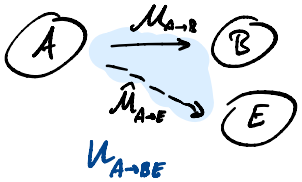
Different choices of a Stinespring dilation isometry give rise to different choices of complementary channels. Complementary channels may also differ in their output system dimensions.

If we fix a Stinespring isometry $U_{A\to BE}$ along with its environment system E, we obtain $\mathcal{M}_{A\to B}$ by tracing out E and we obtain $\widehat{\mathcal{M}}_{A\to E}$ by tracing out B:

$$\mathcal{M}_{A\to B} = \mathrm{tr}_E\Big\{U_{A\to BE}\,(\cdot)\,U^\dagger\Big\}\,; \tag{2.46}$$

$$\widehat{\mathcal{M}}_{A\to E} = \mathrm{tr}_B\Big\{U_{A\to BE}\,(\cdot)\,U^\dagger\Big\}\,. \tag{2.47}$$

The system B is then the Stinespring environment of $\widehat{\mathcal{M}}$:



In the same way that all Stinespring dilations of a channel are equivalent up to a partial isometry on the environment, all complementary channels are equivalent up to a partial isometry applied onto their output.

Exercise: Suppose $\mathcal{M}_{A\to B}(\cdot) = \sum M_k\,(\cdot)\,M_k^\dagger$. *Show that the channel*

$$\widehat{\mathcal{M}}_{A\to E}(\cdot) = \sum_{k,k'} \mathrm{tr}\big(M_{k'}^\dagger M_k\,(\cdot)\big)\,|k\rangle\langle k'| \tag{2.48}$$

is a complementary channel of $\mathcal{M}_{A\to B}$.

The complementary channel describes information that leaks to the environment during the application of a channel. If the channel is unitary, no information leaks to the environment: The complementary channel is a constant channel that always prepares a fixed state. Otherwise, the environment represents all the information required to purify the output of the channel whenever the input is a pure state.

Example: The amplitude damping channel describes the relaxation of a two-level system to its ground state while emitting a photon which is lost to the environment. The complementary channel describes the emission of this photon.

In quantum error correction, we need to ensure the environment does not receive any information about the encoded logical quantum state, or else the no-cloning theorem would jeopardize our ability to recover the encoded state from the output of the noise channel.

Theorem 4 (Conditions for quantum error correction via the complementary noise channel)**.** *Let $\mathcal{E}_{L\to P}$ be any encoding map and let $\mathcal{N}_{P\to P'}$ be any quantum channel. Let $\widehat{[\mathcal{N}\circ\mathcal{E}]}_{L\to\tilde{E}}$ be a complementary channel to $\mathcal{N}\circ\mathcal{E}$, mapping input states on L to an environment system $\tilde{E}$. Are then equivalent:*

(i) The encoding $\mathcal{E}_{L\to P}$ can correct $\mathcal{N}_{P\to P'}$;

(ii) There exists a state $\tau_{\tilde{E}}$ such that

$$\widehat{[\mathcal{N}\circ\mathcal{E}]}_{L\to\tilde{E}}(\cdot) = \mathrm{tr}(\cdot)\,\tau_{\tilde{E}}\ , \tag{2.49}$$

i.e., $\widehat{[\mathcal{N}\circ\mathcal{E}]}_{L\to\tilde{E}}$ is a constant channel that always outputs $\tau_{\tilde{E}}$;

(iii) For all operators $O_{\tilde{E}}$, we have

$$\widehat{[\mathcal{N}\circ\mathcal{E}]}^{\dagger}_{L\to\tilde{E}}(O_{\tilde{E}}) = c(O_{\tilde{E}})\,\mathbb{1}_L\ , \tag{2.50}$$

for some scalar $c(O_{\tilde{E}}) \in \mathbb{C}$;

(iv) For all operators $O_{\tilde{E}}$, and for any operator A_L on $\mathscr{H}_L$, we have

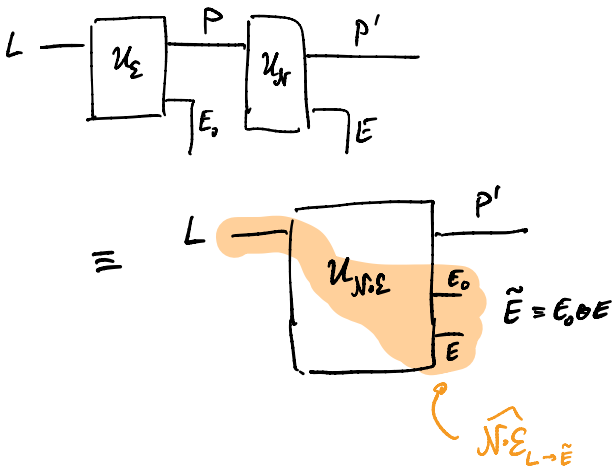
$$\left[\,\widehat{[\mathcal{N}\circ\mathcal{E}]}^{\dagger}_{L\to\tilde{E}}(O_{\tilde{E}})\,,\,A_L\,\right] = 0\ . \tag{2.51}$$

Point (ii) states that the environment receives no information about the state that is input to the encoding map.

Point (iii) makes reference to the Heisenberg picture for the evolution of observables. It states that the environment's observables, when mapped back onto the input through the adjoint of the full complementary map, can only act trivially on the logical system ($\propto \mathbb{1}_L$).

Point (iv) expresses a similar idea: The environment cannot apply any operator on the logical input (via the adjoint of the complementary channel) that can disturb, i.e. fail to commute with, any logical observable.

The complementary channel $\widehat{[\mathcal{N}\circ\mathcal{E}]}_{L\to\tilde{E}}$, if constructed from Stinespring dilations of $\mathcal{E}$ and $\mathcal{N}$, must include both environments present in these dilations:



(In our earlier study of the Knill-Laflamme conditions, we didn't have to worry about the environment of $\mathcal{E}$, since we restricted our attention to isometric encodings.)

The condition in Point (iii) is an extension of the Knill-Laflamme conditions to encoding maps that are not isometric. We recover the Knill-Laflamme conditions (2.22) if $\mathcal{E}$ is isometric: In this case, $\widehat{\mathcal{N}\circ\mathcal{E}} = \widehat{\mathcal{N}}\circ\mathcal{E}$ and $\tilde{E} = E$ consists only of the Stinespring environment of $\mathcal{N}$. We can thus pick

$$\widehat{\mathcal{N}\circ\mathcal{E}}(\cdot) = \widehat{\mathcal{N}}(V\,(\cdot)\,V^\dagger)\ ; \qquad \widehat{\mathcal{N}}(\cdot) = \sum_{k,k'} \mathrm{tr}(E_{k'}^\dagger E_k\,(\cdot))\,|k\rangle\langle k'|_E\ , \tag{2.52}$$

where $\{E_k\}$ are a choice of Kraus operators of $\mathcal{N}$ and V is the encoding isometry. Equivalently, we can write for all k, k',

$$\mathrm{tr}\Big[\widehat{\mathcal{N}}(\cdot)|k'\rangle\langle k|\Big] = \mathrm{tr}(E_{k'}^\dagger E_k\,(\cdot))\ . \tag{2.53}$$

By definition of the adjoint map, and since the equation is true for any operator we choose to plug into $(\cdot)$,

$$\widehat{\mathcal{N}}^\dagger(|k'\rangle\langle k|) = E_{k'}^\dagger E_k\ . \tag{2.54}$$

Applying $V^\dagger(\cdot)V$, we finally see that

$$\widehat{\mathcal{N}\circ\mathcal{E}}(|k'\rangle\langle k|) = V^\dagger\widehat{\mathcal{N}}^\dagger(|k'\rangle\langle k|)V = V^\dagger E_{k'}^\dagger E_k V\ . \tag{2.55}$$

Now let's inspect the condition given in Point (iii). Because $|k'\rangle\langle k|_E$ is a basis of all operators on $\tilde{E} = E$, the condition (iii) reduces using (2.55) to

$$V^\dagger E_{k'}^\dagger E_k V = c(|k'\rangle\langle k|)\,\mathbb{1}_L\ , \tag{2.56}$$

which is precisely the Knill-Laflamme conditions we derived earlier.

Proof of Theorem 4. *(i)⇒(ii)*: Let $\tilde{W}_{L\to P'\tilde{E}}$ denote a Stinespring dilation isometry of $\mathcal{N}\circ\mathcal{E}$, such that

$$[\mathcal{N}\circ\mathcal{E}]_{L\to P'}(\cdot) = \mathrm{tr}_{\tilde{E}}\{\tilde{W}\,(\cdot)\,\tilde{W}^\dagger\}\ ; \tag{2.57}$$

$$[\widehat{\mathcal{N}\circ\mathcal{E}}]_{L\to\tilde{E}}(\cdot) = \mathrm{tr}_{P'}\{\tilde{W}\,(\cdot)\,\tilde{W}^\dagger\}\ . \tag{2.58}$$

By assumption, there exists a quantum channel $\mathcal{D}_{P'\to L}$ such that $\mathcal{D}\circ\mathcal{N}\circ\mathcal{E} = \mathrm{id}_L$; let $K_{P'\to LF}$ be a Stinespring dilation of $\mathcal{D}$ with

$$\mathcal{D}_{P'\to L}(\cdot) = \mathrm{tr}_F\{K_{P'\to LF}\,(\cdot)\,K^\dagger\}\ . \tag{2.59}$$

With $|\Phi\rangle_{L:R} = \sum|i\rangle_L|i\rangle_R$, we have

$$\Phi_{L:R} = [\mathcal{D}\circ\mathcal{N}\circ\mathcal{E}](\Phi_{L:R}) = \mathrm{tr}_{\tilde{E}F}\{K_{P'\to LF}\tilde{W}_{L\to\tilde{E}P'}\,\Phi_{L:R}\,\tilde{W}^\dagger K^\dagger\}\ , \tag{2.60}$$

and therefore there exists some pure quantum state $|\chi\rangle_{\tilde{E}F}$ such that

$$K_{P'\to LF}\tilde{W}_{L\to\tilde{E}P'}\,\Phi_{L:R}\,\tilde{W}^\dagger K^\dagger = \Phi_{L:R}\otimes\chi_{\tilde{E}F}\ . \tag{2.61}$$

Then

$$\begin{aligned}\widehat{\mathcal{N}\circ\mathcal{E}}(\Phi_{L:R}) &= \mathrm{tr}_{P'}\{\tilde{W}_{L\to P'\tilde{E}}\,\Phi_{L:R}\,\tilde{W}^\dagger\}\\ &= \mathrm{tr}_{LF}\{K_{P'\to LF}\tilde{W}_{L\to P'\tilde{E}}\,\Phi_{L:R}\,\tilde{W}K^\dagger\}\\ &= \mathrm{tr}_{LF}\{\Phi_{L:R}\otimes\chi_{\tilde{E}F}\} = \mathbb{1}_R\otimes\chi_{\tilde{E}}\ ,\end{aligned} \tag{2.62}$$

where we inserted $\mathbb{1}_{P'} = K^\dagger K$ inside the trace to obtain the second equality.

We identify in (2.62) the Choi matrix of the channel that always outputs $\chi_{\tilde{E}}$, regardless of its input; i.e.,

$$\widehat{\mathcal{N}\circ\mathcal{E}}(\cdot) = \mathrm{tr}(\cdot)\,\chi_{\tilde{E}}\ . \tag{2.63}$$

(ii)⇔*(iii)*⇔*(iv)*: The adjoint of the channel $\mathcal{T}_\tau(\cdot) = \mathrm{tr}(\cdot)\,\tau$ is $\mathcal{T}_\tau^\dagger(\cdot) = \mathrm{tr}[\tau\,(\cdot)]\,\mathbb{1}$ and *vice versa*. Furthermore, $\mathbb{1}$ commutes with all operators, and is the only operator with this property.

(iv)⇒*(i)*: We assume $\widehat{\mathcal{N}\circ\mathcal{E}} = \mathrm{tr}(\cdot)\,\tau_{\tilde{E}}$. Then

$$\widehat{\mathcal{N}\circ\mathcal{E}}(\Phi_{L:R}) = \mathbb{1}_R\otimes\tau_{\tilde{E}}\ . \tag{2.64}$$

We can purify this nonnormalized state on the system L (recall $\mathscr{H}_L\simeq\mathscr{H}_R$) along with another large enough system F, as:

$$\widehat{\mathcal{N}\circ\mathcal{E}}(\Phi_{L:R}) = \mathrm{tr}_{\tilde{E}F}\{|\Phi\rangle\langle\Phi|_{L:R}\otimes|\tau\rangle\langle\tau|_{\tilde{E}F}\}\ , \tag{2.65}$$

where $|\tau\rangle_{\tilde{E}F}$ is a purification of $\tau_{\tilde{E}}$.

Another purification of (2.64) is

$$\widehat{\mathcal{N}\circ\mathcal{E}}(\Phi_{L:R}) = \mathrm{tr}_{P'}\{\tilde{W}_{L\to P'\tilde{E}}\,\Phi_{L:R}\,\tilde{W}^\dagger\}\ , \tag{2.66}$$

where $\tilde{W}_{L\to P'\tilde{E}}$ is the Stinespring dilation of $\mathcal{N}\circ\mathcal{E}$ that served to define the complementary channel $\widehat{\mathcal{N}\circ\mathcal{E}}$.

Two purifications of the same state are always related by a partial isometry on the purifying systems: There must exist a partial isometry $U_{P'\to LF}$ such that

$$|\Phi\rangle\langle\Phi|_{L:R}\otimes|\tau\rangle\langle\tau|_{\tilde{E}F} = U_{P'\to LF}\,\tilde{W}_{L\to P'\tilde{E}}\,|\Phi\rangle\langle\Phi|_{L:R}\,\tilde{W}^\dagger U^\dagger\ . \tag{2.67}$$

As long as we choose F large enough, $U_{P'\to LF}$ can be completed to an isometry by adding terms that send the kernel of U isometrically to some subspace that is orthogonal to the image of U. Denote by $U'_{P'\to LF}$ the isometry obtained in this fashion.

We may now define $\mathcal{D}_{P'\to L}$ as the quantum channel whose Stinespring dilation isometry is $U'_{P'\to LF}$:

$$\mathcal{D}_{P'\to L}(\cdot) = \mathrm{tr}_F\{U'_{P'\to LF}\,(\cdot)\,U'^\dagger\}\ . \tag{2.68}$$

We find

$$\begin{aligned}\mathcal{D}\circ\mathcal{N}\circ\mathcal{E}(\Phi_{L:R}) &= \mathrm{tr}_{\tilde{E}F}\{U'_{P'\to LF}\,\tilde{W}_{L\to P'\tilde{E}}\,\Phi_{L:R}\,\tilde{W}^\dagger U'^\dagger\}\\ &= \mathrm{tr}_{\tilde{E}F}\{\Phi_{L:R}\otimes\tau_{\tilde{E}F}\} = \Phi_{L:R}\ .\end{aligned} \tag{2.69}$$

Therefore $\mathcal{D}\circ\mathcal{N}\circ\mathcal{E} = \mathrm{id}_L$, as claimed. ∎

This theorem offers an appealing approach to characterize the setting of *approximate* quantum error correction introduced in Section 2.5. Points (i) and (ii) can be made approximate in order to find conditions for approximate quantum error correction that resemble the Knill-Laflamme conditions. Namely, we can show there exists $\mathcal{D}$ such that $\mathcal{D} \circ \mathcal{N} \circ \mathcal{E} \approx \mathrm{id}_L$ if and only if $\widehat{\mathcal{N} \circ \mathcal{E}} \approx \mathcal{T}$ for some channel $\mathcal{T}$ that traces out its output and always outputs a fixed state (a constant channel). The approximation symbols '$\approx$' refer to proximity measured by a suitable distance on channels.

2.7 Further reading

I recommended reading of course Chapters 1 and 2 in Gottesman's book for a detailed construction of the theory of quantum error correction. Another strongly recommended reference, which also influenced the writing of this chapter, is John Preskill's lecture notes [3]. The fundamental theory of quantum error correction is also presented in the various references listed in the "Recommended literature" section in the preface, including in the Nielsen and Chuang textbook [4].

The approach to defining quantum error correction presented here centers around the ability to reverse the effect of noise channels; it differs slightly from the approach presented in Gottesman's book [1], which focuses on the ability to correct sets of operators. As presented in Gottesman's book, a quantum error-correcting code can be defined as an encoding isometry along with a set of error operators such that the errors are correctable.

Some original papers in which the fundamental theory of quantum error correction was developed can be insightful and give useful context, see for instance Knill and Laflamme's original paper [24].

A quantum information-theoretic approach is particularly useful to understand how the environment can affect our ability to reverse the effect of a quantum noise channel [25–27]. Quantum error correction can also be characterized in the picture of operator algebras [28, 29].

Chapter 3

Fundamental Theory of Quantum Error Correction II: Encoding information on multiple subsystems

This chapter is a continuation of the previous chapter on the fundamental concepts of quantum error correction. The definitions and concepts we present here apply to the case of encodings that involve n subsystems and that are designed to correct local errors. This is still a very general setting, and most of the course will refer to this scenario.

Specifically, we make the following assumptions in this chapter:

- The logical space L consists of k particles, where each particle is a q-dimensional system, such that $\mathscr{H}_L \simeq \mathbb{C}^{q^k}$;
- The physical space P consists of n particles, where each particle is a q-dimensional system, such that $\mathscr{H}_P \simeq \mathbb{C}^{q^n}$;
- Unless specified otherwise, the encoding maps $\mathcal{E}_{L\to P}$ we consider are *isometric*, with encoding isometry that we denote by $V_{L\to P}$. As we've seen in the previous chapter, such encodings have a well-defined notion of a code space $\mathcal{C}$ and code space projector $\Pi = VV^\dagger$.

In summary, an isometric encoding maps the logical space, which consists of k qudits of local dimension q, to the physical space, which consists of n such qudits. In the special case $q = 2$, the subsystems are simply qubits. Each subsystem is referred to as a "particle," to emphasize its role as an elementary constituent of the logical and physical spaces. You are welcome to mentally replace "particle" by "qubit" in this chapter, though the more general terminology will help us remember these concepts apply to $q \neq 2$ as well.

This setting is very natural in quantum error correction and, apart from the notable exception of bosonic codes, encompasses most of the effort in developing quantum error correction for quantum computers. Most of the time, we'll think of the particles as hardware qubits, which might be laid out on a chip with a suitable connectivity enabling two-qubit gates to be applied on neighboring qubits.

The concepts in this chapter naturally extend to the case where each individual logical and physical particle has a different local Hilbert space dimension. Perhaps we might be interested in encoding a combined logical qubit and a logical qutrit in a physical space consisting of five qubits, two qutrits, and four 10-dimensional qudits. For simplicity and to avoid tedious notation, we present these concepts in the case where each particle has the same dimensionality q.

3.1 Weights of errors

Suppose that each physical particle is subject to a noise channel $\mathcal{N}_1$ independently of the other particles, with

$$\mathcal{N}_1(\cdot) = A_0\,(\cdot)\,A_0^\dagger + \sum_{k=1}^{m} A_k\,(\cdot)\,A_k^\dagger\ ; \tag{3.1}$$
$$A_0 \approx \mathbb{1}\ ; \quad A_k = O(\sqrt{p}) \text{ for } k = 1, \ldots, m\ ,$$

where $A_0, A_1, \ldots, A_m$ are the Kraus operators of $\mathcal{N}_1$ which depend on a parameter p that parametrizes the intensity of the noise. We assume that $p \approx 0$ and that $\mathcal{N}_1$ equals the identity channel if $p = 0$.

The n particles are then subject to the noise channel $\mathcal{N}_1 \otimes \mathcal{N}_1 \otimes \cdots \otimes \mathcal{N}_1 \equiv \mathcal{N}_1^{\otimes n}$. The Kraus operators of the global channel can be given as:

$$\mathcal{N}_1^{\otimes n}(\cdot) = \sum_{\boldsymbol{x} \in \{0,\ldots,m\}^{\times n}} E_{\boldsymbol{x}}\,(\cdot)\,E_{\boldsymbol{x}}^\dagger\ ; \qquad E_{\boldsymbol{x}} \equiv \bigotimes_{i=1}^{n} A_{x_i}\ . \tag{3.2}$$

I.e., the global Kraus operators are indexed by a string $\boldsymbol{x}$ that indicates which Kraus operator to apply on which particle.

A state (or channel) of the form $\rho^{\otimes n}$ (or $\mathcal{N}_1^{\otimes n}$) is called an ***independent and identically distributed (i.i.d.)*** state (or channel), marking the fact that the state or channel acts independently and in the same way on each of n copies of a system. The Kraus operators associated with an i.i.d. channel are of the form given in (3.2).

Example: Suppose that $\mathcal{N}_1(\cdot) = (1-p)\,\mathbb{1}\,(\cdot)\,\mathbb{1} + p\,Z\,(\cdot)\,Z$. Specifically, $A_0 = \sqrt{1-p}\,\mathbb{1}$ and $A_1 = \sqrt{p}\,Z$ with $m = 1$. Then $\mathcal{N}_1^{\otimes n}$ has Kraus operators:

We see that, in general, $E_{\boldsymbol{x}}$ consists of some number w of Kraus operators that are nontrivial (indexed by $x_i \neq 0$) and $n - w$ copies of A_0. This observation motivates the definition of the *weight* of the string $\boldsymbol{x}$:

The ***weight*** $\mathrm{wgt}(\boldsymbol{x})$ of the string $\boldsymbol{x} \in \{0, \ldots, m\}^{\times n}$ is defined as the number of locations i for which $x_i \neq 0$.

Since $E_{\boldsymbol{x}}$ contains a product of $\mathrm{wgt}(\boldsymbol{x})$ Kraus operators with $x_i \neq 0$, we see that $E_{\boldsymbol{x}} = O(\sqrt{p^{\mathrm{wgt}(x)}})$. This implies that the probability of the error $E_{\boldsymbol{x}}$ occurring is suppressed exponentially in $\mathrm{wgt}(\boldsymbol{x})$:

$$\Pr[E_{\boldsymbol{x}}] = O(p^{-\,\mathrm{wgt}(\boldsymbol{x})})\ . \tag{3.3}$$

Instead of having to deal with all possible errors $\{E_{\boldsymbol{x}}\}$, it is natural to focus on only those errors with $\mathrm{wgt}(\boldsymbol{x}) \leq t$ for some fixed $t > 0$. The errors we leave out have a probability of occurring that is exponentially suppressed in t (at fixed n). Furthermore, the operator $E_{\boldsymbol{x}}$ acts nontrivially essentially on $\mathrm{wgt}(\boldsymbol{x})$ particles only, since $A_0 \approx \mathbb{1}$. We'll see in a moment how this observation significantly simplifies the analysis of the error-correcting properties of the code.

In other words, we start from the noise channel (3.2) and we make two approximations. First, we ignore Kraus operators $E_{\boldsymbol{x}}$ with $\mathrm{wgt}(\boldsymbol{x}) \leq t$, which have a probability of occurring that is exponentially suppressed in t. Second, we replace each remaining $E_{\boldsymbol{x}}$ by a nearby operator that acts nontrivially only on at most t particles and that acts as the identity on all remaining particles.

Formally, the step of replacing the i.i.d. noise channel of (3.2) by a noise channel with Kraus operators that act nontrivially on exclusively $\leq t$ particles must be accompanied by an upper bound on the diamond distance between the original noise channel and the alternative noise channel (cf. Section 2.5). We refer to Ref. [1, Theorem 1.1] for the detailed derivation of such a bound.

We'll proceed with this approximation for now, which is a foundation on which important concepts of n-particle quantum error-correcting codes rely on. I'll sneak in a word of caution though. We argued that we can ignore errors $E_{\boldsymbol{x}}$ whose probability are exponentially suppressed in some parameter t. But what if there are exponentially many possible errors with this property? This is typically the case for i.i.d. noise if we scale to large n. In such a case, the probability of *some* weight-w error might remain significant, despite the probability of an individual error decaying exponentially in w; the approximation above might be inaccurate. We'll revisit this issue when we start discussing concepts of fault tolerance.

One of the happy twists of quantum error correction is that it turns out to be possible to design good general-purpose codes that can not only correct errors $E_{\boldsymbol{x}}$ with low-weight $\boldsymbol{x}$ of a fixed noise model, but that can also simultaneously correct such errors for *any* noise model with the structure of the form (3.2). Intuitively, designing good n-particle codes means being able to correct arbitrary errors that act on few ($\leq t$) particles. This idea is formalized through the concepts of the *weight* of an operator and the *distance* of a code.

The ***weight*** $\mathrm{wgt}(E)$ of an operator E acting on P is the number of particles on which E acts nontrivially. Specifically, it is the size $|I|$ of the smallest set I of indices such that

$$E = E_I \otimes \mathbb{1}_{I^c} \ , \tag{3.4}$$

where E_I acts on the particles indexed by I and where I^c is the complementary set of I in $\{1, 2, \ldots, n\}$.

For example, $\mathrm{wgt}(X \otimes X \otimes \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes Y) = 4$ and $\mathrm{wgt}(\mathbb{1}_1 \otimes O_{23} \otimes \mathbb{1}_4 \otimes Z_5) = 3$.

Thanks to the linearity of quantum error correction (Theorem 1), the ability to correct all errors of weight at most t automatically implies the ability to correct any linear combination of operators of weight at most t.

A linear combination of low-weight operators can have a high weight according to the definition above. But such linear combinations still very much behave like low-weight operators for the purposes of quantum error correction, as correctability of the latter implies correctability of the former. Such linear combinations are markedly different from most high-weight operators (such as $X \otimes X \otimes \cdots \otimes X$) which, generically, cannot be written as linear combinations of low-weight operators.

For $t \geq 0$, we define a ***t-particle error*** as any linear combination of operators of weight at most t. Specifically, E is a t-particle error if

$$E = \sum_j A_j \ ; \qquad\qquad \mathrm{wgt}(A_j) \leq t \quad \forall \ j \ . \tag{3.5}$$

Most quantum error-correcting codes we'll study in this course are designed to correct the set of all t-particle errors

$$\mathsf{Err}_t = \left\{ E \ : \ E = \sum A_j \, , \ \mathrm{wgt}(A_j) \leq t \ \ \forall \ j \right\} . \tag{3.6}$$

3.2 Code distance

In order to quantify the ability of a code to protect against errors, we introduce the notion of *distance* of the code. The distance of a code tells us "how large" an operator we need to apply on the physical system in order to effect a nontrivial logical action on the code space.

We define the ***distance*** d of a n-particle code as the minimal weight $\mathrm{wgt}(E)$ of an operator E that satisfies $\Pi E \Pi \not\propto \Pi$:

$$d = \min\{ w \ : \ \exists \ E \text{ with } \ \mathrm{wgt}(E) = w \text{ and } \Pi E \Pi \not\propto \Pi \} \ . \tag{3.7}$$

A quantum code that encodes k qubits into n qubits (setting $q = 2$) and that has a distance d is called a $[[n, k, d]]$ ***code***.

This notation is sometimes extended to the case $q \neq 2$: A code that encodes k qudits into n qudits is somtimes designated as a $[[n, k, d]]_q$ *code* or a $((n, q^k, d))_q$ *code*. Furthermore, in some references the notation $[[n, k, d]]$ is reserved for qubit *stabilizer* codes, which we'll introduce in the next chapter.

Theorem 5. *A n-particle code can correct the set Err_t of all t-particle errors if and only if its distance d satisfies*

$$d \geq 2t + 1 \ . \tag{3.8}$$

Proof. Let $\{E_k\}$ be a set of operators with $\mathrm{wgt}(E_k) \leq t$ and such that $\mathsf{Err}_t = \mathrm{span}\{E_k\}$. Consider an operator of the form $E_{k'}^\dagger E_k$. Its weight is at most $2t$, since each E_k acts on at most t particles nontrivially.

If $d \geq 2t + 1$, then $2t < d$ and $\Pi E_{k'}^\dagger E_k \Pi \propto \Pi$ by definition of the distance, so the Knill-Laflamme conditions are satisfied.

Conversely, suppose the Knill-Laflamme conditions are satisfied. Let's first pick any tensor product basis of the $2t$-particle operator space, i.e., a basis whose elements are all tensor products across the $2t$ particles. (The existence of such a basis follows from the definition of the tensor product. For qubits, we can also choose the Pauli basis). Any such basis element can be viewed as a product of two operators of weight at most t, and can thus be written as a linear combination of operators of the form $E_{k'}^\dagger E_k$. Now let E be any operator of weight at most $2t$. By decomposing E in such a basis, we see that E can be written out as a linear combination of operators of the form $E_{k'}^\dagger E_k$. By the Knill-Laflamme conditions, we find that $\Pi E \Pi \propto E$. In other words, for an operator E' to satisfy $\Pi E' \Pi \not\propto \Pi$, we must have $\mathrm{wgt}(E') > 2t$. The distance must therefore satisfy $d \geq 2t + 1$. ∎

Exercise: Show that a n-particle code can detect errors up to weight $d-1$, i.e., it can detect errors from the error set Err_{d-1}.

3.3 Erasure errors

An ***erasure channel*** $\mathcal{N}_{\text{Eras. }I}$ traces out a subset I of the n particles, and replaces their state by some fixed state χ_I:

$$\mathcal{N}_{\text{Eras. }I}(\cdot) = \text{tr}_I\{(\cdot)\} \otimes \chi_I \ . \tag{3.9}$$

Such a channel can model the loss of a particle carrier of quantum information, such as a photon, along with the information it was carrying. We assume that we can measure the presence or absence of the particle without disturbing any carried information. As it turns out, the knowledge that a carrier is lost provides an advantage in restoring the encoded logical information as opposed to an error that happens surreptitiously.

⚠ **In the case of erasure errors, we have absolute knowledge about *exactly which particles* are affected by the erasure.** The rather naive channel $\mathcal{N}_{\text{Eras. }I}$ in (3.9) even corresponds to the case where the systems that will be lost are entirely predetermined and are represented by I.

A more interesting setting is where the subset I of particles that are erased is chosen at random. While we don't know in advance which particle(s) will be erased, we still assume that we are given full knowledge of which particles are affected by the erasure after the erasure happens. This knowledge can be modeled explicitly by having the channel produce a state on an additional memory register, storing the labels of the erased particles. We call such a channel an ℓ-particle erasure channel.

An ***ℓ-particle erasure channel*** is a channel of the form

$$\mathcal{N}_{\text{Eras. }\ell}(\cdot) = \sum_{\substack{I \subset \{1,\dots,n\} \\ |I| \leq \ell}} p_I \ \text{tr}_I\{(\cdot)\} \otimes \chi_I \otimes |\text{“}I\text{”}\rangle\langle\text{“}I\text{”}|_X \ , \tag{3.10}$$

where $\{p_I\}$ is a probability distribution, χ_I is any fixed state on the particles indexed by I, and X is an additional classical memory register. (Note that the sum only involves terms with $|I| \leq \ell$.)

An ℓ-particle erasure channel will formally have an output space P' that is larger than P, given that it has to store the information about which particles are erased. Also, we'll occasionally assume in (3.9) and in (3.10) that χ_I is a pure state, although this is mostly an issue of technical convenience.

Another common method to model the knowledge of which particles suffered an erasure is to embed the q-dimensional particle's state space into a $(q+1)$-dimensional space that contains an additional state $|\text{ERASED}\rangle$. Whenever the particle is erased, it is reset to the state $|\text{ERASED}\rangle$. This corresponds to setting $|\chi_I\rangle = |\text{ERASED}\rangle_I^{\otimes |I|}$ in (3.10). A subsequent measurement of each particle according to the POVM $\{|\text{ERASED}\rangle\langle\text{ERASED}|, \mathbb{1} - |\text{ERASED}\rangle\langle\text{ERASED}|\}$ reveals exactly which particles suffered an erasure, without compromising the state of the unaffected particles. In this case, the memory register X in (3.10) provides no additional information and can be omitted.

A $(\ell = 1)$-particle erasure channel can be represented in terms of Kraus operators by resolving the partial trace on the i-th particle explicitly as $\text{tr}_i(\cdot) = \sum_j \langle j|_i \, (\cdot) \, |j\rangle_i$. The

resulting error operators are

$$E_{i,j} \propto |\chi\rangle_i\langle j|_i \otimes \mathbb{1}_{\setminus\{i\}} \otimes |\text{“}i\text{”}\rangle_X \qquad \text{or} \qquad E_{i,j} \propto |\text{ERASED}\rangle_i\langle j|_i \otimes \mathbb{1}_{\setminus\{i\}} \ , \tag{3.11}$$

where $E_{i,j}$ acts on particle i, where $j = 0, \ldots, q-1$, where $|\chi\rangle_i$ is some fixed state on particle i, and where $\mathbb{1}_{\setminus\{i\}}$ is the identity operators on all particles except for the ith particle. (We've assumed that χ_I in Eq. (3.10) is a pure state. If that's not the case, additional Kraus operators are needed to resolve χ_I as an ensemble of pure states.) The two possibilities in (3.11) correspond to whether the information of which particles are erased is modeled explicitly in a register X or whether the affected particles are placed in the additional state $|\text{ERASED}\rangle$. Similarly, the error operators associated with an ℓ-particle erasure channel ($\ell > 1$) can be chosen as

$$E_{I;\boldsymbol{y}} \propto |\chi\rangle_I\langle \boldsymbol{y}|_I \otimes \mathbb{1}_{\setminus I} \otimes |\text{“}I\text{”}\rangle_X \quad \text{or} \quad E_{I;\boldsymbol{y}} \propto |\text{ERASED}\rangle_I^{\otimes|I|}\langle \boldsymbol{y}|_I \otimes \mathbb{1}_{\setminus I} \ , \tag{3.12}$$

where $I \subset \{1, \ldots, n\}$ with $|I| \leq \ell$, where $\boldsymbol{y} \in \{0, \ldots q-1\}^{\times|I|}$ and $\mathbb{1}_{\setminus I}$ is the identity on all particles not indexed by I,

Erasure errors have a simple representation from the environment's perspective:

$$\mathcal{N}_{\text{Eras. }I}(\cdot) = \text{tr}_I(\cdot) \otimes |\chi\rangle\langle\chi|_I \quad \to \quad \widehat{\mathcal{N}}_{\text{Eras. }I}(\cdot) = \text{tr}_{\setminus I}(\cdot) \ , \tag{3.13}$$

where $\text{tr}_{\setminus I}$ denotes the partial trace of all n particles except those labeled by I. The complementary channel thus reveals the reduced state of the input state on the particles affected by the erasure. **The erasure of some particles amounts to handing those particles over to the environment.**

If multiple constellations of erasures appear with corresponding probabilities, as modeled by an ℓ-particle erasure channel (3.10), we find

$$\widehat{\mathcal{N}}_{\text{Eras. }\ell}(\cdot) = \sum_{\substack{I \subset \{1,\ldots,n\} \\ |I| \leq \ell}} p_I \, \text{tr}_{\setminus I}(\cdot) \otimes |I\rangle\langle I|_{X'} \ . \tag{3.14}$$

In this case, the information about which particles are erased is provided explicitly both to the output (via the memory register X) as well as to the environment (via another memory register X').

We have a relation between the *number of particle erasures* a code can correct and its *distance* d:

Theorem 6 (Distance and correction of erasures)**.** *A code can correct the family of all ℓ-particle erasure channels if and only if*

$$d \geq \ell + 1 \ . \tag{3.15}$$

Proof. Consider the error operators spanned by the error operators given in (3.12). Observe that $E_{I',\boldsymbol{y}'}^\dagger E_{I,\boldsymbol{y}} = 0$ whenever $I \neq I'$, because $\langle\text{“}I'\text{”}|\text{“}I\text{”}\rangle_X = \delta_{I,I'}$. (Alternatively, because $|\text{ERASED}\rangle$ is orthogonal to all other particle states, and so they must appear at the same locations in $E_{I,\boldsymbol{y}}$ and $E_{I',\boldsymbol{y}'}$.) If $I = I'$, this means the error operators $E_{I,\boldsymbol{y}}$ and $E_{I',\boldsymbol{y}'}$ both act nontrivially on the same particles $I = I'$, and so $\text{wgt}(E_{I',\boldsymbol{y}'}^\dagger E_{I,\boldsymbol{y}}) = \text{wgt}(E_{I,\boldsymbol{y}}) = \text{wgt}(E_{I',\boldsymbol{y}'}) \leq \ell$. If $\ell < d$, then by definition of the distance, we have $\Pi E_{I',\boldsymbol{y}'}^\dagger E_{I,\boldsymbol{y}} \Pi \propto \Pi$;

the Knill-Laflamme conditions are therefore satisfied and the code can correct the set of all ℓ-particle erasures.

For the converse, we begin by noting that $E_{I,\boldsymbol{y}}^\dagger E_{I,\boldsymbol{y}'} \propto |\boldsymbol{y}\rangle\langle\boldsymbol{y}'|_I \otimes \mathbb{1}_{\setminus I}$. For fixed I and for all $\boldsymbol{y}, \boldsymbol{y}' \in \{0, \ldots, q-1\}^{\times |I|}$, these operators form a basis of all operators acting on the particles I. The collection of all such operators for all $I \subset \{1, \ldots, n\}$ with $|I| \leq \ell$ thus spans all operators of weight at most ℓ. If the $\{E_{I,\boldsymbol{y}}\}$ are correctable, the Knill-Laflamme conditions state that

$$\Pi E_{I,\boldsymbol{y}'}^\dagger E_{I,\boldsymbol{y}} \Pi \propto \Pi \qquad \forall\ |I| \leq \ell, \forall\ \boldsymbol{y}, \boldsymbol{y}'\ . \tag{3.16}$$

Any operator O of weight at most ℓ thus satisfies $\Pi O \Pi \propto \Pi$, as it can be decomposed as a linear combination of the $E_{I,\boldsymbol{y}'}^\dagger E_{I,\boldsymbol{y}}$. By definition, the code must have a distance $d > \ell$. ∎

It is therefore possible to correct more errors if they are erasure errors as opposed to them being more general errors ($2t$ erasures as opposed to t general errors). This property is reminiscent of the difference between *correcting* errors and detecting them, and indeed, both error detection and correction of erasures are possible under the same condition on the distance.

The key reason why a code can correct more erasure errors than general errors is that we are given the knowledge of the *location* of the error. More generally: *A code that can correct t errors at unknown locations can correct $2t$ errors at known locations (a.k.a.* located errors*).* This statement can be seen immediately if we model explicitly the knowledge of the location of an error. For an error E_k that acts nontrivially on particles I, the same error accompanied by location information on a register X is

$$E_k^{\text{loc.}} = E_k \otimes |\text{“}I\text{”}\rangle_X\ . \tag{3.17}$$

In general, the operator $E_{k'}^\dagger E_k$ need not vanish when $E_k, E_{k'}$ act on different sets of particles. But tagging on the location information ensures the errors become orthogonal whenever they act on different sets of particles: $E_{k'}^{\text{loc.}\,\dagger} E_k^{\text{loc.}} = 0$ if $I \neq I'$. Whereas in general, $\text{wgt}(E_{k'}^\dagger E_k) \leq \text{wgt}(E_{k'}^\dagger) + \text{wgt}(E_k)$, for located errors we actually have $\text{wgt}(E_{k'}^{\text{loc.}\,\dagger} E_k^{\text{loc.}}) = \text{wgt}(E_{k'}) = \text{wgt}(E_k)$ or $E_{k'}^{\text{loc.}\,\dagger} E_k^{\text{loc.}} = 0$. We can therefore consider errors acting on more particles before the weight of the error pair reaches the distance of the code:

$$\begin{aligned} &\Pi\ \underbrace{E_{k'}^\dagger E_k}_{\text{wgt}(\cdot) \leq 2t < d}\ \Pi \propto \Pi \quad \forall \text{ weight-}t \text{ errors} \\ &\qquad\qquad \Longrightarrow \quad \Pi \underbrace{E_{k'}^{\text{loc.}\,\dagger} E_k^{\text{loc.}}}_{\text{wgt}(\cdot) \leq 2t < d} \Pi \propto \Pi \quad \forall \text{ weight-}2t \text{ errors.} \end{aligned} \tag{3.18}$$

3.4 Entanglement properties and bounds on code parameters

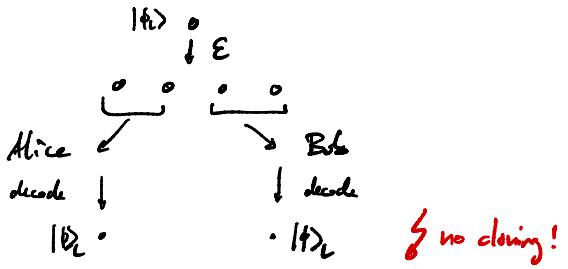
Code words of n-particle codes must be suitably entangled. We can verify this property by computing the reduced state of a code word on a set of sites $I \subset \{1, \ldots, n\}$, with $|I| \leq d-1$ where d is the code distance.

$$\rho_I = \text{tr}_{\setminus I}(|\psi\rangle\langle\psi|) \ :\quad |\psi\rangle \in \mathcal{C}\ . \tag{3.19}$$

We've seen that $\rho_I = \widehat{\mathcal{N}}_{\text{Eras.}\,I}(\psi)$ is what the environment receives if we erase the particles I, so ρ_I better not depend on ψ.

Recall that $\langle \boldsymbol{y}|\rho_I|\boldsymbol{y}'\rangle = \mathrm{tr}(E^\dagger_{I,\boldsymbol{y}'}E_{I,\boldsymbol{y}}\psi)$. Often, the many errors $E_{I,\boldsymbol{y}}$ can all happen on the system, resulting in a ρ_I that is typically well mixed. This, in turn, means that $|\psi\rangle$ must be correspondingly entangled. For nondegenerate stabilizer codes that we'll introduce in the next chapter, it even turns out that $\rho_I = \mathbb{1}_I/2^{|I|}$. Code words of such codes are maximally entangled between any subsystem I and its complement!

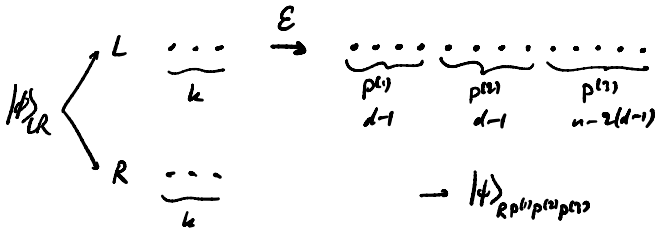
The **no-cloning bound** provides a constraint relating the number n of particles to the distance d of the code. Suppose we had a four-qubit code of distance $d = 3$. Say we send the first two qubits to Alice and the last two to Bob. The code corrects two erasures, so both parties could recover the original logical state, violating the no-cloning theorem:



We find a general bound for any n-particle code (***no-cloning bound***):

$$n > 2(d-1)\ . \tag{3.20}$$

The **quantum Singleton bound** is a refinement of the no-cloning bound that involves the number of encoded logical particles. Consider a pure, maximally entangled state $|\phi\rangle_{LR} = q^{-k/2}\sum_i |i\rangle_L|i\rangle_R$ between the k logical particles $\mathscr{H}_L \simeq \mathbb{C}^{q^k}$ and the reference system $R \simeq L$. The k local particles are encoded into the n physical particles ($\mathscr{H}_P \simeq \mathbb{C}^{q^n}$). Let's split the n particles into three subsystems $P^{(1)}$, $P^{(2)}$, and $P^{(3)}$, where $P^{(1)}$ and $P^{(2)}$ each consist of $d-1$ particles and where $P^{(3)}$ collects the remaining $n-2(d-1)$ particles. We obtain the state $|\psi\rangle_{RP^{(1)}P^{(2)}P^{(3)}}$ that can be visualized as follows:



The local reduced state of $|\psi\rangle$ on R coincides with that of $|\phi\rangle$ on R, since R is not affected by the encoding, and so its von Neumann entropy is that of a maximally mixed state:

$$H(R) = \log_2(q^k) = k\,\log_2(q)\ . \tag{3.21}$$

(All von Neumann entropies in this argument apply to the state $|\psi\rangle_{RP^{(1)}P^{(2)}P^{(3)}}$.) Since $P^{(1)}$ contains $d-1$ particles, its erasure is correctable. The reduced state on $RP^{(1)}$ must therefore be tensor product between R and $P^{(1)}$, since the reduced state on $P^{(1)}$ cannot depend on the logical input state:

$$\rho_{RP^{(1)}} := \mathrm{tr}_{P^{(2)}P^{(3)}}(\psi) = \rho_R \otimes \rho_{P^{(1)}}\ . \tag{3.22}$$

(More precisely: $P^{(1)}$ is the output of the complementary channel of the channel that erases $P^{(1)}$, which is correctable; it must hence be a constant channel that always outputs a fixed state regardless of its input.) Recall the von Neumann entropy is additive for tensor product states:

$$H(RP^{(1)}) = H(R) + H(P^{(1)}) \ . \tag{3.23}$$

Recall furthermore that the von Neumann entropy depends only on the spectrum of its argument; because $|\psi\rangle$ is pure, the Schmidt decomposition guarantees the reduced states on $RP^{(1)}$ and on $P^{(2)}P^{(3)}$ have the same spectrum:

$$H(RP^{(1)}) = H(P^{(2)}P^{(3)}) \ . \tag{3.24}$$

Combining these equalities, we find:

$$\begin{aligned} H(R) &= H(RP^{(1)}) - H(P^{(1)}) = H(P^{(2)}P^{(3)}) - H(P^{(1)}) \\ &\leq H(P^{(2)}) + H(P^{(3)}) - H(P^{(1)}) \ , \end{aligned} \tag{3.25}$$

where the last inequality follows from the subadditivity of the von Neumann entropy. Repeating the same argument while swapping the roles of $P^{(1)}$ and $P^{(2)}$, we find

$$H(R) \leq H(P^{(1)}) + H(P^{(3)}) - H(P^{(2)}) \ . \tag{3.26}$$

Since both inequalities hold simultaneously, we may write

$$H(R) \leq -|H(P^{(1)}) - H(P^{(2)})| + H(P^{(3)}) \leq H(P^{(3)}) \ . \tag{3.27}$$

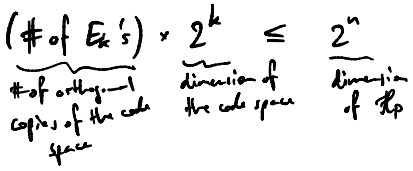
Recalling the value of $H(R)$ we computed earlier, we find

$$\begin{aligned} k \log_2(q) = H(R) &\leq H(P^{(3)}) \\ &\leq \log_2 \dim(\mathscr{H}_{P^{(3)}}) = \big(n - 2(d-1)\big) \log_2(q) \ . \end{aligned} \tag{3.28}$$

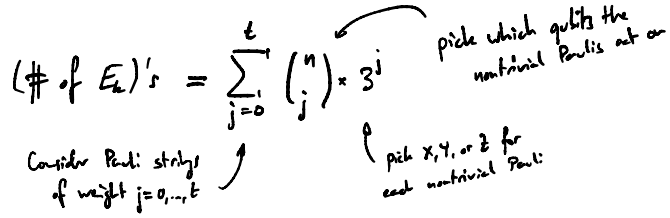
We therefore obtain the ***quantum singleton bound***, applicable to any n-particle code:

$$n - k \geq 2(d-1) \ . \tag{3.29}$$

The **quantum Hamming bound** applies to *nondegenerate n-qubit codes* ($q = 2$) that can correct t-qubit errors. Consider a basis of error operators $\{E_k\}$ spanning the set of t-qubit errors Err_t that is orthogonal with respect to the Hilbert-Schmidt inner product, $\mathrm{tr}(E_{k'}^\dagger E_k) = 0$ whenever $k \neq k'$. From the Knill-Laflamme conditions, and since the code is nondegenerate, the error spaces $\{E_k\mathcal{C}\}$ are all nontrivial and orthogonal to each other. There must be enough room in the physical Hilbert space $\mathscr{H}_P$ to accommodate all these error spaces:



How many E_k's do we have? We can choose as a basis of errors all Pauli strings of weight $\leq t$ to find:



Putting everything together, we obtain the ***quantum Hamming bound*** which is valid for nondegenerate n-qubit codes that can correct t-qubit errors:

$$\sum_{j=0}^{t} \binom{n}{j} 3^j \leq 2^{n-k} \ . \tag{3.30}$$

3.5 Further reading

I encourage readers to look up additional details in Gottesman's book [1]. Readers will find in particular some technical arguments that I omitted, such as an upper bound on the distance between an i.i.d. noise channel and a noise channel whose Kraus operators have weight at most t (see Theorem 1.1 in [1]).

A significant majority of n-particle quantum error-correcting codes that are commonly used and studied are stabilizer codes, and we'll cover the stabilizer formalism in the next chapter. Stay tuned!

The landscape of n-particle codes (especially n-qubit codes) that are not stabilizer codes is a bit scattered. Examples of n-particle codes that are not stabilizer codes include permutation-invariant codes EC Zoo, spin codes EC Zoo, and more EC Zoo.

Chapter 4

Qubit Stabilizer Codes

Now we'll introduce powerful techniques to actually design useful, performant codes. Because these techniques draw deep inspiration from classical error correction, we'll start by talking about classical linear codes.

4.1 Classical binary linear codes

A classical bintstring $x \in \{0,1\}^{\times n}$ can be viewed as an element of the binary vector space $\mathbb{F}_2^n$, over the field of binary numbers $\mathbb{F}_2 = \{0,1\}$, with the addition modulo 2.

A ***binary linear code*** is defined as a linear subspace C of $\mathbb{F}_2^n$. That is, the bitwise XOR of two code words is another code word.

Specifically, binary linear code encodes k bits into n bits via a mapping that can be specified by linearly independent binary vectors $v_1, \ldots, v_k \in \mathbb{F}_2^n$. The k logical bits $\alpha = (\alpha_1, \ldots, \alpha_k)$ are encoded into the bitstring

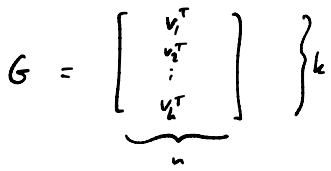
$$v(\alpha) = \sum_{i=1}^{k} \alpha_i v_i \quad \text{(addition modulo 2)} \; . \tag{4.1}$$

The $\{v_i\}$ form a basis of C.

The ***generator matrix*** G of the code C maps α to $v(\alpha)$ via

$$v(\alpha) = G^T \alpha \; . \tag{4.2}$$

It is a $k \times n$ matrix whose rows are the $\{v_i\}$'s:



Alternatively, we can specify the k-dimensional code subspace C of $\mathbb{F}_2^n$ by giving $n-k$ linear constraints. The ***parity check matrix*** H collects these constraints into $(n-k)\times n$ matrix satisfying

$$Hv = 0 \quad \Leftrightarrow \quad v \in C \quad \Leftrightarrow \quad \exists\, \alpha :\ v = G^T\alpha\ . \tag{4.3}$$

The rows of H are linearly independent and span the space of all bitstrings that are orthogonal to all the bitstrings in C. This implies:

$$HG^T = 0\ . \tag{4.4}$$

Orthogonality is meant with respect to the ***inner product*** $u\cdot v \equiv u^T v = \sum u_i v_i$ with addition modulo 2. We have $u^T v = 0$ if and only if both u and v take the value 1 at an even number of locations. Contrary to our intuition in real or complex vector spaces, a bitstring can be orthogonal to itself!

The parity check matrix gives us precious information about what errors the encoded bitstring might have suffered. Suppose a code word $v = G^T\alpha$ has some of its bits corrupted. In such a case, $v \to v' = v + e$ where e is a bitstring that takes the value 1 whenever v' differs from the noiseless bitstring v. If we apply H, we find:

$$Hv' = H(v+e) = \underbrace{Hv}_{=0} + He = He\ . \tag{4.5}$$

The bitstring He which results from applying the parity check matrix H on an error e is called the ***syndrome*** of the error e.

Recovery of the encoded bitstring is possible if we can infer e from its syndrome He. In this case, it suffices to add e again to restore the noiseless bitstring:

$$v' \to v' + e = v + e + e = v\ . \tag{4.6}$$

(Recall addition is modulo 2.)

Inferring e from He is not possible if any arbitrary bitstring $e \in \mathbb{F}_2^n$ can occur as an error. Instead, we fix a set of errors E that we wish to be able to correct, and we design a code to ensure that all $e_i \in \mathsf{E}$ have distinct syndromes.

If two errors $e_1, e_2 \in \mathsf{E}$ have the same syndrome, $He_1 = He_2$, then recovery is likely to fail. Indeed, if e_1 occurs ($v \to v' = v + e_1$) but we attempt to correct the error by applying e_2 ($v' \to v' + e_2$), we obtain

$$v' + e_2 = v + e_1 + e_2 \neq v\ . \tag{4.7}$$

That is, we failed to restore the original code word. Even worse, we have $H(v+e_1+e_2) = 0$, meaning that we've restored a bitstring in the code space but it's the wrong one. We caused a logical error!

The ***distance*** of a binary linear code C is the minimum Hamming distance between any two code words. It is equal to the minimal Hamming weight of any nonzero code word.

Remember that the ***Hamming weight*** $|x|$ of a bitstring x is the number of 1's that appear in x. The ***Hamming distance*** between bitstrings x, y is the number of locations where the bitstrings differ.

Exercise: Show the equivalence of the two definitions of the distance of a binary linear code given above.

A n-bit binary linear code encoding k bits with a distance d is referred to as a $[n, k, d]$ ***code***. Note the single brackets—as you see, we chose to use double brackets versus single brackets to distinguish quantum codes from classical codes.

Theorem 7. *A binary linear code of distance d can correct up to $t = (d-1)/2$ corrupted bits.*

Proof. For any distinct e_1, e_2 with $|e_1|, |e_2| \leq t$, we have $|e_1 + e_2| \leq 2t < d$; by definition of the distance, $e_1 + e_2$ cannot be a code word. Hence $H(e_1 + e_2) \neq 0$, i.e. $He_1 \neq He_2$, and all errors of weight $\leq t$ give rise to distinct syndromes. ∎

Taking the transpose of the fundamental equation (4.4) relating the generator matrix G and the parity check matrix H, we find

$$GH^T = 0 \ . \tag{4.8}$$

If we swap the roles of G and H, we obtain a new code $C^\perp$ with generator matrix $G' := H$ and parity check matrix $H' := G$. The code $C^\perp$ encodes $n - k$ bits and has k parity check constraints. The code $C^\perp$ is called the ***dual code*** of C.

A useful identity relating a code C to its dual code $C^\perp$ is the following.

Lemma 8. *For any bitstring u,*

$$\sum_{v \in C} (-1)^{u \cdot v} = \begin{cases} 2^k & \text{if } u \in C^\perp \\ 0 & \text{if } u \notin C^\perp \ . \end{cases} \tag{4.9}$$

Proof. The first case is straightforward, as the 2^k terms in the sum each take the value $+1$. To prove the second case, we start from the identity

$$\sum_{\alpha \in \{0,1\}^{\times k}} (-1)^{\alpha \cdot w} = 0 \quad \text{for any } w \neq 0 \ , \tag{4.10}$$

where $\alpha, w \in \{0,1\}^{\times k}$.

Exercise: Prove the identity (4.10).

We parametrize $v \in C$ as $v = G^T \alpha$ with $\alpha \in \{0,1\}^{\times k}$. Then

$$\sum_{v \in C} (-1)^{u \cdot v} = \sum_{\alpha \in \{0,1\}^{\times k}} (-1)^{u^T G^T \alpha} = \sum_{\alpha \in \{0,1\}^{\times k}} (-1)^{\alpha \cdot (Gu)} = 0 \quad \text{if } Gu \neq 0 \ . \tag{4.11}$$

Since G is the parity check matrix of $C^\perp$, we are guaranteed to have $Gu \neq 0$ whenever $u \neq C^\perp$. ∎

4.2 The qubit stabilizer formalism

Stabilizer codes can be thought of as a quantum analog of binary linear codes defined by their parity check matrix. The idea is to specify the code space through a set of measurements for which all valid code words have a definite outcome (analogously to the constraints represented by the parity check matrix), and where incorrect measurement outcomes provide valuable information about the errors that might have occurred (analogously to how the syndrome informs which error occurred).

The n-qubit ***Pauli group*** P_n is the group composed of all tensor products of the single-qubit Pauli operators

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \; ; \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \; ; \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \; , \tag{4.12}$$

along with $\mathbb{1}$ and a possible global phase multiple of i. It is generated by single-qubit X and Z operators acting on each individual qubit:

$$\mathsf{P}_n = \langle i\mathbb{1}, X_1, \ldots, X_n, Z_1, \ldots Z_n \rangle \; . \tag{4.13}$$

A generic element of the Pauli group, called *Pauli operator* or *Pauli string*, consists of a global phase in $\{\pm 1, \pm i\}$ and a tensor product of $\mathbb{1}$, X, Y, and Z's over each qubit. For instance:

$$P = (-i)\, X \otimes \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes \mathbb{1} \otimes X \otimes Y \quad \in \mathsf{P}_7 \; . \tag{4.14}$$

Alternative ways to write Pauli operators are often preferred for the concision, and we'll use them interchangably:

$$P = (-i)\, XIZIIXY \; ; \qquad P = (-i)\, X_1 Z_3 X_6 Y_7 \; . \tag{4.15}$$

In the first variant, tensor product symbols are omitted and single-qubit identity operators are written as 'I'. In the second variant, identity operators are omitted entirely and an index indicates on which qubit each single-qubit operator acts. We'll only use the first notation when it is clear that no multiplication is involved; for instance, the notation is useful when displaying a list of Pauli operators.

Recall some elementary properties of X, Y, and Z:

$$\begin{array}{cc} X^2 = Y^2 = Z^2 = \mathbb{1} & \text{(Paulis square to } \mathbb{1}\text{)} \\ XY = iZ \; ; \quad YZ = iX \; ; \quad ZX = iY \; ; & \text{(multiplication rules)} \\ XY = -YX \; ; \quad YZ = -ZY \; ; \quad ZX = -XZ \; . & \text{(Paulis anticommute)} \end{array} \tag{4.16}$$

Some important properties of a generic element of P_n are:

- Any $P \in \mathsf{P}_n$ is unitary: $P^{-1} = P^\dagger$;
- For any $P \in \mathsf{P}_n$, either $P = P^\dagger$ (if its global phase is ± 1), or $P = -P^\dagger$ (if its global phase is $\pm i$);
- Every $P \not\propto \mathbb{1}$ has two distinct eigenvalues. These are ± 1 if $P = P^\dagger$ or $\pm i$ if $P = -P^\dagger$. In either case, both eigenspaces have the same dimension 2^{n-1}, half the full Hilbert space;

- For any $P \in \mathsf{P}_n$, either $P^2 = \mathbb{1}$ (if $P = P^\dagger$) or $P^2 = -\mathbb{1}$ (if $P = -P^\dagger$);
- For any $P \in \mathsf{P}_n$:

$$\operatorname{tr}(P) = \begin{cases} i^k 2^n & \text{if } P = i^k \mathbb{1} \ ; \\ 0 & \text{otherwise.} \end{cases} \tag{4.17}$$

- For any $P, Q \in \mathsf{P}_n$, either $PQ = QP$ or $PQ = -QP$. Equivalently, either $[P, Q] = 0$ or $\{P, Q\} = 0$, where $[A, B] = AB - BA$ and $\{A, B\} = AB + BA$.

It turns out that the Pauli operators are useful to define quantum versions of "checks" as measurements for which all code states have a single, fixed outcome. If we pick r Pauli operators that commute, we can define the code space as a fixed common eigenspace of those operators. The Pauli group structure, along with the properties above, will turn out to provide a range of useful properties of the associated code space, such as its dimension, a characterization of which errors the code can correct, and simple physical implementations of basic logical operators.

Before we define a qubit stabilizer code, it is useful to understand more precisely under which circumstances a common $+1$ eigenspace of a set of Pauli operators can identify a nontrivial subspace.

For any arbitrary subspace $\mathcal{C}$ of the n-qubit Hilbert space $\mathbb{C}^{2^n}$, the ***Pauli stabilizer group*** (or simply ***stabilizer group*** or even ***stabilizer***) of $\mathcal{C}$ is

$$\mathcal{S}(\mathcal{C}) = \left\{ P \in \mathsf{P}_n \ : \ P|\psi\rangle = |\psi\rangle \ \ \forall \ |\psi\rangle \in \mathcal{C} \right\} . \tag{4.18}$$

(At this point, $\mathcal{S}(\mathcal{C})$ is a set of operators. We'll see shortly that it is, in fact, a group.)

Some important properties of the Pauli stabilizer group are:

- The set $\mathcal{S}(\mathcal{C})$ never contains $-\mathbb{1}$, i.e., $-\mathbb{1} \notin \mathcal{S}(\mathcal{C})$.

  Proof. Follows from the fact that $-\mathbb{1}|\psi\rangle = -|\psi\rangle \neq |\psi\rangle$ for any $|\psi\rangle \in \mathcal{C}$. ∎

- For any $P \in \mathcal{S}(\mathcal{C})$, we have $P = P^\dagger$, i.e., P cannot have $\pm i$ global phases.

  Proof. If P had global phase $\pm i$, then its eigenvalues would be $\pm i$ and P would not have a $+1$ eigenvalue. ∎

- $\mathcal{S}(\mathcal{C})$ is a group.

  Proof. If $P, Q \in \mathcal{S}(\mathcal{C})$, then $QP|\psi\rangle = |\psi\rangle$ for all $|\psi\rangle \in \mathcal{C}$, so $PQ \in \mathcal{S}(\mathcal{C})$. Also $P^{-1} = P^\dagger = P \in \mathcal{S}(\mathcal{C})$. ∎

- $\mathcal{S}(\mathcal{C})$ is Abelian.

  Proof. Let $P, Q \in \mathcal{S}(\mathcal{C})$. Either $PQ = QP$ or $PQ = -QP$. If the latter were true, then $|\psi\rangle = PQ|\psi\rangle = -QP|\psi\rangle = -|\psi\rangle$ which is a contradiction, so P and Q must commute. ∎

Conversely, any Abelian subgroup $\mathcal{S}$ of P_n with $-\mathbb{1} \notin \mathcal{S}$ defines a nontrivial subspace $\mathcal{C} \equiv \mathcal{C}(\mathcal{S})$ for which $\mathcal{S}(\mathcal{C}) = \mathcal{S}$:

$$\mathcal{C}(\mathcal{S}) = \big\{|\psi\rangle \,:\, P|\psi\rangle = |\psi\rangle \;\; \forall\, P \in \mathcal{S}\big\} \;. \tag{4.19}$$

Here, $\mathcal{C}(\mathcal{S})$ is the common $+1$ eigenspace of all elements in $\mathcal{S}$, noting that all elements of $\mathcal{S}$ commute.

An arbitrary subspace of the n-qubit Hilbert space generically cannot be fully identified by such a stabilizer group $\mathcal{S}$. (Such is the case, for instance, if the subspace dimension is not a power of two.) So generically, $\mathcal{C}(\mathcal{S}(\mathcal{C})) \neq \mathcal{C}$. However, for any $\mathcal{S}$ we do have $\mathcal{S}(\mathcal{C}(\mathcal{S})) = \mathcal{S}$.

Any Abelian subgroup $\mathcal{S}$ of P_n with $-\mathbb{1} \notin \mathcal{S}$ is the stabilizer group of its associated subspace $\mathcal{C}(\mathcal{S})$, so necessarily satisfies the properties listed above. In particular, we have $S = S^\dagger$ for all $S \in \mathcal{S}$. Also, by a slight abuse of terminology, we often refer to elements of $\mathcal{S}$ as *stabilizers*.

A ***stabilizer code*** is specified by an Abelian subgroup $\mathcal{S} \subseteq \mathsf{P}_n$ with $-\mathbb{1} \notin \mathcal{S}$. That is, its code space $\mathcal{C}$ is the subspace of all states $|\psi\rangle$ that are stabilized by all elements of $\mathcal{S}$:

$$\mathcal{C} = \big\{S|\psi\rangle = |\psi\rangle \;\; \forall\, S \in \mathcal{S}\big\} \;. \tag{4.20}$$

The group $\mathcal{S}$ can be specified by a choice of *independent generators* $\mathcal{S} = \langle S_1, \ldots S_r\rangle$. These are elements $S_i \in \mathcal{S}$ such that any $S \in \mathcal{S}$ can be written as a product of the chosen S_i's (the S_i's *generate* $\mathcal{S}$), and furthermore, removing any individual S_i causes the other S_i's not to generate all of $\mathcal{S}$ (they are *independent*).

All elements of $\mathcal{S}$ commute, and $S_i^2 = \mathbb{1}$, so any $S \in \mathcal{S}$ can be written uniquely as

$$S = S_1^{i_1} S_2^{i_2} \cdots S_r^{i_r} \;, \quad i_j \in \{0, 1\} \;. \tag{4.21}$$

Therefore, the size of $\mathcal{S}$ is the number of bitstrings of length r:

$$|\mathcal{S}| = 2^r \;. \tag{4.22}$$

Recall that the projector onto the ± 1 eigenspace of a Pauli operator P can be written as $(\mathbb{1} \pm P)/2$ if $P = P^\dagger$. We can therefore write the projector onto $\mathcal{C}(\mathcal{S})$ as

$$\Pi_{\mathcal{S}} = \prod_{j=1}^{r} \frac{\mathbb{1} + S_j}{2} \;. \tag{4.23}$$

Expanding the product, we find

$$\Pi_{\mathcal{S}} = \Big(\frac{\mathbb{1} + S_1}{2}\Big) \cdots \Big(\frac{\mathbb{1} + S_r}{2}\Big) = \frac{1}{2^r} \sum_{i_1, \ldots, i_r \in \{0,1\}} S_1^{i_1} \cdots S_r^{i_r} = \frac{1}{2^r} \sum_{S \in \mathcal{S}} S \;. \tag{4.24}$$

This formula is useful as it directly relates the stabilizer elements to the code space projector.

We can compute the size of the code space $\mathcal{C}$ thanks to (4.24):

$$\dim(\mathcal{C}) = \mathrm{tr}(\Pi_{\mathcal{S}}) = \frac{1}{2^r} \sum_{S \in \mathcal{S}} \mathrm{tr}(S) = 2^{n-r} \;, \tag{4.25}$$

noting that all terms in the sum vanish except for the term $S = \mathbb{1}$.

Therefore, any n-qubit Pauli stabilizer group with r independent generators encodes $k = n - r$ logical qubits.

Intuitively, each independent generator S_i halves the remaining Hilbert space: S_1 selects half the original Hilbert space, S_2 divides the $+1$-eigenspace of S_1 into those that belong to the $+1$-eigenspace of S_2 and those in the -1-eigenspace, and so on.

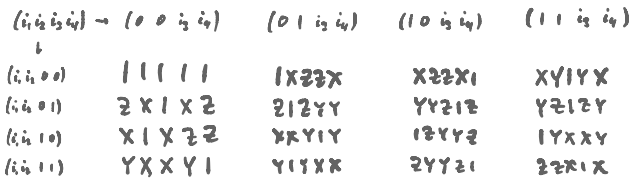
Lemma 9. *Let $\mathcal{S}$ be a stabilizer group with independent generators $S_1, \ldots S_r$. For any j, there exists $P_j \in \mathsf{P}_n$ such that $\{P_j, S_j\} = 0$ and $[P_j, S_i] = 0$ for all $i \neq j$.* *(Proof on page 60.)*

(We'll defer the proof of this lemma until we introduce a useful formalism for stabilizers based on binary linear algebra.)

Example: With $n = 5$, consider the stabilizer group $\mathcal{S} = \langle S_1, \ldots, S_4 \rangle$ with

$$\begin{aligned} S_1 &= X \otimes Z \otimes Z \otimes X \otimes \mathbb{1} \equiv XZZXI \equiv X_1 Z_2 Z_3 X_4 \ ; \\ S_2 &= IXZZX \ ; \\ S_3 &= XIXZZ \ ; \\ S_4 &= ZXIXZ \ . \end{aligned} \tag{4.26}$$

As you can see, S_2, S_3, and S_4 are cyclically permuted versions of the stabilizer S_1. The last cyclic permutation, $ZZXIX$, is not independent of the other generators, since $ZZXIX = S_1 S_2 S_3 S_4$. There are $r = 4$ independent generators, so *this code encodes $k = 1$ qubit.* We can list all $2^4 = 16$ elements of $\mathcal{S}$:



No two possible products of the S_i's are the same, which indeed confirms that the S_i are independent.

For a general qubit stabilizer code, code words can be found by projecting your favorite n-qubit states onto $\mathcal{C}$ by applying $\Pi_{\mathcal{S}}$.

For the 5-qubit code example given above, we find a code word by starting for instance with the state $|00000\rangle$:

$$\begin{aligned} \Pi_{\mathcal{S}} |00000\rangle \propto \big(& |00000\rangle + |01010\rangle + |10100\rangle - |11110\rangle \\ & + |01001\rangle - |00011\rangle - |11101\rangle - |10111\rangle \\ & + |10010\rangle - |11000\rangle - |00110\rangle - |01100\rangle \\ & - |11011\rangle - |10001\rangle - |01111\rangle + |00101\rangle \big) \end{aligned}$$

In general, computing the code words explicitly for a stabilizer code is possible, but is tedious. Fortunately, finding explicit code words is often unnecessary since we can determine which *operators* encode the logical information using the stabilizer formalism.

For the 5-qubit code example given above, we can see the operator $XXXXX$ commutes with all elements of $\mathcal{S}$ (it commutes with all its generators) but is not in $\mathcal{S}$. Had we included $XXXXX$ in our stabilizer group, the stabilized subspace would have dimension 1 (i.e., 0 qubits) instead of 2 (i.e., 1 qubit). Therefore $XXXXX$ must have a nontrivial action on the code space. I'ts a x*logical operator*. We can take it to represent the "X" operator of our encoded logical qubit.

The stabilizer formalism is also useful to represent certain specific quantum states. We simply pick $r = n$ independent stabilizer generators, which stabilize a one-dimensional subspace spanned by a single quantum state. States that can be described in this way are called ***stabilizer states***.

Exercise: Find the states that are stabilized by the following stabilizer groups: (i) $\mathcal{S} = \langle XX, ZZ\rangle$*; (ii)* $\mathcal{S} = \langle Z_1Z_2,\, Z_2Z_3,\, \ldots,\, Z_{n-1}Z_n,\, X_1X_2\cdots X_n\rangle$.

This representation is useful to simulate certain quantum circuits on a classical computer. If the circuits have the property of mapping stabilizer states to stabilizer states (such circuits are called *Clifford circuits*), then we can simulate the state evolution generated by the circuit by keeping track of the n stabilizers that identify the quantum state, rather than tracking the state's 2^n complex coefficients.

4.3 Correcting errors in stabilizer codes

By design, measuring any $S \in \mathcal{S}$ on a code state $|\psi\rangle \in \mathcal{C}$ always yields $+1$ in the absence of errors. If, when measuring $S \in \mathcal{S}$, we find -1, an error must have occurred.

Remember that all elements of $\mathcal{S}$ commute, so they can be measured simultaneously. It actually suffices to measure the generators $\{S_i\}$ of the stabilizer group $\mathcal{S} = \langle S_1, \ldots S_r\rangle$, rather than each individual element of the group. This is because the measurement outcome of a product SS' for $S, S' \in \mathcal{S}$ is predetermined as the product of the measurement outcomes of the individual operators S, S'.

A stabilizer code operates by measuring a choice of stabilizers over multiple rounds. Usually, we choose to measure the independent stabilizer generators. The measurement outcomes are called the ***syndromes*** (or ***syndrome vector***). The syndromes represent all the information we have about an error that might have occurred.

If we flip a generator $S_i \to -S_i$, we obtain an *equivalent* stabilizer group that stabilizes a copy of $\mathcal{C}$ that is orthogonal to the original $\mathcal{C}$. Thus, the syndrome bitstrings associated with $S_1, \ldots S_r$ label mutually orthogonal copies of $\mathcal{C}$. Taking 2^r copies of a 2^k-dimensional space takes up $2^{k+r} = 2^n$ dimensions, filling up the n-qubit state space. That is, the copies of $\mathcal{C}$ labeled by the syndrome vector associated with independent stabilizer generators form a partition of the entire n-qubit Hilbert space into error spaces.

Suppose a Pauli error $E \in \mathsf{P}_n$ occurs:

$$|\psi\rangle \to E|\psi\rangle \ . \tag{4.27}$$

How is the measurement outcome of a stabilizer $S \in \mathcal{S}$ affected? We have

$$SE|\psi\rangle = \left\{ \begin{array}{ll} ES|\psi\rangle & \text{if } ES = SE \\ -ES|\psi\rangle & \text{if } ES = -SE \end{array} \right\} = \left\{ \begin{array}{ll} E|\psi\rangle & \text{if } ES = SE \\ -E|\psi\rangle & \text{if } ES = -SE \ . \end{array} \right. \tag{4.28}$$

Therefore, $E|\psi\rangle$ is always an eigenvector of S and the measurement outcome of S is deterministic. **The measurement outcome of S is $+1$ whenever E commutes with S and it is -1 whenever E anticommutes with S. A stabilizer measurement S can thus detect any error E that anticommutes with S.**

Consider the set Err_t of t-qubit errors. They are spanned by Pauli operators $P \in \mathsf{P}_n$ that satisfy $\mathrm{wgt}(P) \leq t$. Recall a n-qubit code can correct the set of t-qubit errors if and only if its distance d satisfies $d \geq 2t+1$. Therefore, if we can determine the distance d of a qubit stabilizer code, we'll know for which t the code can correct the set of t-qubit errors.

Lemma 10. *Let $\mathcal{S}$ be a stabilizer group and let $E \in \mathsf{P}_n$. Then $\Pi_{\mathcal{S}} E \Pi_{\mathcal{S}} \propto \Pi_{\mathcal{S}}$ if and only if either $E \in \mathcal{S}$ up to a phase, or there exists $S \in \mathcal{S}$ such that $ES = -SE$.*

In other words, a Pauli E does not corrupt logical information if it is a stabilizer or if it anticommutes with a stabilizer.

Proof. We can assume without loss of generality that $E = E^\dagger$, or else we can replace $E \to iE$. If $\pm E \in \mathcal{S}$, then $\Pi E \Pi = \pm \Pi \propto \Pi$, where we write $\Pi \equiv \Pi_{\mathcal{S}}$. If there exists $S \in \mathcal{S}$ with $SE = -ES$, we find that

$$\Pi E \Pi = \Pi E \Pi S = \Pi E S \Pi = -\Pi S E \Pi = -\Pi E \Pi \ , \tag{4.29}$$

which implies that $\Pi E \Pi = 0 \propto \Pi$. On the other hand, if $E \notin \mathcal{S}$, $-E \notin \mathcal{S}$, and $ES = SE$ for all $S \in \mathcal{S}$, then E is independent of $S_1, \ldots, S_r$ while commuting with those operators. The eigenspaces of E further halve $\mathcal{C}$ and E acts nontrivially on the code space. ∎

Therefore, **the distance of a stabilizer code is the smallest weight of a Pauli operator P that commutes with all elements of $\mathcal{S}$ and is not itself a stabilizer up to a phase.**

It is convenient to introduce some additional mathematical concepts to give a more concise version of the above statement.

The ***normalizer*** $N(\mathcal{S})$ of $\mathcal{S}$ in P_n is

$$N(\mathcal{S}) = \{P \in \mathsf{P}_n \ : \ P\mathcal{S} = \mathcal{S}P\} \ . \tag{4.30}$$

It turns out that for Pauli stabilizer groups, the normalizer coincides with the ***centralizer*** $Z(\mathcal{S})$ of $\mathcal{S}$ in P_n, defined as the set of all Pauli operators that commute with all elements of $\mathcal{S}$:

$$Z(\mathcal{S}) = \{P \in \mathsf{P}_n \ : \ \forall\, S \in \mathcal{S}\,,\ PS = SP\} = N(\mathcal{S}) \ . \tag{4.31}$$

On the other hand, we can use the notation $\langle i, \mathcal{S}\rangle \equiv \langle i\mathbb{1}, S_1, \ldots, S_r\rangle \equiv \bigcup_{\ell=0}^{3} i^\ell \mathcal{S}$ to designate the group formed by all elements of the stabilizer group, multiplied by a possible phase in $\{\pm 1, \pm i\}$.

We always have $\langle i, \mathcal{S}\rangle \subseteq N(\mathcal{S})$.

Therefore, the set $N(\mathcal{S}) \setminus \langle i, \mathcal{S}\rangle$ consists precisely of all Paulis that commute with all elements of $\mathcal{S}$, without being themselves a stabilizer up to a phase. The distance of a Pauli stabilizer code with Pauli stabilizer group $\mathcal{S}$ can thus be expressed as

$$d = \min\{\mathrm{wgt}(P) \,:\, P \in N(\mathcal{S}) \setminus \langle i, \mathcal{S}\rangle\} \,. \tag{4.32}$$

The Knill-Laflamme conditions guarantee that, provided $d \geq 2t+1$, any $E \in \mathsf{P}_n$ that is a t-qubit error is correctable. How can we find the correction operation and what does it look like?

Not every t-qubit error $E \in \mathsf{P}_n$ leads to a distinct syndrome vector. For example, if E happens to be a stabilizer, the syndrome vector remains all-$+1$, as if no error had happened. But a stabilizer acts trivially on the code space, meaning that no correction operation is required if E is a stabilizer, even up to a phase. But the syndromes associated with independent stabilizer generators of a code with $d \geq 2t+1$ can uniquely identify a t-qubit error *up to a stabilizer*:

Lemma 11. *If $E, F \in \mathsf{P}_n$ lead to the same syndrome vector with respect to a set of independent stabilizer generators, then $F^\dagger E \in N(\mathcal{S})$. If, additionally, $\mathrm{wgt}(E), \mathrm{wgt}(F) \leq t$ with $t \leq (d-1)/2$, then $F^\dagger E \in \langle i, \mathcal{S}\rangle$.*

Proof. If E, F lead to the same syndrome vectors, then they both commute and anticommute exactly with the same generators: For each $i = 1, \ldots r$, we have $ES_i = S_iE \Leftrightarrow FS_i = S_iF$. Then, for any S_i, we have $F^\dagger E S_i = S_i F^\dagger E$ as either $E, F^\dagger$ both commute with S_i or both anticommute with it. This means that $F^\dagger E \in N(\mathcal{S})$. If, additionally, $\mathrm{wgt}(E), \mathrm{wgt}(F) \leq t$, then $\mathrm{wgt}(F^\dagger E) \leq 2t < d$ so by (4.32), $F^\dagger E \notin N(\mathcal{S}) \setminus \langle i, \mathcal{S}\rangle$. Since $F^\dagger E \in N(\mathcal{S})$, we must have $F^\dagger E \in \langle i, \mathcal{S}\rangle$. ∎

We find the following procedure to correct the set of t-qubit errors. If we observe a given syndrome vector, we first find a Pauli operator F with $\mathrm{wgt}(F) \leq t$ that leads to the observed syndromes. We then apply $F^\dagger$ as a correction operation. Whatever t-qubit error E happened, we know that $F^\dagger E$ is a stabilizer up to a phase, so $F^\dagger E|\psi\rangle \propto |\psi\rangle$. The code state is restored, and the error E is corrected up to an unobservable global phase.

A procedure that maps the measured syndrome vector to a decision about which correction to apply is called a ***decoder***.

The decoder we proposed above has two issues:

(i) It might be hard to find the Pauli correction operation F;

(ii) The decoder fails completely if we observe a syndrome vector that is not compatible with any t-qubit error, but which would correspond to a higher-weight error. (Those will happen in practice.)

Issue (i) is a general issue when designing good decoders. Decoding tends to be a hard problem!

We can deal with issue (ii) by coming up with an algorithm that always outputs some reasonable correction operation for any possible observed syndrome vector.

Suppose an error $E \in \mathsf{P}_n$ happens with probability p_E. In fact, phases are irrelevant in error operators because multiplying a noise channel's Kraus operators by arbitrary phases

leaves the channel invariant. Therefore, we'll take p_E to represent the probability of an error E happening up to an irrelevant global phase, so $p_E = p_{i^\ell E}$ and $\sum_{E\in\mathsf{P}_n/\text{phase}} p_E = 1$.

A decoder might try to find the Pauli E with the highest probability p_E of occurring and that is compatible with the observed syndrome vector. In standard error models, the error E with the highest probability p_E typically coincides with the error E with the smallest weight $\text{wgt}(E)$. Guessing that the error that occurred is E, we apply the correction operation $F^\dagger = E^\dagger$. This decoding strategy is called *minimum weight decoding*:

The strategy of ***minimum weight decoding*** consists in finding the error operator $E \in \mathsf{P}_n$ of minimal weight $\text{wgt}(E)$ and such that E gives rise to the observed syndrome vector. Then apply the correction operation $F^\dagger = E^\dagger$.

Because stabilizers act trivially on the code space, it is not necessary to identify specifically which error E occurred. Instead, it suffices to identify the error E up to a stabilizer. A good decoding strategy is to determine which entire class of errors up to stabilizer had the highest likelihood of occurring.

The strategy of ***maximum likelihood decoding*** consists in finding a Pauli operator $E \in \mathsf{P}_n/\text{phase}$ that maximizes the probability of the entire error class $E\mathcal{S} = \{ES \,:\, S \in \mathcal{S}\}$:

$$E = \text{argmax}_{E'\in\mathsf{P}_n/\text{phase}} \sum_{S\in\mathcal{S}} p_{E'S} \ , \tag{4.33}$$

And applying the correction operation $F^\dagger = E^\dagger$.

Maximum likelihood decoding is an optimal decoding strategy in that it maximizes the probability of successfully restoring the noiseless code word if we also include errors whose weight exceeds the code distance. Denote by $\boldsymbol{s} = \text{syndrome}(E)$ the (deterministic) syndromes that an error $E \in \mathsf{P}_n$ gives rise to and let $F(\boldsymbol{s})$ be the output of the decoder given the input syndromes $\boldsymbol{s}$. Denoting by $\chi\{\ldots\}$ the characteristic function equal to one (zero) whenever its argument is true (false), the overall probability that the recovery is successful is

$$\begin{aligned} p_{\text{success}} &= \sum_{E\in\mathsf{P}_n/\text{phase}} p_E \ \chi\{F(\boldsymbol{s})^\dagger E\,|\psi\rangle \propto |\psi\rangle\} \\ &= \sum_{E\in\mathsf{P}_n/\text{phase}} p_E \ \chi\{F(\boldsymbol{s})^\dagger E \in \mathcal{S} \ \ (\text{up to phase})\} \\ &= \sum_{E\in\mathsf{P}_n/\text{phase}} p_E \ \chi\{E \in F(\boldsymbol{s})\mathcal{S} \ \ (\text{up to phase})\} \\ &= \sum_{\boldsymbol{s}} \sum_{\substack{E\in F(\boldsymbol{s})\mathcal{S} \\ \text{syndrome}(E)=\boldsymbol{s}}} p_E = \sum_{\boldsymbol{s}} \left(\sum_{S\in\mathcal{S}} p_{F(\boldsymbol{s})S} \right) . \end{aligned} \tag{4.34}$$

In the first three lines, we write $\boldsymbol{s} \equiv \text{syndrome}(E)$. In the last equality, we implicitly assume that the decoder always picks $F(\boldsymbol{s})$ to give rise to the same syndrome as the observed $\boldsymbol{s}$; otherwise, remaining uncorrected syndromes would mean that the code word would not be restored in the code space and the success probability for that particular $\boldsymbol{s}$ would be zero. The maximum value of the expression (4.34) is obtained if we maximize each term in the first sum independently; this is exactly what maximum likelihood decoding does.

Designing good decoders for different types of codes and error models is a rich area of active research. We'll dig deeper into this topic later, as we begin discussing fault tolerance.

4.4 Logical operators in stabilizer codes

Logical operators can be identified with elements of the quotient group $N(\mathcal{S})/\mathcal{S}$, i.e., all elements of the normalizer up to a stabilizer. (Remember, stabilizers act trivially on the code space.)

In fact, we have $N(\mathcal{S})/\mathcal{S} \simeq \mathsf{P}_k$, which can be taken to be the logical Pauli group, i.e., the Pauli group associated with the encoded logical qubits.

Example: In the 5-qubit code discussed above, the operator $XXXXX$ commutes with $S_1, \dots S_4$ and thus with all the elements of $\mathcal{S}$. But $XXXXX$ is not itself a stabilizer: $XXXXX \in N(\mathcal{S}) \setminus \langle i, \mathcal{S}\rangle$. It is a logical operator which we can take as the logical X operator of the encoded qubit:

$$\overline{X} = XXXXX \ . \tag{4.35}$$

Similarly, the operator $ZZZZZ$ also commutes with all stabilizers without itself being a stabilizer up to a phase: $ZZZZZ \in N(\mathcal{S}) \setminus \langle i, \mathcal{S}\rangle$. Furthermore, $ZZZZZ$ anticommutes with $XXXXX$. We can take it to be the logical Z operator of the encoded qubit,

$$\overline{Z} = ZZZZZ \ . \tag{4.36}$$

These choices are not unique! Multiplying $\overline{X}$ or $\overline{Z}$ by a stabilizer yields another representative of the same logical operator, with identical action on the code space. Let $S \in \mathcal{S}$ and say we picked $\overline{X}' = \overline{X}S = S\overline{X}$. Then for any $|\psi\rangle \in \mathcal{C}$,

$$\overline{X}'|\psi\rangle = \overline{X}S|\psi\rangle = \overline{X}|\psi\rangle \ . \tag{4.37}$$

In the above 5-qubit code, $\overline{X}S_3 = (XXXXX)(XIXZZ) = -IXIYY$ is another logical X representative. This one happens to have a lower weight, and depending on available hardware operations, might be more practical to implement.

In the 5-qubit code, we can check that any Pauli operator of weight ≤ 2 anticommutes with at least one stabilizer generator $S_1, \dots S_4$. For instance, if $E = YIIZI$, we find that E anticommutes with S_3. (Because $\mathcal{S}$ is unchanged if we cyclically shift the qubits, it suffices to check the preceding property for weight-1 and weight-2 operators of the form $*IIII$, $**III$, and $*I*II$ where each '$*$' is to be replaced by X, Y, Z.) On the other hand, we found a weight-3 logical operator $\overline{X}S_3$. Hence, the distance of this code is $d = 3$.

The stabilizer code with independent generators (4.26) is the well-known $[[5, 1, 3]]$ ***code***.

The fact that a logical operator can have many equivalent representatives can be seen as a manifestation of the fact that the logical quantum information is encoded in the entanglement degrees of freedom of the n qubits and that the recovery of logical information is possible even after the loss of certain qubits. Specifically, if $d-1$ qubits are erased, we know that all logical operators must be recoverable from the remaining $n-d+1$ qubits; in consequence, there must be a logical representative of any logical operator acting only on those qubits. This property is referred to as the *cleaning lemma*.

Lemma 12 (Cleaning lemma)**.** *Let M be a set qubits whose erasure is correctable and let $\overline{O}$ be a logical Pauli operator. Then there exists $S \in \mathcal{S}$ such that $\overline{O}S$ acts trivially on M.*

Proof. Write the erasure of M as a quantum channel $\mathcal{N}_{\text{Eras. }M}(\cdot) = \mathrm{tr}_M(\cdot)$, with $P =$ n qubits and $P' = M^c$. Let $\mathcal{E}(\cdot) = V_{L\to P}(\cdot)\,V^\dagger$ be the isometric encoding channel, where L consists of the k logical qubits. Since the erasure of M is correctable, there exists by definition a quantum channel $\mathcal{D}_{P'\to L}$ such that $\mathcal{D}\,\mathcal{N}_{\text{Eras. }M}\,\mathcal{E} = \mathrm{id}_L$.

Let $O_L = V^\dagger \overline{O} V$ be the action of $\overline{O}$ on the code space. We find an operator $\overline{O}'$ supported on the complement M^c of M that reveals the value of $\overline{O}$ by simply choosing the operator $\overline{O}' := \mathcal{D}^\dagger(O_L) \otimes \mathbb{1}_M$. Indeed, for any ϕ_L,

$$\begin{aligned}\mathrm{tr}\big(\mathcal{E}(\phi_L)\,\overline{O}'\big) &= \mathrm{tr}\big(\mathcal{E}(\phi_L)\,(\mathcal{D}^\dagger(O_L)\otimes \mathbb{1}_M)\big) = \mathrm{tr}\big((\mathcal{D}\,\mathcal{N}_{\text{Eras. }M}\,\mathcal{E})(\phi_L)\,O_L\big)\\ &= \mathrm{tr}(\phi_L\,O_L)\ .\end{aligned} \tag{4.38}$$

It remains to see that this operator can be chosen to be a Pauli operator. For stabilizer codes, the encoding isometry V maps Pauli operators to Pauli operators by construction. Furthermore, we've seen that we can choose the decoding channel $\mathcal{D}_{P'\to L}$ as the channel that (1) appends maximally mixed qubits to P' to reach n qubits again, (2) measures an independent set of stabilizer generators $S_1, \ldots, S_r$, (3) applies a corresponding Pauli correction $F_{\{s_j\}} \in \mathsf{P}_n$ depending on the syndromes $\{s_j\}$, and (4) conjugates by $V^\dagger$ to keep only the code space as the output of the channel. Overall, such a map $\mathcal{D}_{P'\to L}$ is of the form $\mathcal{D}^{(3)}_{P\to L}\,\mathcal{D}^{(2)}_{P\to P}\,\mathcal{D}^{(1)}_{P'\to P}$ with $\mathcal{D}^{(1)}(\cdot) = (\cdot)\otimes(\mathbb{1}_M/2^{|M|})$,

$$\mathcal{D}^{(2)}_{P\to P} = \sum_{\{s_j\}\in\{\pm1\}^{\times r}} F_{\{s_j\}}\big(\textstyle\prod_j \frac{1+s_jS_j}{2}\big)\,(\cdot)\,\big(\prod_j \frac{1+s_jS_j}{2}\big)F^\dagger_{\{s_j\}}\ , \tag{4.39}$$

and $\mathcal{D}^{(3)}(\cdot) = V^\dagger(\cdot)V$. One can check that $\mathcal{D}^{(2)}_{P\to P}$ maps a Pauli operator to an operator that is proportional to a Pauli operator. If O_L is a Pauli operator, then $\overline{O}' = \mathcal{D}^\dagger(O_L)\otimes\mathbb{1}_M$ is proportional to a Pauli operator. The proportionality coefficient must be 1 as we can check from the action of the operator on a code word that is an eigenstate of $\overline{O}$. Since $\overline{O}$ and $\overline{O}'$ are both Pauli logical operators with the same action on the code space, they must be stabilizer-equivalent. ∎

Exercise: Convince yourself that the decoding channel $\mathcal{D}^{(2)}_{P\to P}$ in the proof of the cleaning lemma above is a Pauli channel*: It is diagonal as a superoperator in an operator basis of Pauli operators, i.e., it maps any Pauli operator to a scaled version of that Pauli operator.*

It is also instructive to prove the cleaning lemma directly using the stabilizer formalism; see further reading.

4.5 Binary symplectic representation

The binary symplectic representation of Pauli operators is a powerful tool to describe Paulis up to a phase, capturing their commutation relations, while bringing in the power of binary linear algebra.

To a Pauli $P = \alpha\bigotimes_{i=1}^n P_i \in \mathsf{P}_n$ with $P_i \in \{\mathbb{1}, X, Y, Z\}$ and $\alpha \in \{\pm1, \pm i\}$, we associate a length-$2n$ bitstring $v_P = (x_P|z_P)$ with n-bitstrings x_P, z_P that take the following values:

For $i = 1, \dots n$,

$$(x_i, z_i) = \begin{cases} (0,0) & \text{if } P_i = \mathbb{1} \\ (0,1) & \text{if } P_i = Z \\ (1,0) & \text{if } P_i = X \\ (1,1) & \text{if } P_i = Y. \end{cases} \tag{4.40}$$

For example, the Pauli string $XIYZZ$ is represented as

$$\left(\begin{array}{ccccc|ccccc} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{array} \right) . \tag{4.41}$$

Up to an unspecified global phase, the Pauli operator corresponding to a binary vector $(x|z)$ can be written as

$$P = \Big(\bigotimes_{i=1}^{n} X^{x_i}\Big)\Big(\bigotimes_{i=1}^{n} Z^{z_i}\Big) \equiv X^x Z^z \ , \tag{4.42}$$

where we use the notation $X^x \equiv \bigotimes_{i=1}^{n} X^{x_i}$ for a bitstring $x \in \mathbb{F}_2^n$ to denote the tensor product of $\mathbb{1}$'s and X's in which the X's appear exactly at the locations where $x_i = 1$, and similarly for Z^z.

The ***symplectic product*** (or ***symplectic form***) $\odot$ is defined as

$$(x_1|z_1) \odot (x_2|z_2) = x_1 \cdot z_2 + z_1 \cdot x_2 \ , \tag{4.43}$$

where $\cdot$ is the inner product in $\mathbb{F}_2^n$ defined earlier and where addition is modulo 2.

With $v_1 = (x_1|z_1)$ and $v_2 = (x_2|z_2)$, we can also write

$$v_1 \odot v_2 = (x_1|z_1) \odot (x_2|z_2) = v_1^T \Lambda v_2 \ , \tag{4.44}$$

where v_1, v_2 are considered as column vectors for matrix multiplication, and where Λ is a $2n \times 2n$ binary matrix given by

$$\Lambda = \begin{pmatrix} 0 & \mathbb{1} \\ \mathbb{1} & 0 \end{pmatrix} \ . \tag{4.45}$$

Observe that a vector is always orthogonal to itself according to the binary symplectic inner product:

$$(x|z) \odot (x|z) = x \cdot z + x \cdot z = 0 \ . \tag{4.46}$$

The symplectic product reveals the commutation relation of the corresponding Pauli operators: For all $P, Q \in \mathsf{P}_n$,

$$(x_P|z_P) \odot (x_Q|z_Q) = \begin{cases} 0 & \text{if } PQ = QP \\ 1 & \text{if } PQ = -QP \, . \end{cases} \tag{4.47}$$

We can prove this property as a consequence of how to multiply Paulis in terms of their symplectic representation. For $P = \alpha_P X^{x_P} Z^{z_P}$ and $Q = \alpha_Q X^{x_Q} Z^{z_Q}$ with $\alpha_P, \alpha_Q \in$

$\{\pm 1, \pm i\}$, we find

$$PQ = \alpha_P \alpha_Q \, X^{x_P} Z^{z_P} X^{x_Q} Z^{z_Q} = \alpha_P \alpha_Q \, (-1)^{z_P \cdot x_Q} \, X^{x_P + x_Q} Z^{z_P + z_Q} \ , \tag{4.48}$$

where we pick up a (-1) sign whenever we have to commute a X from the expression for Q through a Z at the same qubit location in the expression for P. We see that

$$v_{PQ} = (x_{PQ} | z_{PQ}) = (x_P + x_Q | z_P + z_Q) = v_P + v_Q \ , \tag{4.49}$$

because the symplectic notation ignores global phases. The phase in the product PQ is still given explicitly in (4.48), enabling us to compare the phase acquired in the product PQ compared to the product QP. We find:

$$\begin{aligned} QP &= \alpha_Q \alpha_P \, (-1)^{z_Q \cdot x_P} \, X^{x_P + x_Q} Z^{z_P + z_Q} = (-1)^{x_P \cdot z_Q + z_P \cdot x_Q} \, PQ \\ &= (-1)^{(x_P | z_P) \odot (x_Q | z_Q)} \, PQ \ , \end{aligned} \tag{4.50}$$

arriving at the claimed property (4.47).

A set of Paulis $P_1, \dots P_\ell$ are independent, i.e., neither one can be written as a product of the other operators, if they are represented in the binary symplectic formalism by *linearly independent vectors* $v_{P_1}, \dots v_{P_\ell}$. (This property is a consequence of (4.49), i.e. of the fact that a product of Pauli operators is represented by a sum of the individual Paulis' representing vectors.)

A stabilizer group $\mathcal{S} = \langle S_1, \dots, S_r \rangle$ is represneted by a linear subspace V spanned by the linearly independent vectors $v_1, \dots v_r$, where $v_i \equiv v_{S_i}$ is the binary symplectic representation of the generator S_i. Because the generators all commute, $S_i S_j = S_j S_i$ for all i, j, we have that $v_i \odot v_j = 0$ for all i, j. In other words, the fact that all stabilizer generators commute is represented by the fact that the vectors are all orthogonal to each other (along with being orthogonal to themselves, $v_i \odot v_i = 0$).

The space $V^\perp$ of all vectors that are orthogonal to all the v_i's corresponds to the normalizer of $\mathcal{S}$. The space $V^\perp$ always includes V as a subspace, which corresponds to the statement that $\mathcal{S} \subseteq N(\mathcal{S})$.

The space V is spanned by $r = n - k$ independent generators; equivalently, it can be specified by $n + k$ constraints. The constraint vectors span $V^\perp$. Therefore, the normalizer's size can be computed as

$$|N(\mathcal{S})| = 4 \times 2^{n+k} \ , \tag{4.51}$$

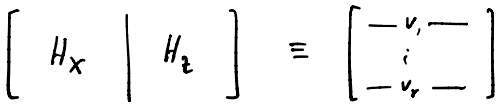
where the factor 4 stems from the choice of global phase in $\{\pm 1, \pm i\}$.

Let us now return to the proof we deferred earlier, of Lemma 9. This proof illustrates the advantage of the binary symplectic formalism of being able to draw on rich tools from binary linear algebra.

Proof of Lemma 9. Given a stabilizer group $\mathcal{S}$ with independent generators $S_1, \dots S_r$, we want to show that for any j there exists $P_j \in \mathsf{P}_n$ such that $P_j S_j = -S_j P_j$ and $P_j S_i = S_i P_j$ for all $i \neq j$. The vectors $v_1, \dots v_r$ are linearly independent and span V. Given any j, we seek $w_j \in \mathbb{F}_2^{2n}$ such that $v_j \odot w_j = 1$ and $v_i \odot w_j = 0$ for all $i \neq j$. These are r independent constraints for $2n$ variables, so there are $2n - r = n + k$ solutions (with $r = n - k$). Any one gives a w_j corresponding to a Pauli P_j with the desired commutation relations. ∎

As a consequence, we can show that there always exists some error $E \in \mathsf{P}_n$ that gives rise to any fixed syndrome vector associated to an independent set of generators. In other words, all syndrome vectors are theoretically possible outcomes in such a case. (There is no guarantee that the error E be a *correctable* error. It might be the case that the error E has high weight.)

We can build a binary matrix representation of a stabilizer code by collecting the vectors $v_1, \dots v_r$ as rows of a binary matrix:

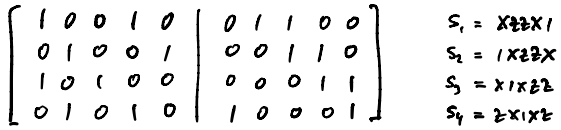


This matrix can be thought of as a "parity check matrix" for a quantum stabilizer code: To any error $E \in \mathsf{P}_n$ with symplectic representation v_E corresponds the syndrome vector

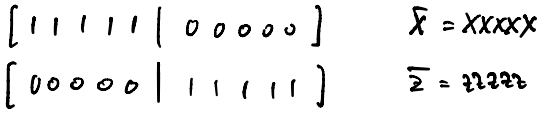
$$s = \left[\begin{array}{c|c} H_X & H_Z \end{array} \right] \Lambda \begin{bmatrix} | \\ v_E \\ | \end{bmatrix}, \tag{4.52}$$

recalling that $\Lambda = \left[\begin{smallmatrix} 0 & \mathbb{1} \\ \mathbb{1} & 0 \end{smallmatrix}\right]$ and $u^T \Lambda v \equiv u \odot v$.

Example: The stabilizer group of the $[[5, 1, 3]]$ code can be represented as:



and with logical operators given by:



Additional remark: Gaussian elimination can be very useful to check linear independence of a set of binary vectors. For instance, you can use it to check that you've found a logical operator by ensuring it commutes with all stabilizer generators and that it is independent of those generators. (See also [1, Procedure 6.6].)

4.6 Calderbank-Shor-Steane (CSS) codes

Qubit Calderbank-Shor-Steane (CSS) codes are an important class of qubit stabilizer codes. They are defined by imposing an additional structure on their stabilizer group: We must be able to generate the group with a list of stabilizer generators that are either of "X type" (containing only X's and $\mathbb{1}$'s) or of "Z type" (containing only Z's and $\mathbb{1}$'s).

A ***CSS code*** is a qubit stabilizer code for which there is a choice of stabilizer generators $S_{Z_1}, \dots, S_{Z_{r_Z}}, S_{X_1}, \dots, S_{X_{r_X}}$ such that each S_{Z_i} is a tensor product of only $\mathbb{1}$'s and Z's, while each S_{X_i} is a tensor product of only $\mathbb{1}$'s and X's. The $S_{Z_1}, \dots S_{Z_{r_Z}}$ are called ***Z-type stabilizer generators*** while the $S_{X_1}, \dots, S_{X_{r_X}}$ are called ***X-type stabilizer generators***.

A CSS code can be represented as a binary matrix of the form:

$$\left[\begin{array}{c|c} & \\ 0 & H_Z \\ & \\ H_X & 0 \\ & \end{array}\right] \tag{4.53}$$

The submatrices H_Z and H_X must necessarily satisfy the following relation to ensure that all stabilizer generators commute:

$$H_X H_Z^T = 0 \ . \tag{4.54}$$

One of the important features of CSS codes is that they are obtained from a specific construction, termed the CSS construction, starting from two binary linear classical codes. This construction was among the first quantum error-correcting codes to be proposed, historically, in the mid-1990's.

The ***CSS construction***: Let C_Z and C_X be two classical n-bit binary linear codes such that $C_X^\perp \subseteq C_Z$. The corresponding CSS code is the qubit stabilizer code with symplectic representation (4.53), in which H_Z and H_X are the parity check matrix associated with the binary linear codes C_Z and C_X.

The condition $C_X^\perp \subseteq C_Z$ ensures that the X-type and the Z-type stabilizers all commute. Indeed, the condition $C_X^\perp \subseteq C_Z$ is equivalent to any $x \in C_X^\perp$ passing the checks for C_Z; such x's span the image of the generator matrix H_X of $C_X^\perp$, meaning that $H_Z H_X^T = 0$. We find:

$$C_X^\perp \subseteq C_Z \quad \Leftrightarrow \quad H_Z H_X^T = 0 \quad \Leftrightarrow \quad H_X H_Z^T = 0 \quad \Leftrightarrow \quad C_Z^\perp \subseteq C_X \ , \tag{4.55}$$

It might have appeared that the condition $C_X^\perp \subseteq C_Z$ introduced an asymmetry between the X-type code and the Z-type code; we see that this is not the case, since the conditions $C_X^\perp \subseteq C_Z$ and $C_Z^\perp \subseteq C_X$ are equivalent.

Suppose H_Z has size $r_Z \times n$ and H_X is $r_X \times n$, i.e., there are r_Z independent Z-type stabilizers and r_X independent X-type stabilizers. Equivalently, the code C_Z encodes $k_Z = n - r_Z$ bits and C_X encodes $k_X = n - r_X$ bits. Then there are $r_X + r_Z$ total independent stabilizers and the quantum CSS code therefore encodes k qubits with

$$k = n - r_X - r_Z = k_X + k_Z - n \ . \tag{4.56}$$

CSS codes can be thought of as a combination of one classical code to correct bit flips, and one classical code to correct phase flips. Any error $E \in \mathsf{P}_n$ can be decomposed in terms of X's and Z's, $E \propto X^{x_E} Z^{z_E}$, up to a phase. The Z's will be picked up by the X-type stabilizers while the X's will be detected by the Z-type stabilizers. This property makes the decoding of CSS codes often easier, because we can focus on each type of error separately. A decoder typically first corrects one type of error (say, X type) using the syndromes that involve the other type of operator (Z-type), and then corrects the other type of error.

Suppose C_Z and C_X have distance d_Z and d_X. Consider an error $E \propto X^{x_E} Z^{z_E}$. If $1 \leq |Z_E| \leq d_X$, then one of the X-type stabilizers will anticommute with E and $E \neq N(\mathcal{S})$.

Similarly, if $1 \leq |x_E| \leq d_Z$, then E is detected by a Z-type stabilizer and $E \neq N(\mathcal{S})$. Therefore, any $E \in N(\mathcal{S}) \setminus \langle i, \mathcal{S} \rangle$ must have weight at least $\mathrm{wgt}(E) \geq \min(d_X, d_Z)$. Therefore the CSS code's distance satisfies

$$d \geq \min(d_X, d_Z) \ . \tag{4.57}$$

This bound is generally not tight, as it fails to account for errors that turn out to be stabilizers. In other words, it is fine if an error E is not detected by any of the classical codes, as long as E is in fact a stabilizer.

Theorem 13. *The distance d of an arbitrary CSS code is determined by the smallest weight of bitstrings in C_X, C_Z which do not correspond to stabilizers:*

$$d = \min(d_X^*, d_Z^*) \ ; \tag{4.58}$$

$$d_X^* = \min\{|x| \ : \ x \in C_X \setminus C_Z^\perp\} \ ; \tag{4.59}$$

$$d_Z^* = \min\{|x| \ : \ x \in C_Z \setminus C_X^\perp\} \ . \tag{4.60}$$

Proof. The condition that $x \in C_Z \setminus C_Z^\perp$ is equivalent to $x \in C_X$ and $Z^x \notin \langle i, \mathcal{S} \rangle$. Indeed, Z^x is a Z-type stabilizer if and only if x is in the linear span of the rows of H_Z, which is $C_Z^\perp$. Similarly, $x \in C_Z \setminus C_X^\perp$ is equivalent to $x \in C_Z$ and $X^x \notin \langle i, \mathcal{S} \rangle$. Consider any $E \propto X^{x_E} Z^{z_E}$ with $E \in N(\mathcal{S}) \setminus \langle i, \mathcal{S} \rangle$. Then $x_E \in C_X$ and $z_E \in C_Z$, otherwise one of the stabilizers would anticommute with E and $E \notin N(\mathcal{S})$. Also, either $x_E \notin C_Z^\perp$ or $z_E \notin C_X^\perp$, as otherwise $E \in \langle i, \mathcal{S} \rangle$. Thus either $x_E \in C_X \setminus C_Z^\perp$ or $z_E \in C_Z \setminus C_X^\perp$. So either $d \geq |x_E| \geq d_X^*$ or $d \geq |z_E| \geq d_Z^*$, which implies $d \geq \min(d_X^*, d_Z^*)$.

On the other hand, let $x \in C_X \setminus C_Z^\perp$ with $|x| = d_X^*$. Let $E = Z^x$. We have $E \notin \langle i, \mathcal{S} \rangle$ since $x \notin C_Z^\perp$. Also, $E \in N(\mathcal{S})$ because $x \in C_X$. Therefore $d \leq \mathrm{wgt}(E) = d_X^*$. Similarly, $d \leq d_Z^*$. Therefore $d \leq \min(d_X^*, d_Z^*)$, completing the proof. ∎

Exercise: Prove the following explicit expression for the code words of a CSS code in the computational basis:

$$|u + C_X^\perp\rangle \propto \sum_{v \in C_X^\perp} |u + v\rangle \qquad (u \in C_Z) \ , \tag{4.61}$$

where $u + C_X^\perp$ enumerates possible cosets of $C_X^\perp$ in C_Z. How do the code words of a CSS code look like in the Hadamard basis?

An important classical binary linear code is the $[7, 4, 3]$ *Hamming code* with parity check matrix

$$H_{\mathrm{Hamming}} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \ . \tag{4.62}$$

Exercise: Check that the $[7, 4, 3]$ Hamming code has distance 3.

We can see that the rows of H_{Hamming} themselves pass the parity check matrix, as well as the bitstring (1 1 1 0 0 0 0). The generator of the code can thus be taken as

$$G_{\text{Hamming}} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{4.63}$$

The $[7, 4, 3]$ Hamming code therefore contains its own dual as a subcode. That is, we can pick C_X and C_Z both to be the Hamming code and we'll have that $C_X^\perp \subseteq C_Z$, as required by the CSS construction. The corresponding CSS code has stabilizer generators

$$\begin{array}{ll} ZIZIZIZ & XIXIXIX \\ IZZIIZZ & IXXIIXX \\ IIIZZZZ & IIIXXXX \ . \end{array} \tag{4.64}$$

It encodes $k = n - r = 7 - 6 = 1$ qubit. The distance of the CSS code is $d = 3$, which follows from $d \geq \min(d_X, d_Z) = 3$ along with $XXXIIII \in N(\mathcal{S}) \setminus \langle i, \mathcal{S} \rangle$. We can identify the logical operators

$$\overline{X} = XXXXXXX \tag{4.65}$$
$$\overline{Z} = ZZZZZZZ \ . \tag{4.66}$$

This code is known as the $[[7, 1, 3]]$ ***Steane code***. The Steane code has some interesting properties related to the implementation of logical gates and which we'll return to when we get to the topic of fault tolerance.

Exercise: For any CSS code, show that we can always choose logical Pauli operators such that logical X's contain only X's and $\mathbb{1}$*'s while logical Z's contain only Z's and* $\mathbb{1}$*'s.*

4.7 Some essential stabilizer codes

Here are some important stabilizer codes.

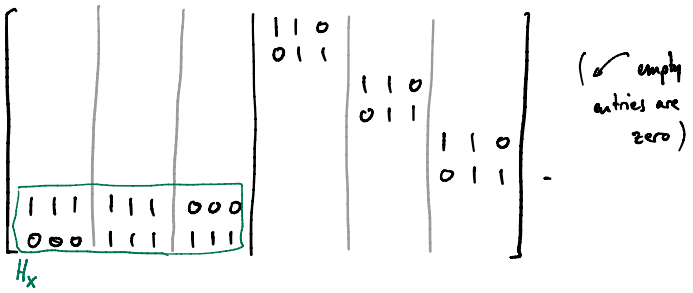
- The $[[5, 1, 3]]$ code (introduced above) ;
- The $[[7, 1, 3]]$ Steane code (introduced above) ;
- The $[[9, 1, 3]]$ Shor code seen in Chapter "Chapter 1" is in fact a CSS code, with stabilizer generators

$$Z_1Z_2, \ Z_2Z_3, \ Z_4Z_5, \ Z_5Z_6, \ Z_7Z_8, \ Z_8Z_9, \tag{4.67}$$
$$X_1X_2X_3X_4X_5X_6, \ X_4X_5X_6X_7X_8X_9 \ , \tag{4.68}$$

  and with logical operators

$$\overline{X} = Z_1Z_4Z_7 \ ; \qquad \overline{Z} = X_1X_2X_3 \ . \tag{4.69}$$

  The stabilizer generators in binary matrix representation is:



The $9, 1, 3$ Shor code is an example of a degenerate code for which the naive distance bound $d \geq \min(d_X, d_Z)$ is not tight. Here, C_X has the parity check matrix H_X and we see that $x = (\,1\;1\;0\;0\;0\;0\;0\;0\;0\,)$ is a code word of C_X. Thus $d_X = 2$. But we've seen that the distance of the $[[9, 1, 3]]$ Shor code is $d = 3$. Indeed, the C_X code word x corresponds to the operator $Z_1 Z_2$, which is in fact a stabilizer.

- The $[[4, 2, 2]]$ ***code*** has stabilizer generators

$$XXXX \;; \quad ZZZZ \;. \tag{4.70}$$

It encodes $4 - 2 = 2$ qubits, for which we can choose the following logical operators:

$$\begin{aligned} X^{(1)} &= X_1 X_3 \;; \quad X^{(2)} = X_1 X_2 \;; \\ Z^{(1)} &= Z_1 Z_2 \;; \quad Z^{(2)} = Z_1 Z_3 \;. \end{aligned} \tag{4.71}$$

The code states are spanned by

$$|\overline{00}\rangle = \frac{1}{\sqrt{2}}\big(|0000\rangle + |1111\rangle\big) \;; \tag{4.72}$$

$$|\overline{01}\rangle = \frac{1}{\sqrt{2}}\big(|0011\rangle + |1100\rangle\big) \;; \tag{4.73}$$

$$|\overline{10}\rangle = \frac{1}{\sqrt{2}}\big(|0101\rangle + |1010\rangle\big) \;; \tag{4.74}$$

$$|\overline{11}\rangle = \frac{1}{\sqrt{2}}\big(|0110\rangle + |1001\rangle\big) \;. \tag{4.75}$$

- The $[[4, 2, 2]]$ code can be generalized to a $[[2m, 2m-2, 2]]$ code for any $m \geq 1$, where the stabilizer generators are taken as

$$ZZZ \ldots Z \;; \qquad XXX \ldots X \;. \tag{4.76}$$

A basis of code words can be chosen of the form

$$|\psi\rangle_x = \frac{1}{\sqrt{2}}\big(|x\rangle + X^{\otimes n}|x\rangle\big) \;, \tag{4.77}$$

for even-weight bitstrings x.

- We'll discuss many more stabilizer codes in the upcoming chapters. We'll begin right away in the next chapter, where we'll cover Kitaev's surface code.

4.8 Further reading

There are many qubit stabilizer codes. Some important families of qubit stabilizer codes include topological codes such as the surface code (the subject of the next couple of chapters), and various constructions of quantum low-density parity check (qLDPC) codes.

I'll also point to an instructive proof of the cleaning lemma that directly uses the stabilizer formalism in the paper which introduced this lemma [30].

There are many additional topics related to qubit stabilizer codes that I wanted to include in these notes in additional chapters but didn't yet have the chance (stay tuned!). These include subsystem codes (see e.g. the Bacon-Shor code), quantum weight enumerators (see here[1]), and holographic codes.

The topic of decoding qubit stabilizer codes is equally rich. Decoding topological codes can exploit a rich structure, see upcoming chapters. For general qubit stabilizer codes, decoding techniques (such as methods based on belief propagation [31]) are listed in the Zoo.[2] Several papers present hardness proofs related to decoding qubit codes [32–34].

The qubit stabilizer formalism can be extended to qudits in multiple ways. The Pauli-X and Z operators can be promoted to qudit operators that obey Weyl commutation relations, giving rise to a family of codes that can be called modular-qudit stabilizer codes. On the other hand, if the qudit states label elements of a Galois field, we can construct Galois-qudit stabilizer codes.

The qubit Pauli stabilizer formalism can also extend to yield so-called XP stabilizer and XS stabilizer codes.

[1] https://errorcorrectionzoo.org/c/qubits_into_qubits#defterm-Quantum_20Xweight_20Xenumerator
[2] https://errorcorrectionzoo.org/c/qubit_stabilizer#features_decoders

Chapter 5

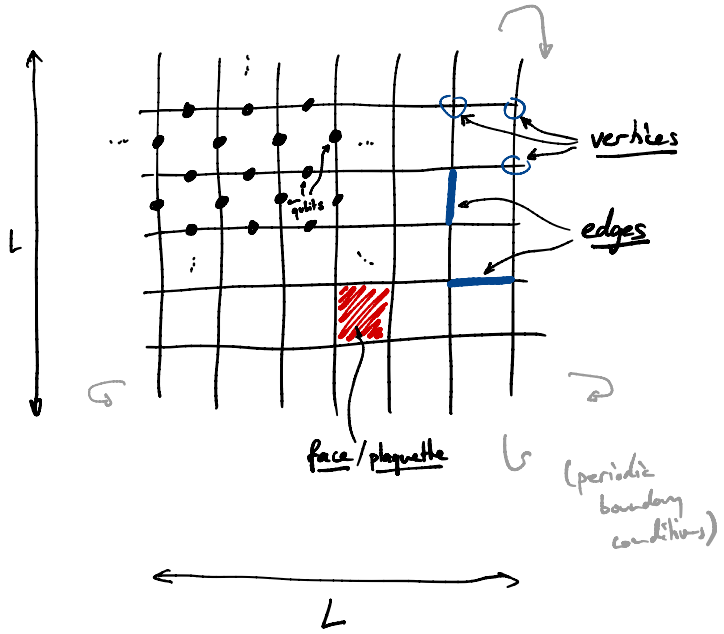
The Surface Code

The surface code is one of the best-studied and most well-known quantum error-correcting codes. It will serve to introduce many important concepts, especially related to computation and fault tolerance, and we'll make some useful connections to condensed matter physics.

A key feature of the surface code is that its stabilizer generators act locally on qubits arranged in a lattice. This geometric property make it the prime example of a *topological code*.

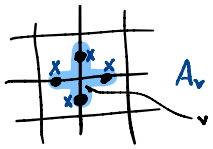
5.1 Construction of Kitaev's toric code

Consider a $L \times L$ square lattice with periodic boundary conditions. Let's place a single qubit on each edge of the lattice:



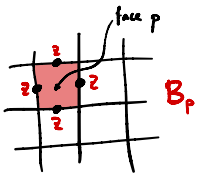
To each vertex v of the lattice, we associate an operator acting on the qubits adjacent to the vertex. These are the ***star operators***:

$$A_v = \prod_{i \text{ adjacent to } v} X_i$$

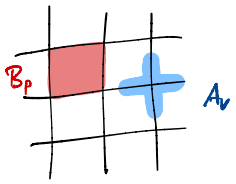


To each face p of the lattice, we associate an operator acting on the qubits along the sides of the face. These are the ***plaquette operators***:

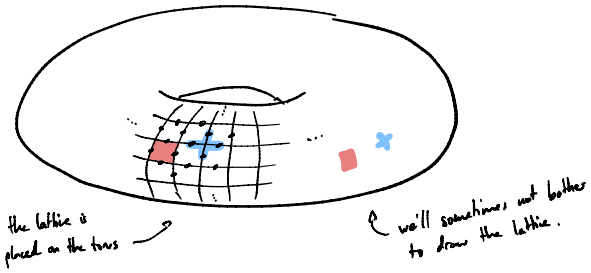
$$B_p = \prod_{i \text{ on bdry } p} Z_i$$



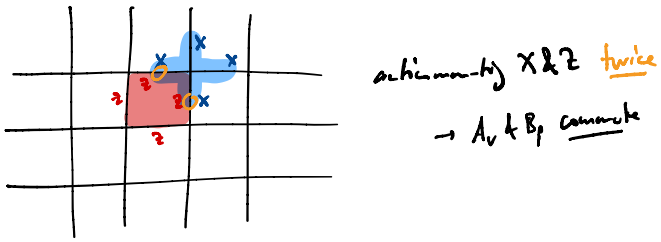
We'll represent star and plaquette operators simply by highlighting the edges and faces where they act:



Remember, there are no boundaries to the lattice because we placed it on the torus. The star and plaquette operators might look like this:



Any two star and plaquette operators *commute*! This property follows from the fact that any two B_p and A_v operators, if they overlap, always do so on an *even* number of qubits:



There are $2L^2$ qubits: L^2 qubits on the horizontally oriented edges and L^2 on the vertically oriented edges. The total number of qubits is

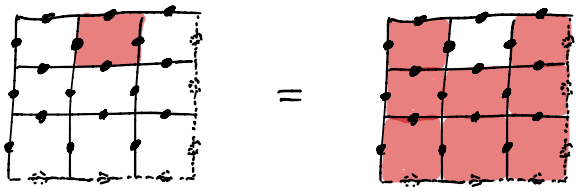
$$n = 2L^2 \ . \tag{5.1}$$

We count L^2 plaquette operators and L^2 star operators, by counting the number of faces and vertices.

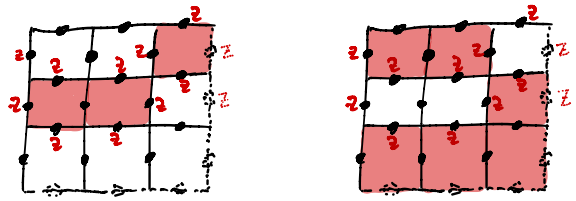
The plaquette operators are not all independent. If we take the product of all plaquettes, each qubit has two Z's acting on it, which cancel out:

$$\prod_{\text{all } p} B_p = \mathbb{1} \ . \tag{5.2}$$

So, one plaquette can be written as the product of all the other plaquettes:



A product of two neighboring plaquettes apply Z's twice on inner qubits which cancel out. So, if we take a product of any subset of plaquettes, we get a collection of loops of Z operators. There are exactly two ways of writing the same configuration of Z loops as a product of plaquettes:



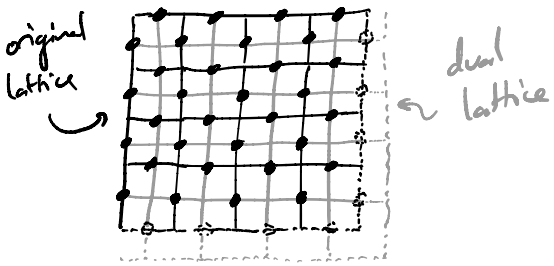
This shows that, if we remove one of the L^2 plaquette operators, the remaining are independent: No other product of plaquettes can create the same loops of Z operators. Therefore, *there are* $L^2 - 1$ *independent plaquette operators.*

Similarly, the product of all star operators applies X twice on each qubit, so

$$\prod_{\text{all } v} A_v = \mathbb{1} \ . \tag{5.3}$$

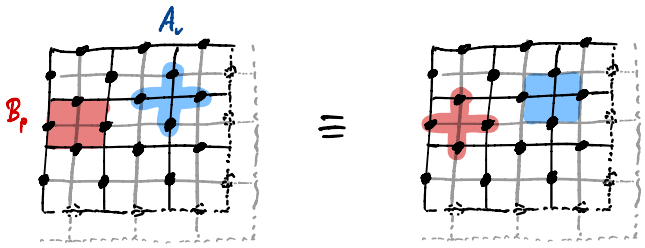
So the star operators are not all independent.

To visualize products of a subset of star operators, it is convenient to consider the ***dual lattice***. We associate a vertex to each face of the original lattice and a face to each vertex. Edges have their orientation flipped, so they connect neighboring vertices of the dual lattice:



Qubits remain on the edges; the edges are indeed shared between the original lattice and the dual lattice.

Star operators A_v on the original lattice are associated with plaquettes of the dual lattice, and plaquette operators B_p are associated with vertices on the dual lattice:



To avoid confusion, I'll keep referring to A_v as star operators and B_p as plaquette operators, and I'll stick to drawing them as they act on the original lattice.

Therefore, products of star operators A_v create a constellation of X operators that forms loops along the dual lattice edges, as we saw for plaquette operators on the original lattice. Applying a similar argument as for the plaquettes B_p, we find that there are $L^2 - 1$ independent A_v star operators.

Kitaev's toric code is the stabilizer code with stabilizer group generated by all the A_v's and B_p's:

$$\mathcal{S}_{\text{toric}} = \langle \{A_v\}, \{B_p\} \rangle \ . \tag{5.4}$$

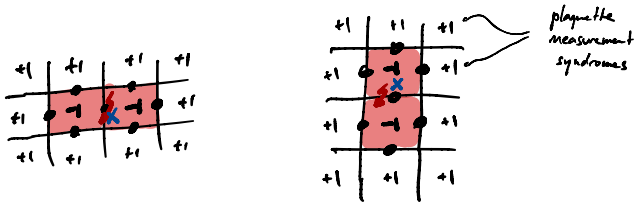
All stabilizer generators are either X-type (A_v) or Z-type (B_p), so Kitaev's toric code is a CSS code.

There are $r = 2(L^2 - 1)$ independent stabilizer generators and we have $n = 2L^2$ qubits. **Kitaev's toric code therefore encodes $k = n - r = 2$ logical qubits.**

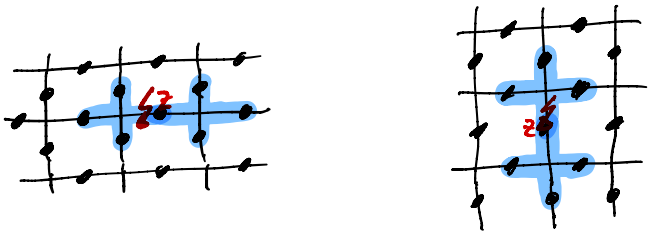
Before we can identify logical operators, it is useful to understand the effect of single-qubit errors on the syndromes.

5.2 Errors and logical operators in the toric code

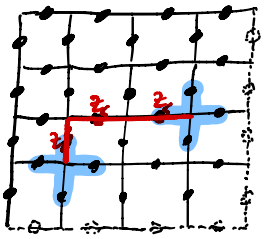
A single X error anticommutes with two neighboring plaquette operators, causing them to flip the sign of their outcomes when measured. (We can think of them as warning lights that light up to indicate the wrong outcome was observed at that location.)



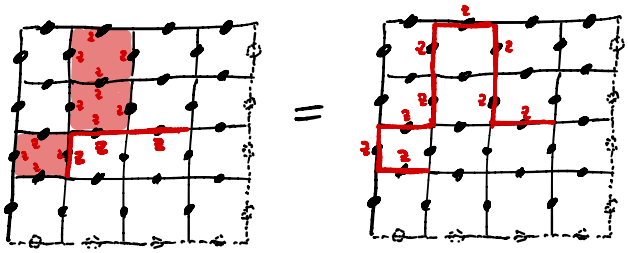
Analogously, a single Z error anticommutes with two neighboring star operators, causing them to "light up:"



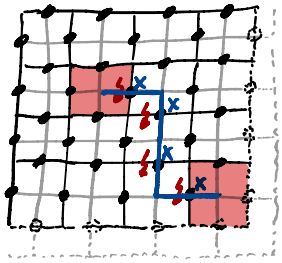
If we happen to have two or more neighboring Z errors, then only the stars at the endpoints of the error string light up:



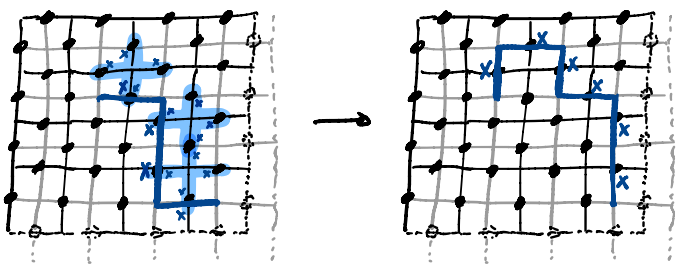
If we have a string of neighboring Z errors, the effect of multiplying the string by adjascent plaquette operators is to distort the shape of the string, without changing its endpoints:



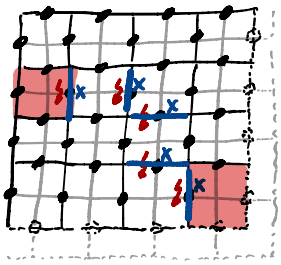
To understand the effect of strings of X errors, we consider the dual lattice. A string of X errors that connect neighboring edges of the dual lattice anticommutes with the two plaquette operators at the endpoints of the string, and commutes with all other plaquettes. When we measure the syndromes, these two plaquette operators "light up:"



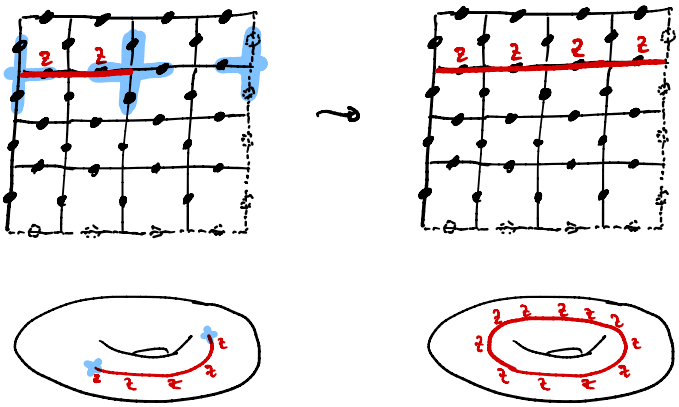
If we multiply a string of X's by star operators, we distort the string on the dual lattice, but we cannot change its endpoints:



To avoid confusion between the original lattice and the dual lattice, we'll often represent strings of X operators by highlighting the corresponding edges in the original lattice, giving a rugged path of facing edges:

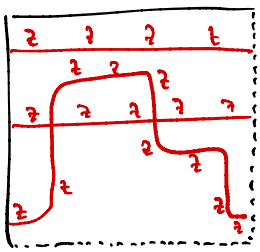


Suppose we extend a string of Z errors up until it wraps around the torus. (Remember, we have periodic boundary conditions.) Then the two endpoint star operators annihilate each other:

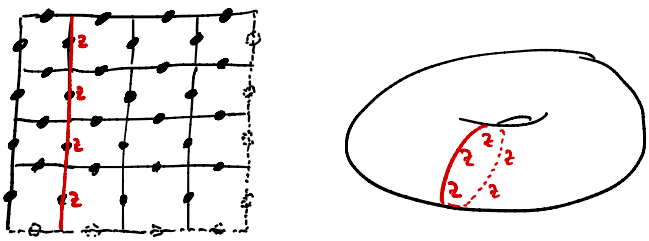


This Z string commutes with all the stabilizers. But it cannot be in the stabilizer group: Multiplying this Z string by any stabilizer can distort the Z string, but it cannot change the fact that the string wraps around the torus. The string is thus not stabilizer-equivalent to $\mathbb{1}$. It is a nontrivial logical operator. We can take it to be the $\overline{Z}_1$ logical operator, i.e., the Z operator of the first of our two logical qubits.

Any Z string that wraps once around the torus horizontally is stabilizer-equivalent to $\overline{Z}_1$:

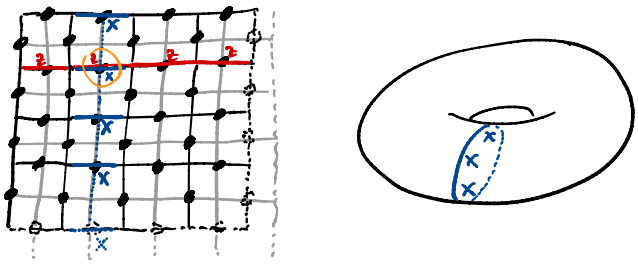


A string of Z's that wraps around vertically is another, independent logical operator:



Multiplying the string by stabilizers might move the string around, but it won't change the fact that the string wraps around vertically. This is another logical operator, which we can take to be $\overline{Z}_2$, the Z operator of the second logical qubit.

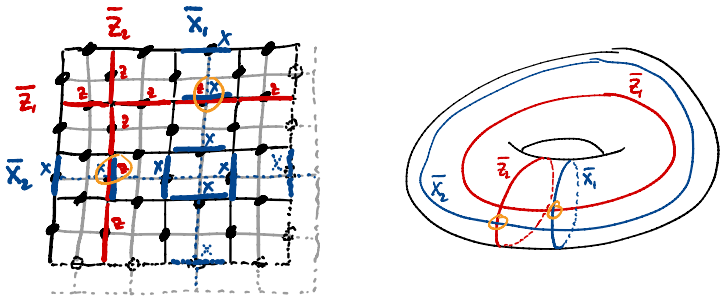
Now, consider an X-type string that wraps around vertically:



It is also a logical operator. Observe that it anticommutes with a horizontal string of Z's on the original lattice. These strings overlap on one site (highlighted in yellow), where the respective X and Z operators anticommute. We can take the vertical X-type string to be $\overline{X}_1$, the X operator of the first logical qubit.

Similarly, a horizontal X-type string is a logical operator, and can be chosen to represent $\overline{X}_2$.

In summary, the Pauli logical operators of the toric code are:



We don't have to worry about $\overline{Z}_1$ overlapping with $\overline{X}_2$, because they act on different sets of qubits: $\overline{Z}_1$ acts on qubits placed on *horizontal edges* and $\overline{X}_2$ acts on *vertical-edge* qubits. (These operators can be distorted to stabilizer-equivalent strings that overlap on some edges, for instance by distorting a vertical Z string to have overlap on some horizontal edges; yet such strings necessarily overlap on an even number of sites, ensuring that any two representatives of the logical operators $\overline{Z}_1$ and $\overline{X}_2$ always commute.) A similar argument holds for $\overline{Z}_2$ and $\overline{X}_1$.

The *distance* of the toric code is $d = L$: A nontrivial logical operator needs to wrap around the torus, ensuring its weight must be at least L.

A convenient picture emerges if we think of flipped stabilizer generator outcomes (the "warning lights" that turned on) as a sort of "particles" in a curious phase of matter with some exotic properties. To construct this picture, we begin with the following Hamiltonian:

$$H = -\sum_v A_v - \sum_p B_p \ . \tag{5.5}$$

An error-free code state of the toric code is a simultaneous $+1$-eigenstate of all A_v and B_p operators; it therefore minimizes the energy of each individual Hamiltonian term and is

therefore a ground state of H. The ground space of H is degenerate, since the toric code has multiple distinct code states.

Code states that are affected by a Pauli error E (i.e., states of the form $E|\psi\rangle$ with $|\psi\rangle \in \mathcal{C}$, $E \in \mathsf{P}_n$) are also simultaneous eigenstates of all the A_v's and B_p's; they have eigenvalue -1 for all stabilizer generators that anticommute with E and they have eigenvalue $+1$ for all remaining stabilizer generators. Such states are also eigenstates of H. Their energy, compared to the ground space, is precisely determined by the number of stabilizer generators whose signs are flipped: Each time we flip a stabilizer generator's sign, we increase the energy by 2. These states are all the possible eigenstates of H, as we can see by remembering that the error spaces associated with all possible Pauli errors in a stabilizer code fill up the entire Hilbert space. Because each stabilizer generator sign flip increases the energy by a fixed amount, we can think of them as elementary excitations of H.

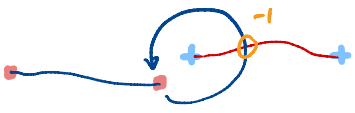
We identify stabilizer generators with a flipped sign as ***excitations of quasiparticles*** of H.

There are two kinds of quasiparticles, associated with flipping either a B_p term (giving rise to a plaquette-type excitation) or a A_v term (representing a star-type excitation):

Quasiparticles of each kind always come in pairs. This reflects the dependence of the plaquette and the star operators (recall $\prod_v A_v = \mathbb{1} = \prod_p B_p$), which implies that there can only ever be an even number of A_v's and an even number of B_p's whose signs are flipped.

We can think of creating, moving, and annihilating these quasiparticle excitaitons:

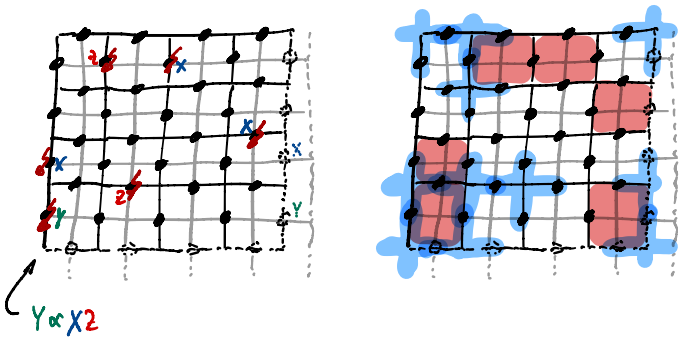
- A single-qubit X or Z error *creates* a pair of quasiparticle excitaitons;
- We can *move* one quasiparticle around by applying a single-qubit X or Z at one of the edges adjacent to the excitation;
- Two quasiparticles of the same kind *annihilate* if they are moved to the same location;
- One quasiparticle type, when moved along a loop that encloses a single quasiparticle of the other type, picks up a -1 phase:

  These particles thus exhibit exotic particle statistics; particles with such exotic statistics are often referred to as *anyons*. In the toric code, both kinds of excitations behave individually like bosons, since an exchange of two particles of the same kind does not give rise to any nontrivial phase. But star-type and plaquette-type excitations behave mutually like fermions: Wrapping one around the other gives rise to a -1 phase.

5.3 Decoding syndromes and correcting errors in the toric code

Let us suppose that X and Z errors can happen at random, independently on each qubit. Through exposure to the noise, the qubits suffer a random pattern of error operators (left below). Of course, we don't know which errors happened. We'll only see some syndromes corresponding to a configuration of stabilizer generators that "light up" (right below):



Recall that a -1 in the syndromes tell us that the corresponding stabilizer generator anticommutes with the total Pauli error the system suffered. We still need to find out what error happened, up to stabilizers, in order to restore the error-free state. Remember, this process is referred to as *decoding the syndromes*.

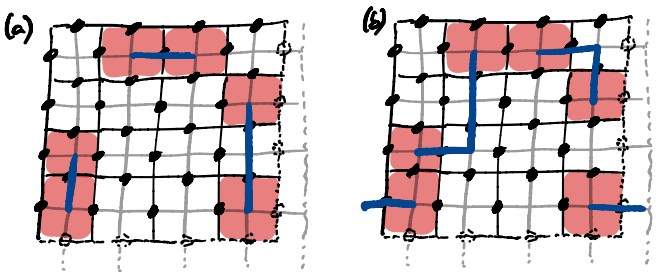
As in any CSS code, we can correct separately X-type errors (which are caught by Z-type stabilizers) and Z-type errors (caught by X-type stabilizers).

Remember that star-type and plaquette-type syndrome excitations always come in pairs. I.e. there is always an even number of star excitations, and an even number of plaquette excitations.

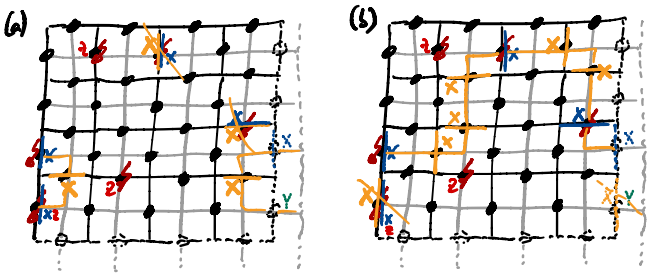
Flash exercise: Wait up, how come the above statement doesn't contradict the general property of stabilizer codes we saw in Chapter 4 that for any syndrome vector, we can always find some Pauli operator that causes this syndrome?

Decoding the toric code amounts to *pairing up excitations* of the same type and applying strings of X's or Z's to restore the state back to the code space, where all syndrome outcomes are $+1$.

Here are two possible ways of pairing up plaquette excitations to correct X errors in our example above:



In each case, applying X's along the strings to remove the excitations leads to the following operators being applied in total. The correction operation is multiplied with the original error to obtain the overall operation that is applied to the original state:



The yellow X's above are the operations that are applied as a correction operation; the yellow lines represent the strings of X's resulting from the original error multiplied with the correction operation.

In the case (a), two X errors are directly eliminated, and the four remaining X operators form a stabilizer which acts trivially on the logical qubits. *All X-type errors were successfully corrected by choosing this pairing.*

In the case (b), the X operators we introduced as a correction operation eliminate one X-type error (turning the $Y \propto XZ$ error into a pure Z error) but also cause a string of X's to wrap around the torus. *By choosing this pairing, the errors and the ensuing decoding cause a logical error, because $\overline{X}_2$ is accidentally applied on the logical qubits.*

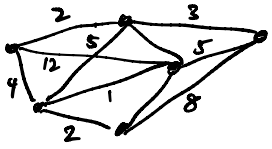
The decoder's task is to pair up excitations while minimizing the risk of causing a logical error.

Recall that optimal decoding requires identifying the most likely *class of stabilizer-equivalent errors* compatible with the syndromes (maximum likelihood decoding). This is a hard computation in general.

However, it turns out that a simpler decoder that simply selects the most likely error usually performs pretty well. For simple noise models, this corresponds to pairing up excitations of the same type by applying the least possible number of operators (minimum weight decoding). In our example, a decoder should prefer the pairing (a), which corresponds to applying four X operators, to the pairing (b), which requires six X operators.

We'll carry out a more detailed analysis of the relative performance of these two strategies later on.

An efficient decoder for the toric code comes from an algorithm in graph theory known as ***minimum-weight perfect matching***. Suppose we are given a graph with weighted edges:



A *matching* of the graph is a subgraph where every vertex is connected to at most one edge. The matching is *perfect* if each vertex is connected to an edge. An algorithm by Edmonds (1965) finds efficiently the perfect matching with the minimal sum of weights on the remaining edges.

We can use minimum-weight perfect matching to pair up quasiparticle excitations of the toric code efficiently. First, we build a graph where we associate one vertex to each observed

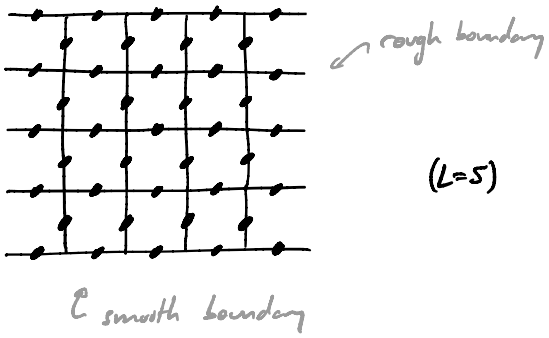
quasiparticle excitaiton, and we connect all vertices to one another with weighted edges whose weights are the distance between the excitation locations on the lattice (taxi-cab distance). The minimum-weight perfect matching algorithm then provides the error of minimal weight that is compatible with the observed syndromes.

Once the excitations are paired up, we can apply a corresponding string of X's or Z's to remove the excitation pairs.

5.4 Planar surface codes: introducing boundaries

Laying out qubits on a physical torus is highly impractical. Can we get away with a flat, two-dimensional surface?

Consider the following chunk of the square lattice:

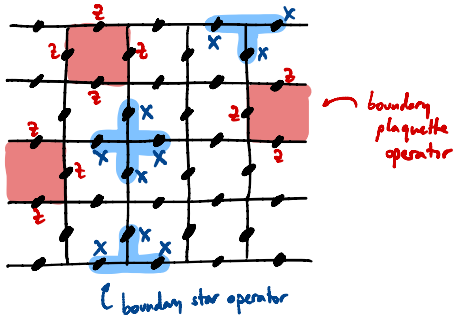


Observe that we picked the boundaries in a particular fashion: "open" or ***rough*** boundaries on the left and the right sides, and "closed" or ***smooth*** boundaries on the top and bottom sides.

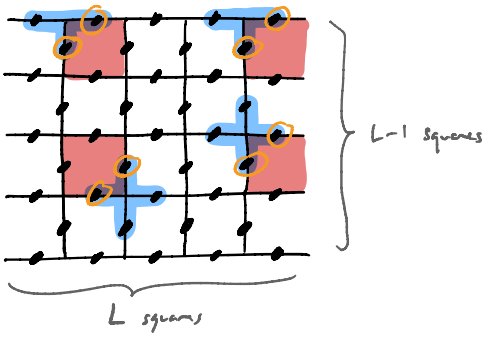
As for the toric code, there are two types of qubits: Those on horizontally oriented edges, forming a $L \times L$ grid, and those on vertically-oriented edges, forming a $(L-1) \times (L-1)$ grid. In total, we have n qubits with

$$n = L^2 + (L-1)^2 = 2L^2 - 2L + 1 \ . \tag{5.6}$$

In the bulk of the lattice patch, we define the star operators and plaquette operators as for the toric code. At the boundaries, we simply omit one X operator for star operators on smooth boundaries and we omit a Z operator for plaquettes on a rough boundary:



As before, all star and plaquette operators, including those on the boundaries, *commute*. Again, if a star and a plaquette operator overlap, they do so necessarily on an even number of qubits:



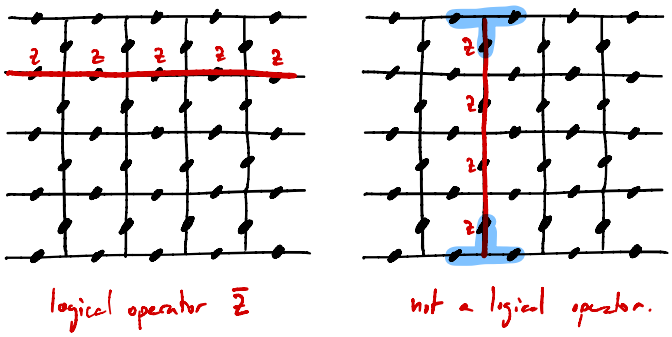
In contrast to the toric code on the torus, all star operators and plaquette operators are now independent; there are no redundant checks. There are $L(L-1)$ plaquette operators and $L(L-1)$ star operators. In total, we have r independent stabilizer generators with

$$r = 2L^1 - 2L \ . \tag{5.7}$$

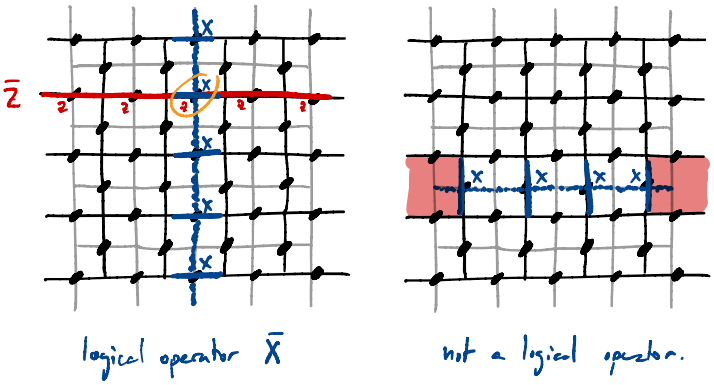
The number of encoded logical qubits is therefore

$$k = n - r = 1 \ . \tag{5.8}$$

A horizontal string of Z's is a logical operator, which we define as the logical Z operator $\overline{Z}$ of the encoded logical qubit. In contrast, a vertical string of Z's is no longer a logical operator, since it anticommutes with the boundary star operators:



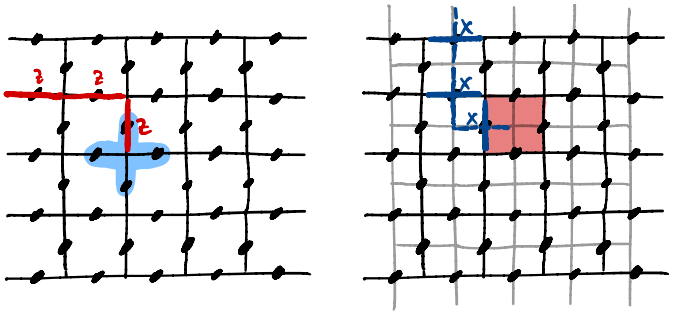
A vertical string of X operators following a path on the dual lattice from top to bottom is a logical operator. It anticommutes with $\overline{Z}$, so we define it to be the logical X operator $\overline{X}$ of the encoded logical qubit. A horizontal string of X's is no longer a logical operator:



(Note that in the dual lattice, rough and smooth boundaries are interchanged.)

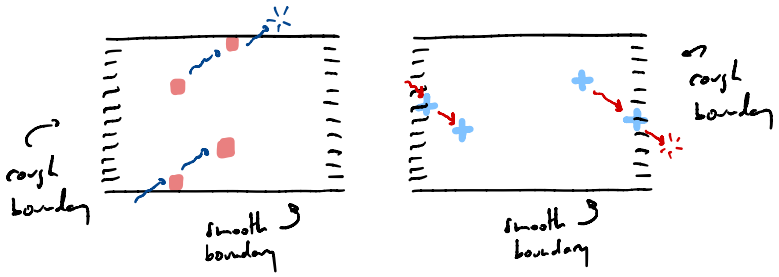
In the bulk (away from the boundaries), star and plaquette excitations behave as in the toric code: A single Z or X error *produces a pair* of excitations, the latter can be *moved* apart by applying further Z or X operators, and a pair of excitations *annihilate* whenever they are brought at the same location.

If a string of Z's ends on a rough boundary, only a single star-type excitation appears. Similarly for strings of X's ending on a smooth boundary. Excitations no longer necessarily appear in pairs!

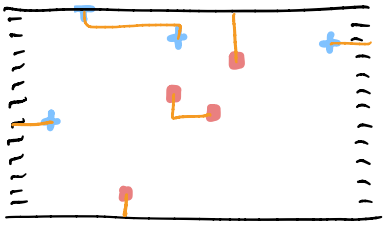


Therefore, there are two additional operations we can perform with our quasiparticle excitations:

- We can "pull" excitations out of a boundary of the corresponding type, by applying a single Pauli on that boundary. Applying a Z operator on a rough boundary produces a single star-type excitation; applying an X operator on a smooth boundary produces a single plaquette-type excitation;
- We can annihilate a single star-type (plaquette-type) excitation by moving it through a rough boundary (smooth boundary).



A decoder for a surface code patch with boundaries may choose to pair up excitations together, or to pair up an individual excitation with the corresponding boundary type:



A possible pairing of the excitations, yielding a possible correction operation, is drawn above in yellow.

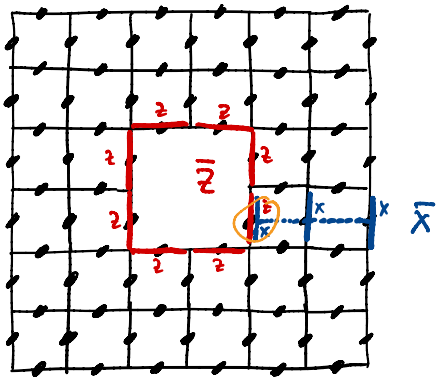
Minimum-weight perfect matching yields an efficient algorithm to pair up excitations for decoding, as for the toric code.

5.5 A surface code with punctures

The ability of the toric code to host two qubits rather than a single qubit is related to its topology: We can find strings forming logical operators that wrap around in the right way to get the desired anticommutation relations.

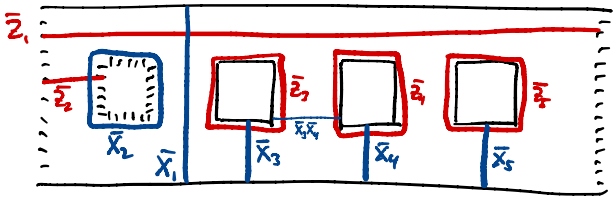
A way of generating a nontrivial topology on a flat surface is to poke holes in the surface. Can we encode more logical qubits on the planar surface code in this way?

Consider the following patch with smooth boundaries all around, and in which we carved out a hole:



This surface also encodes a single qubit, where the logical Z operator is a string of Z's that wrap around the hole, and the logical X operator is a string of X's connecting the inner boundary with the outer boundary.

We can combine different boundary types, and poke multiple holes, to encode multiple qubits:



Two pairs of boundary types, as well as each hole, each contribute one logical qubit.

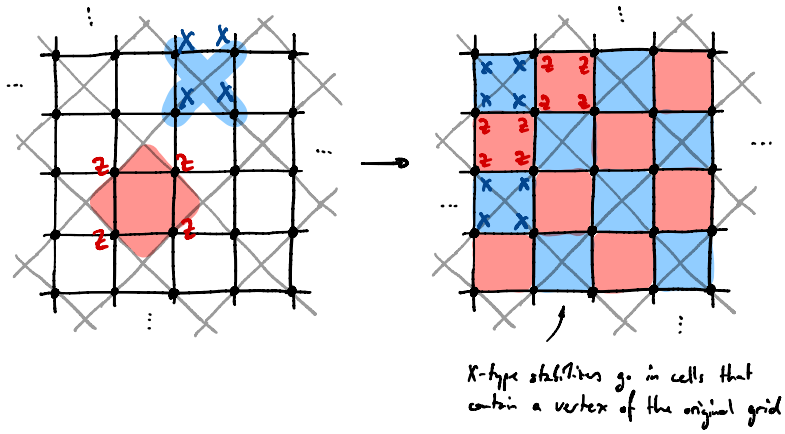
Remember: Strings can be deformed (and their endpoints moved along a boundary) by multiplying with stabilizers. For instance, $\overline{X}_3\overline{X}_4$ in the example above can be represented equivalently as a simple string of X's connecting the holes associated with qubits 3 and 4.

The distance of the code is the length of the shortest logical operator. It might be, for instance, the separation between two holes.

Puncturing holes through a big patch of a surface code is an appealing technique for encoding multiple logical qubits and performing gates on them. However, it turns out that simply encoding multiple qubits in different, independent single-qubit surface code patches is more efficient, thanks to clever techniques for performing gates between independent single surface code patches that we'll cover later as we discuss fault tolerance.

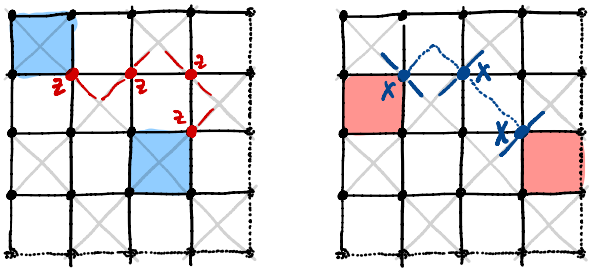
5.6 Rotated surface code

We can represented the surface code in a way that exhibits more symmetry between the X-type and Z-type stabilizers as follows. We rotate the surface code by 45° and place the qubits on the *vertices* of a new square grid:

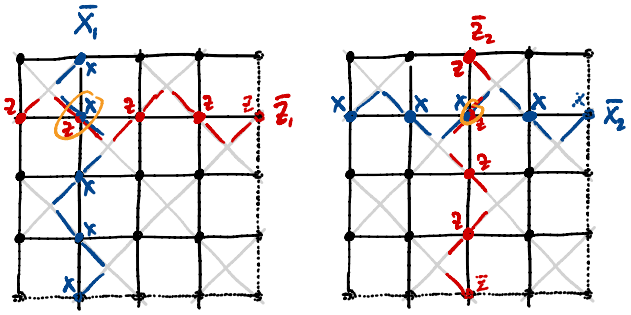


For the moment, let us suppose we have periodic boundary conditions in the rotated square lattice, with the effect of placing the lattice on the torus. The code is still for now the same code as the toric code, we're simply viewing it from a different angle. Observe that the size of the new lattice grid is forced to be *even* because of the checkerboard pattern.

As in the surface code, Z-type (respectively, X-type) excitations are produced in pairs, and can be moved around, by applying X (respectively, Z) operators:

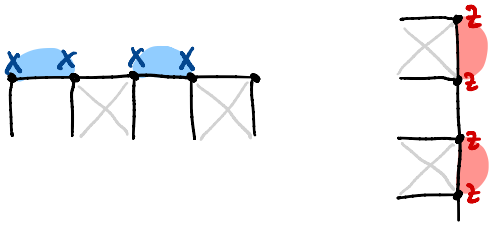


Extending an X string or a Z string to wrap around the torus, in either direction, gives a logical operator:

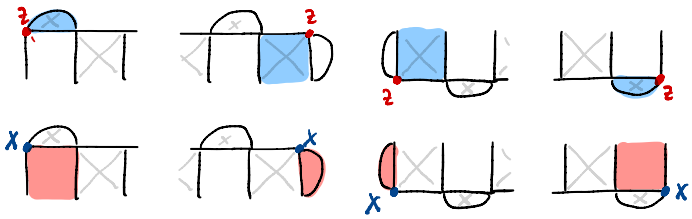


The lattice size being even ensures commutation of $\overline{Z}_1$ with $\overline{X}_2$ and of $\overline{Z}_2$ with $\overline{X}_1$.

Now we consider the rotated surface code with *boundaries*, giving up the periodic boundary conditions. (This is the setting which is commonly referred to simply as the "rotated surface code.") To define a rotated surface code patch, we need suitable boundary stabilizer terms that can create or annihilate individual excitations. We define the following boundary stabilizers:



The boundary terms enable the creation and annihilation of single excitations of either type at the *corners* of the grid, while enabling them to propagate along each boundary:



5.7 Further reading

The surface code has been reviewed on numerous occasions in a variety of lectures, we point to a selection of references here that is far from comprehensive. Additional references are also listed in the Kitaev Surface Code entry of the Error Correction Zoo.

A few key, seminal papers that discuss fundamental properties of the surface code, and which served as inspiration for this chapter, include a seminal paper by Kitaev, who discovered the toric code [35], a brief and highly insightful explanation of the toric and planar surface codes by Bravyi and Kitaev [36], and a study of the reliability of the surface code by Dennis *et al.* [37].

For some more useful and insightful material we further point to an extensive review paper by Fujii [5], a book chapter by Bombín [38], lecture notes by Dan Browne [13], presentations by Kubica and Haah and Boulder Summer Schools, as well as an interactive blog post by Pesah [39].

The surface code was demonstrated experimentally in superconducting qubits [40] and forms the basis of proposals for fault-tolerant quantum computation on such platforms [41, 42].

The surface code draws a close link between quantum error correction and condensed matter physics. We've only hinted towards this connection in this chapter and we refer to a swath of great references for the interested reader: a book by Zeng *et al.* [43], lectures by Levin and Nayak at a Boulder Summer School, Fujii's review paper [5], and many more.

Keep an eye for additional references in the upcoming chapters as explore further topics surrounding topological codes and as we begin studying the reliability of the surface code.

Chapter 6

More topological codes: homology and colors

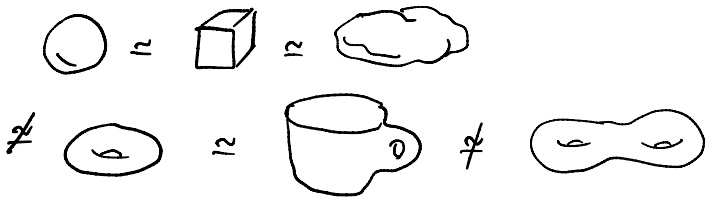
In this chapter, we'll extend the ideas at the root of the surface code in two main directions. First, we'll explain a general construction of CSS codes based on the mathematical concept of *homology*. Second, we'll construct another important class of topological codes, named *color codes*.

6.1 Topological codes from homology

We'll see that the surface code generalizes naturally to arbitrary surfaces, by invoking the mathematical toolbox of *homology*.

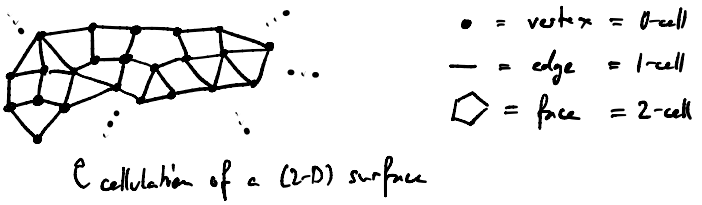
The connection between error correction and homology that we'll develop here extends naturally beyond topological codes to general CSS codes, in particular to quantum LDPC codes that we'll study later in this course.

Homology concerns the topology of objects, i.e., properties of objects that are invariant under the application of a suitably continuous, invertible function.



We'll now introduce some elements of homology in the simplest form we need to construct qubit quantum error-correcting codes. Specifically, we'll restrict our attention to so-called $\mathcal{Z}_2$ *cellular homology*.

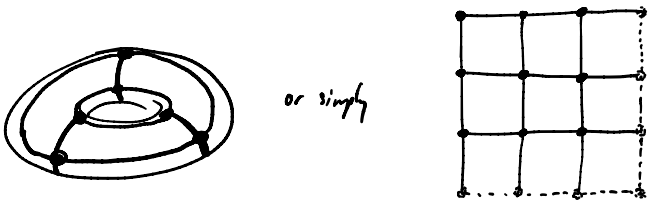
A ***cellulation*** of a surface (or hypersurface) is a decomposition of the surface into vertices, edges connecting vertices, polygons enclosing edges, and so on, up to the dimension of the object.



Vertices are 0-dimensional objects, and we call them ***0-cells***. Edges, faces, etc. are ***1-cells***, ***2-cells***, etc.

Observe that in any cellulation of a surface, 1-cells are surrounded by 0-cells (the vertices at the endpoints of an edge), 2-cells are surrounded by 1-cells (the edges around each polygon), etc. Indeed, a central concept in homology is to study how n-dimensional objects can or cannot be associated with the boundary of an $(n+1)$-dimensional object.

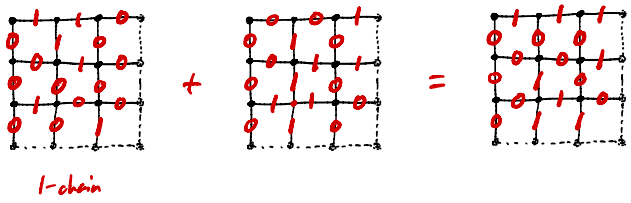
For example, a cellulation of the torus might look like this:



We define a n***-chain*** as a mapping of each n-cell of a cellulation to $\mathcal{Z}_2 = \{0, 1\}$. Equivalently, an n-chain is any subset of n-cells.

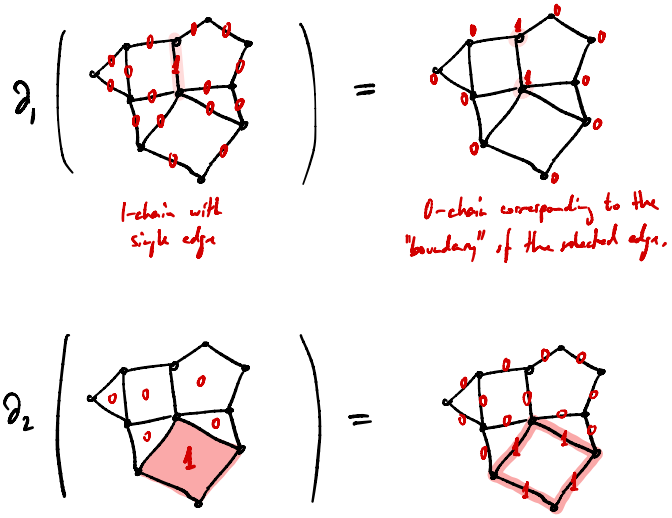
A 0-chain is a subset of vertices, a 1-chain is any subset of edges, etc. Despite their name, n-chains do not have to resemble a usual chain in any way.

The n-chains form a group, for each n, in which we simply add the $\mathbb{Z}_2$ values in the n-chain modulo 2. For example:

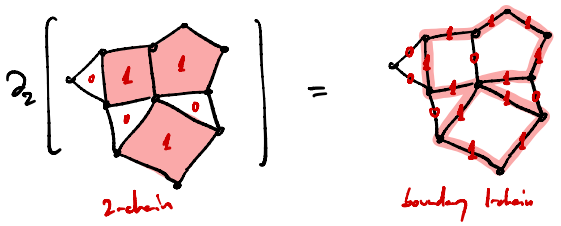


Furthermore, each n-chain is its own inverse.

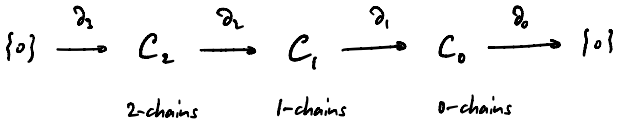
The n-boundary map ∂_n is a linear map from n-chains to $(n-1)$-chains defined as follows. To an n-chain that is a single-element subset of n-cells, ∂_n associates the $(n-1)$-chain corresponding to the subset of $(n-1)$-cells that immediately surround that n-cell in the cellulation. The map ∂_n extends to all n-cells by linearity under addition modulo two.



The map ∂_n coincides with our intuition of a "boundary" of strings of edges or collections of faces:



We define the set of (-1)-cells to be the empty set $\emptyset$, such that there is (by convention) only a single possible (-1)-chain, the null chain 0. We take ∂_0 to map any 0-chain to the null chain 0. Similarly, the set of $(D+1)$-cells is taken to be $\emptyset$, where D is the dimension of the cellulated (hyper)surface, and $\partial_D(0)$ embeds the only possible $(D+1)$-chain 0 into the space of D-chains as the null D-chain 0. We obtain a sequence of spaces connected by the boundary maps, which we refer to as a *chain complex*:



An n-chain c whose boundary is trivial, i.e., $\partial_n c = 0 \equiv$ the null $(n-1)$-chain $\emptyset$, is called an ***n*-cycle**. The sum of two n-cycles is an n-cycle, so n-cycles *form a group* which we denote by

$$Z_n = \ker \partial_n \ . \tag{6.1}$$

An n-chain c that happens to be the boundary of some $(n+1)$-chain c', i.e., $\exists\, c' \ : \ \partial_{n+1} c' = c$, is called an ***n*-boundary chain**, or ***n*-boundary**. These also *form a group* which we denote by

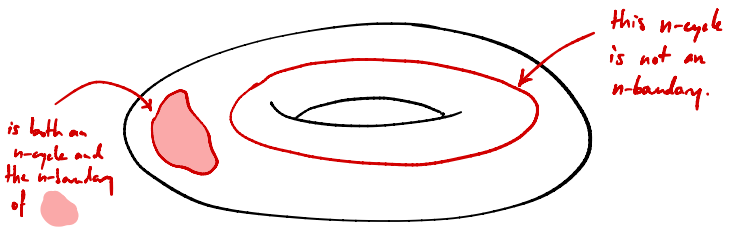
$$B_n = \mathrm{Im}\, \partial_{n+1} \ . \tag{6.2}$$

A fundamental feature of homology is the following: *Every n-boundary is a n-cycle.* Equivalently, applying the boundary operator twice always gives a trivial map:

$$\boxed{\partial_{n-1}\partial_n = 0} \ . \tag{6.3}$$

"The boundary of a boundary is trivial."

On the other hand, not all n-cycles need be n-boundaries. Intuitively, such n-cycles avoid enclosing an $(n+1)$-dimensional surface. A n-cycle representing a subset of 1-cells wrapping around the torus, for example, is not an n-boundary:



The information about which n-cycles are not n-boundaries reveals interesting information about the topology of a surface. For instance, on the flat two-dimensional plane, all n-cycles are, in fact, n-boundaries. But this is not the case on the torus.

The ***n-th homology group*** is defined as

$$H_n = Z_n/B_n = \ker \partial_n / \operatorname{Im} \partial_{n+1} \ . \tag{6.4}$$

It is the group of equivalence classes of n-cycles, up to addition by n-boundaries. (Two n-cycles are deemed 'equivalent' if one can be obtained by adding an n-boundary to the other.)

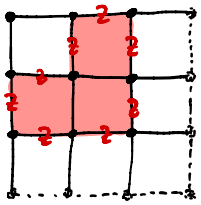
We started from the cellulation of a surface, for which it turns out that we always have the fundamental property of homology, $\partial_{n-1}\partial_n = 0$. Actually, we could consider more general chain complexes that do not have to originate from the cellulation of a surface. We lose our original geometric interpretation; the C_i's become abstract spaces and the boundary maps are linear maps connecting them. In these more general cases, the boundary operator ∂_n defines how n-cells are related to $(n-1)$-cells, and it must obey the fundamental property $\partial_{n-1}\partial_n = 0$.

We can draw inspiration from the surface code to start inferring a general way of constructing qubit error-correcting codes starting from the cellulation of an arbitrary surface:

1. (1) Fix a cellulation of a $2D$ surface with 0-cells, 1-cells, 2-cells, and n-boundary maps ∂_n;
2. (2) Place a qubit on each 1-cell;
3. (3) Define a *plaquette operator* B_p for each 2-cell p as Pauli-Z operators acting on the qubits of its boundary 1-cells:

$$B_p = \prod_{i \in \partial p} Z_i \ . \tag{6.5}$$

Any product of plaquette operators simply corresponds to a 2-chain, and represents a product of Z operators on the qubits on the 1-cells that make up the boundary of the 2-chain:



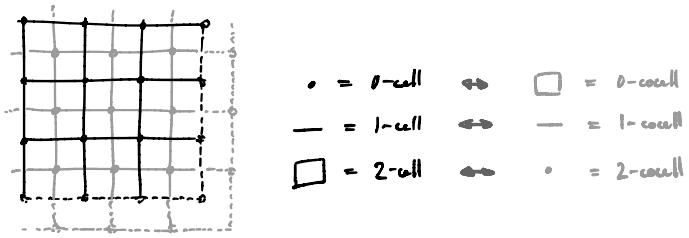
In these diagrams, the heavy dots ("$\bullet$") placed at the vertices of the lattice represent 0-cells and not qubits. Qubits are placed on the 1-cells, i.e., the edges of the lattice. I won't draw them explicitly to avoid overloading the diagrams.

Any string of Z's that corresponds to a 1-boundary is therefore a product of plaquettes, so is a Z-type stabilizer and acts trivially on the code space.

We'd like to associate a star operator to each 0-cell (vertex), but it's not immediately obvious how to define what we mean by the "edges that are connected to that vertex."

It turns out that X stabilizer generators are naturally described by a "dual" picture of homology known as *cohomology*.

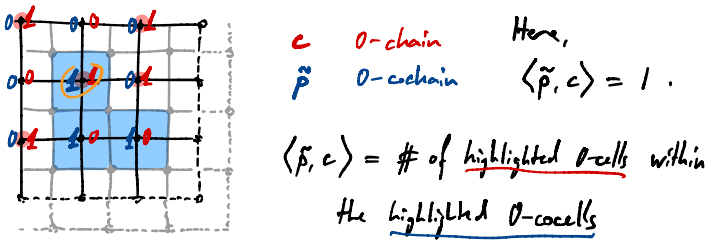
To each n-cell is associated a ***n*-cocell**. The cocells represent the vertices, edges, and faces of the dual lattice:



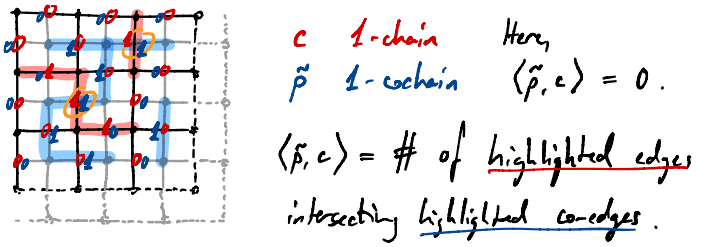
A ***n*-cochain** is a linear functional on n-chains. It can be specified as an assignment $\tilde{p}$ of n-cocells to $\mathbb{Z}_2$ where the functional is determined via the binary inner product modulo 2:

$$\text{functional on } n\text{-chains} \quad \equiv \quad \langle \tilde{p}\,,\,(\cdot)\rangle\ . \tag{6.6}$$

A 0-cochain $\tilde{p}$ can be specified by a subset of 0-cocells or as an assignment of 0-cocells to $\mathbb{Z}_2$; its action on a 0-chain c is computed as the parity of the number of locations of 0-cell–0-cocell pairs on which both $\tilde{p}$ and c take the value 1:



A 1-cochain $\tilde{p}$ can be specified by a subset of 1-cocells or as an assignment of 1-cocells to $\mathbb{Z}_2$; its action on a 1-chain c is computed as the parity of the number of locations of 1-cell–1-cocell pairs on which both $\tilde{p}$ and c take the value 1:

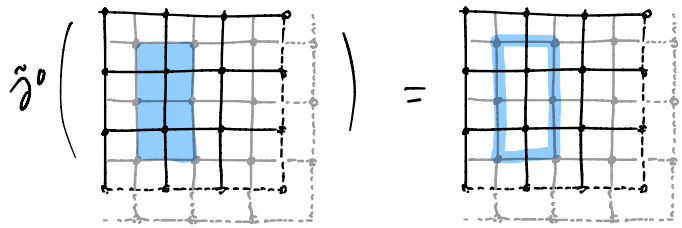


The ***n*-coboundary map** $\tilde{\partial}^n$ assigns to an n-cochain $\tilde{p}$ the $(n+1)$-cochain $\tilde{\partial}^n\tilde{p}$ which acts on any $(n+1)$-chain c in the same way as $\tilde{p}$ would act on the boundary $\partial_{n+1}c$ of c:

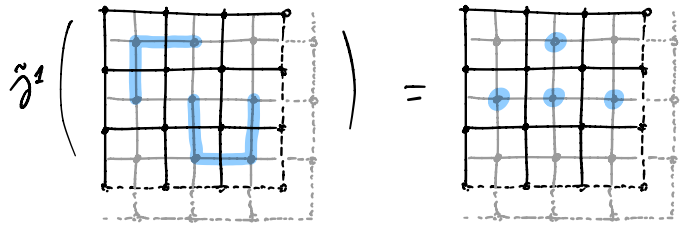
$$\langle \tilde{\partial}^n\tilde{p}\,,\ c\rangle = \langle \tilde{p}\,,\ \partial_{n+1}c\rangle\ . \tag{6.7}$$

Identifying n-chains and n-cochains with linear vector spaces over $\mathbb{F}_2$ we find that $\tilde{\partial}^n = \partial_{n+1}^T$, where T denotes the transpose in the standard basis.

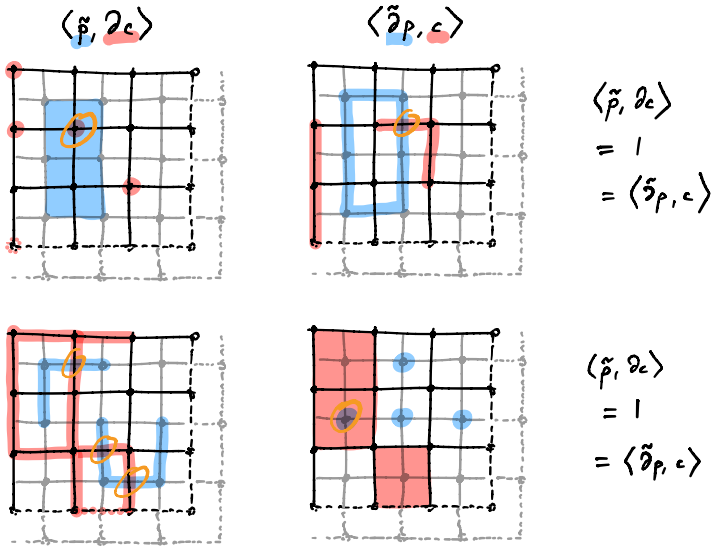
The coboundary map identifies the intuitive notion of a boundary on the dual lattice. The coboundary of a 0-cochain is its enclosing 1-cochain:



The coboundary of a 1-cochain is its enclosing 2-cochain:



We can visually convince ourselves of this property by comparing $\langle \tilde{\partial}^n \tilde{p}\,,\ c\rangle$ with $\langle \tilde{p}\,,\ \partial_{n+1} c\rangle$ for the examples above and an example $(n+1)$-chain c:



We can again define the group of ***n*-cocycles** $\tilde{Z}^n$ and the group of ***n*-coboundaries** $\tilde{B}^n$ as:

$$\tilde{Z}^n = \ker \tilde{\partial}^n \ ; \tag{6.8}$$

$$\tilde{B}^n = \mathrm{Im}\, \tilde{\partial}^{n-1} \ . \tag{6.9}$$

The ***n-th cohomology group*** is

$$\tilde{H}^n = \tilde{Z}^n / \tilde{B}^n = \ker \tilde{\partial}^n / \operatorname{Im} \tilde{\partial}^{n-1} \ . \tag{6.10}$$

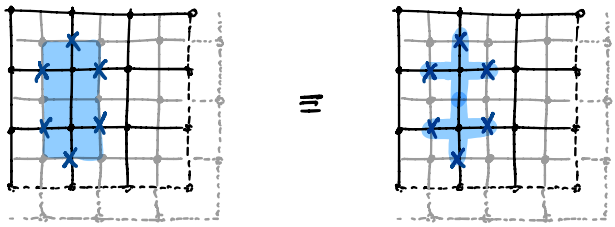
I'll keep the tilde '˜' in the notation for clarity. Some references simply use a superscript (H^n) instead of a subscript (H_n) to distinguish the cohomology groups from the homology groups.

We can thus complete our general homology-based code construction. Recall that qubits are placed on 1-cells, which directly correspond to 1-cocells.

1. (4) Define a star operator $A_{\tilde{v}}$ for each 0-cocell $\tilde{v}$ as Pauli-X operators acting on the qubits of its coboundary 1-cocells:

$$A_{\tilde{v}} = \prod_{i \in \tilde{\partial} \tilde{v}} X_i \ . \tag{6.11}$$

Any product of star operators simply corresponds to a 0-cochain and represents a product of X operators on the qubits on the 1-cocells that make up the coboundary of the 0-cochain:



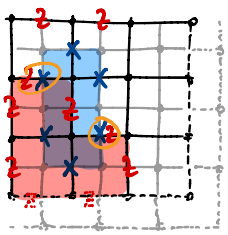
It is instructive to check that the A_v's and the B_p's commute, and to understand why this property is guaranteed. A product of plaquettes is represented by a 2-chain w. A product of star operators is represented by a 0-cochain $\tilde{c}$. The plaquettes act with Z's on the qubits lying on $\partial_2 w$. The star operators act with X's on the qubits lying on $\tilde{\partial}^0 \tilde{c}$. The number of qubits on which we have both an X and a Z applied will determine if the operators commute (if it is an even number) or anticommute (if the number is odd). This parity can be computed as the inner product

$$\langle \tilde{\partial}^0 \tilde{c}, \partial_2 w \rangle = \langle \tilde{c}, \partial_1 \partial_2 w \rangle = 0$$

definition of $\tilde{\partial}_n$

fundamental property of the boundary operator : $\partial_n \partial_{n+1} = 0$

So any product of $A_{\tilde{v}}$'s commutes with any product of B_p's as an immediate consequence of the fundamental property of homology that $\partial_n \partial_{n+1} = 0$.



Again, any 1-coboundary cochain corresponds to a product of star-type stabilizers and acts trivially on the code space.

To find nontrivial logical operators, consider any Pauli operator $P \propto X^{\tilde{x}} Z^{z}$ that we represent with a pair $(\tilde{x}|z)$ of a 1-cochain $\tilde{x}$ and a 1-chain z in binary symplectic notation. For P to commute with all B_p, we demand that $\langle \tilde{x}, \partial p \rangle = 0$ for all p. With $\langle \tilde{x}, \partial p \rangle = \langle \tilde{\partial} \tilde{x}, p \rangle = 0$ for all p, we find $\tilde{x} \in \ker \tilde{\partial}$, so $\tilde{x}$ *must be a 1-cocycle*. Similarly, for P to commute with all $A_{\tilde{v}}$'s, we need $0 = \langle \tilde{\partial} \tilde{v}, z \rangle = \langle \tilde{v}, \partial z \rangle$ for all $\tilde{v}$, so z must be a 1-cycle.

If z is a 1-boundary, then Z^z is a product of B_p-type stabilizers and acts trivially on the code space. But if it is a 1-cycle that is not a 1-boundary, then it is a nontrivial logical operator since it is not stabilizer-equivalent to $\mathbb{1}$.

Logical Z operators are determined by the homology group H_n.

If $\tilde{x}$ is a 1-coboundary, then $X^{\tilde{x}}$ is a product of $A_{\tilde{v}}$'s and acts trivially on the code space. Similarly to the Z's, 1-cocycles that are not 1-coboundaries are nontrivial logical operators.

Logical X operators are determined by the cohomology group $\tilde{H}^n$.

One can check it is always possible to find a set of generators of H_n and $\tilde{H}^n$ that obey the correct commutation/anticommutation relations to be logical qubit X and Z operators.

A summary of the homological construction of a CSS code is presented in Fig. 6.1.

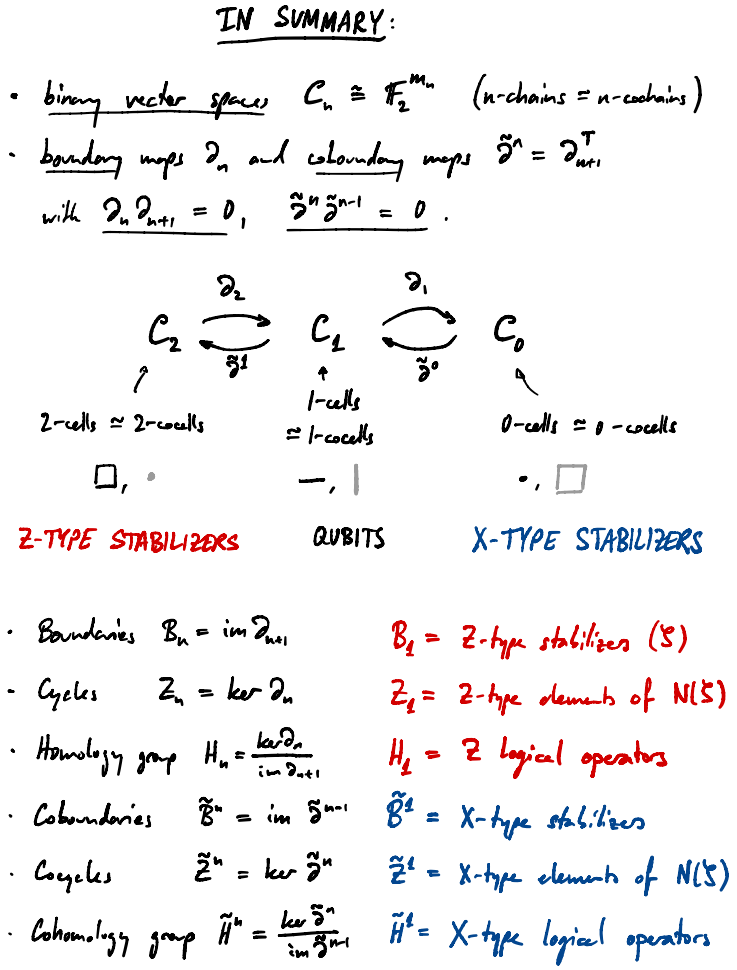


Figure 6.1: Summary of the construction of a CSS code from a homological chain complex.

The CSS code defined in this way corresponds to the choices of classical codes with parity check matrices

$$H_Z = \partial_2^T \ ; \qquad\qquad H_X = (\tilde{\partial}^0)^T = \partial_1 \ . \tag{6.12}$$

Indeed, the Z-type stabilizer generators correspond to $B_1 = \operatorname{Im} \partial_2$ which are spanned by the columns of ∂_2, i.e. the rows of ∂_2^T which we may thus take as H_Z. Similarly, the X-type stabilizers $\tilde{B}^1 = \operatorname{Im} \tilde{\partial}^0$ are spanned by the rows of $(\tilde{\partial}^0)^T = \partial_1$, which we may take as H_X.

The property $H_X H_Z^T = 0$ required by the CSS construction is automatically satisfied since

$$H_X H_Z^T = \partial_1 \partial_2 = 0 \ , \tag{6.13}$$

by the fundamental property of boundary maps in homology.

The distance of a homologically-constructed CSS code depends on the particular cellulation used (e.g., the lattice size used to cellulate the torus), and does not only depend on the topology of the surface.

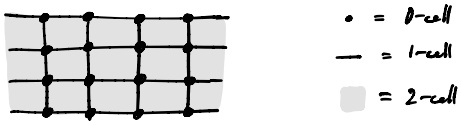
6.2 Some homology-constructed CSS codes

While we started off with the cellulation of a surface, all we need to apply the above construction of a CSS code based on homology is a *chain complex*

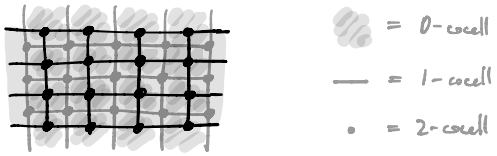
$$\mathcal{C}_2 \xrightarrow{\partial_2} \mathcal{C}_1 \xrightarrow{\partial_1} \mathcal{C}_0 \ , \tag{6.14}$$

with the crucial property that $\partial_1 \partial_2 = 0$. A cellulation of a surface automatically enforces this property, but the chain complex need not have such a geometric origin.

Example: A cellulated surface with boundaries. Consider a piece of a planar 2D lattice, with boundaries. The n-cells are:

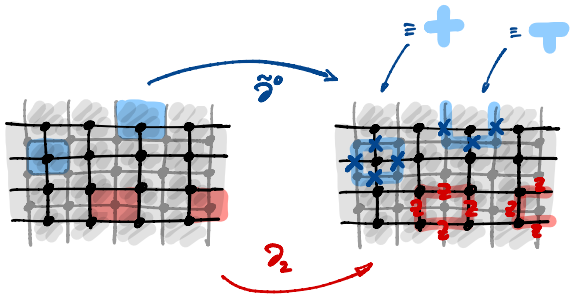


In the dual picture, the n-cocells are:



(Technically, such settings require us to speak of "relative homology," because we are considering subsets of n-cells and n-cocells from a larger cellulation without boundaries, and inheriting the relevant homological concepts.)

With these n-cells and n-cocells, and following through the homological CSS code construction, we obtain the stabilizers of the planar surface code with rough and smooth boundaries. The boundary stabilizers emerge through the boundary operators:



We have:

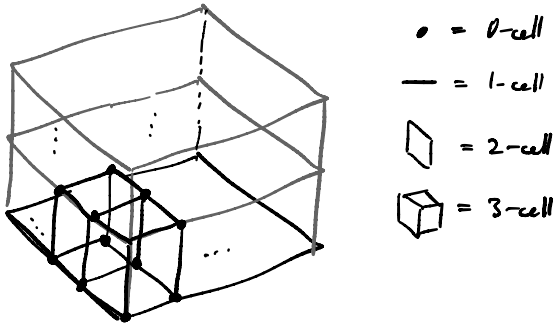
- $B_1 =$ strings that close up or terminate on a smooth boundary;
- $\tilde{B}^1 =$ costrings that close up or terminate on a rough boundary;
- $Z_1 =$ strings that wrap across horizontally;
- $\tilde{Z}^1 =$ costrings that wrap across vertically.

We find that

$$H_1 \simeq \mathbb{Z}_2 \ ; \qquad\qquad \tilde{H}^1 \simeq \mathbb{Z}_2 \ , \tag{6.15}$$

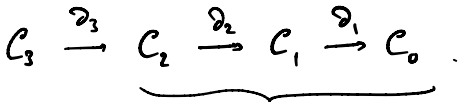
from which we immediately see that we encode 1 logical qubit.

Example: 3D toric code. Let's start from a cellulation of the 3D torus:



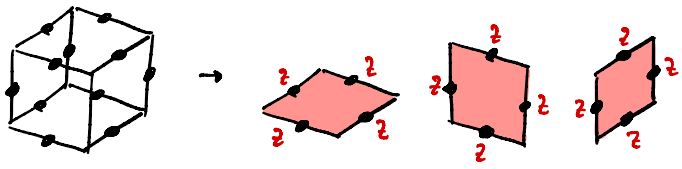
where opposing faces of the cube are identified to form the 3D torus.

We get a chain complex with one additional layer:

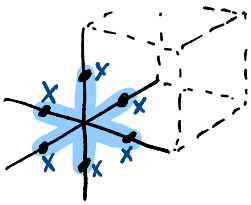


Let's ignore $\mathcal{C}_3$ and use the part of the chain complex $\mathcal{C}_2 \xrightarrow{\partial_2} \mathcal{C}_1 \xrightarrow{\partial_1} \mathcal{C}_0$ to build a CSS code according to the recipe derived above.

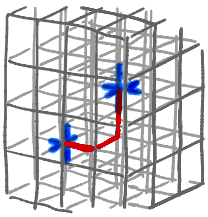
The Z-type stabilizers are associated with 2-D faces which come in three types:



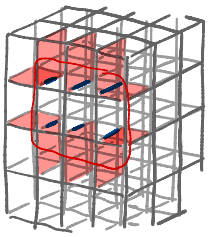
The X-type stabilizers correspond to 0-D vertices and act on the incident edges:



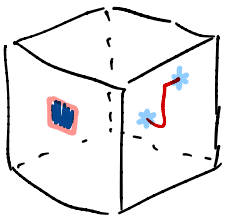
Star-type excitations are point-like, as for the 2-D toric code:



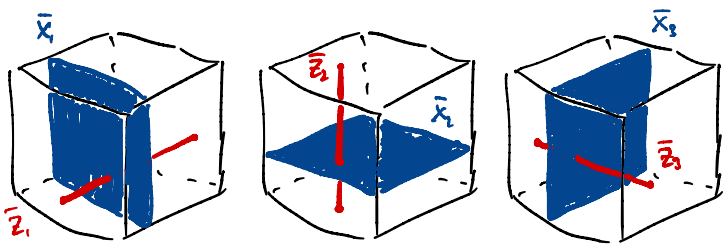
The plaquette-type excitations take the form of loops of plaquettes that surround membranes of errors:



We can represent these excitations pictorially as:



Logical Z operators are topologically nontrivial strings of Z's, while logical X operators are topologically nontrivial membranes. There are three logical qubits:



6.3 The color code

Another well-studied class of topological codes are *color codes*. They are closely related to surface codes but differ in their construction.

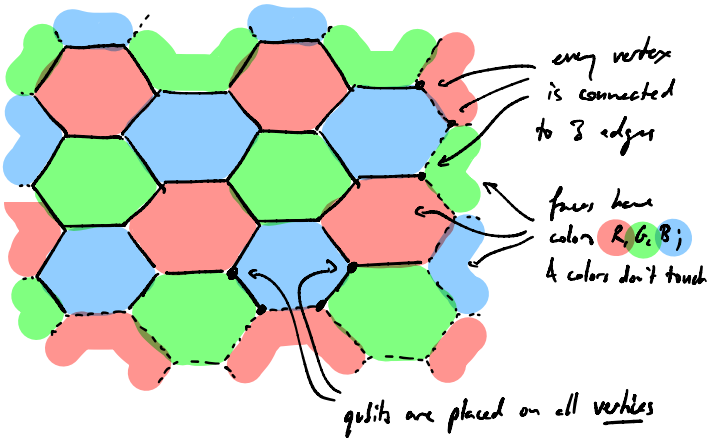
Consider a lattice consisting of vertices, edges, and faces such that:

(1) Each vertex touches exactly 3 edges (*the lattice is 3-valent*); and

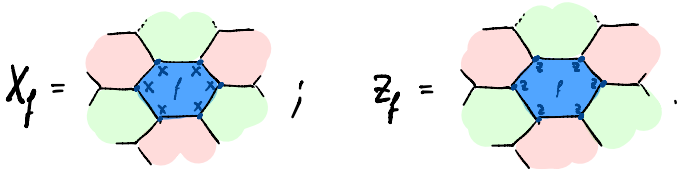
(2) Three labels, conventionally the colors red, green, and blue, can be attached to the faces such that no two neighboring faces have the same label (*the lattice is 3-colorable*).

We place a qubit on each *vertex* of the lattice.

For example, consider the hexagonal honeycomb lattice on the torus, i.e. with periodic boundary conditions:



To each face we associate an X-type stabilizer, acting with X's on each vertex connected to that face, as well as a Z-type stabilizer, acting with Z's on the same vertices. The X-type and Z-type stabilizer generators have the same supports, in contrast to the surface code.



The colors now designate the faces' coloring, rather than the type (X or Z) of the corresponding stabilizers. A darker-colored face is used to highlight a specific stabilizer generator.

The stabilizers are not all independent on the torus. Taking the product of all X-type (or Z-type) stabilizers associated with a given color applies a single X (or Z) to every qubit of the lattice. If R, G, and B designate the set of red, green, and blue faces, we find that

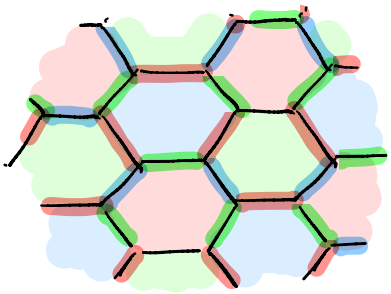
$$\prod_{f\in R} X_f = \prod_{f\in G} X_f = \prod_{f\in B} X_f \ ; \tag{6.16}$$

$$\prod_{f\in R} Z_f = \prod_{f\in G} Z_f = \prod_{f\in B} Z_f \ . \tag{6.17}$$

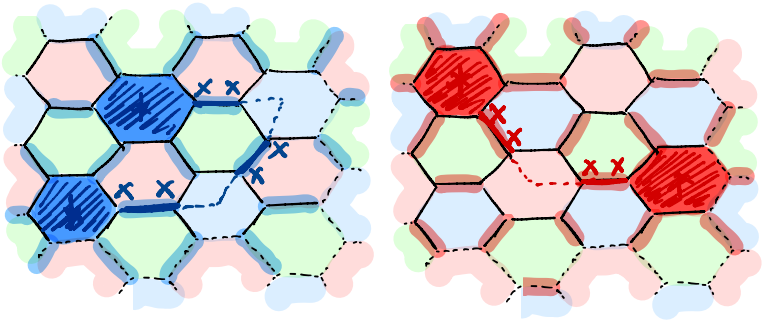
Exercise: Show that all X_f's and Z_f's commute, using the fact that the lattice is 3-valent and 3-colorable.

The color code is invariant if we swap the roles of X and Z. As a CSS code, $H_X = H_Z$. We can thus focus on one type of errors (say X's) and we know that the exact same analysis applies to errors of the other type (Z).

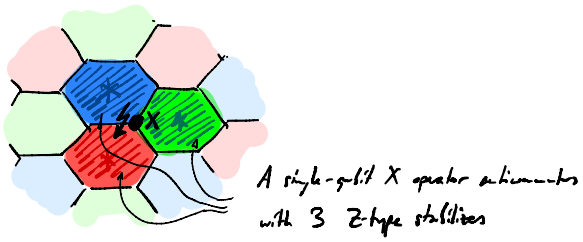
Let's assign a color to each edge, fixed to be distinct from the colors of the adjoining faces:



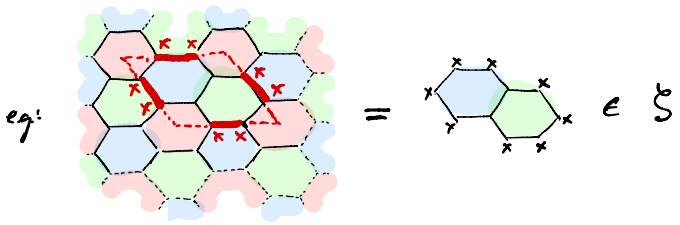
A string of X's hopping between faces of color c along c-colored edges give rise to a pair of c-color face excitations of Z_f stabilizers:



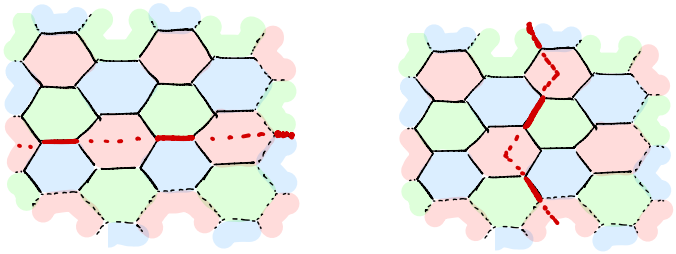
On the other hand, a single X error produces a triplet of excitations, each a Z_f stabilizer on the adjoining faces, and one of each color:



Closed loops of X errors along edges of any color commute with all stabilizers:

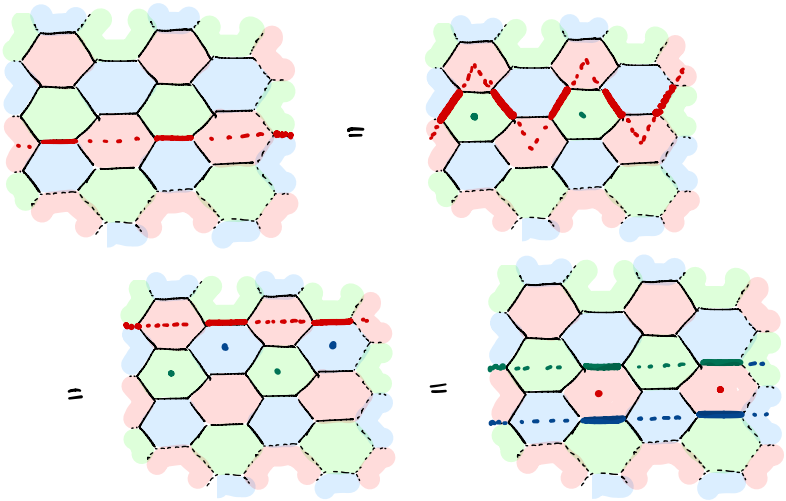


Logical operators are represented as strings of X's and Z's along colored edges that wrap around the torus:



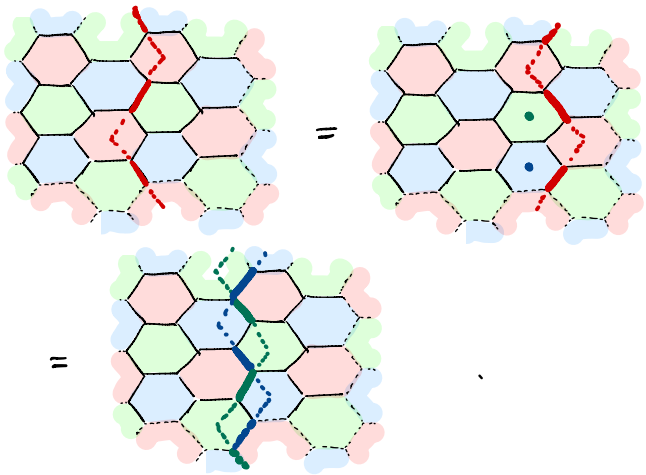
These strings cannot be contracted to a point. They are not stabilizer-equivalent to the identity, and they act nontrivially on the code space.

Strings that wrap around horizontally can be deformed by multiplying by plaquettes. Note there are only two independent string colors: A red string (for example) is equivalent to a product of a green string with a blue string:

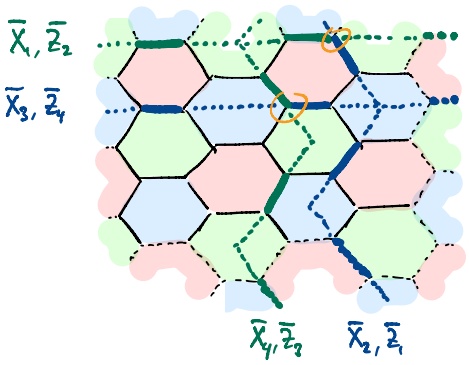


Faces that are marked by a thick dot ("$\bullet$") indicates multiplication by that face stabilizer operator.

Similarly for strings that wrap around vertically:

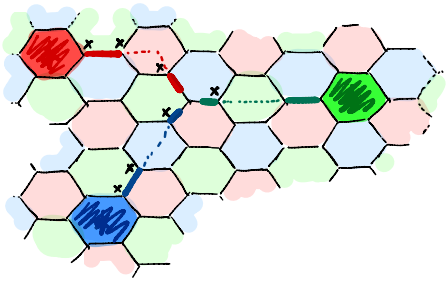


The 2D color code on the torus therefore encodes $k = 4$ *logical qubits*, with the following logical operators:

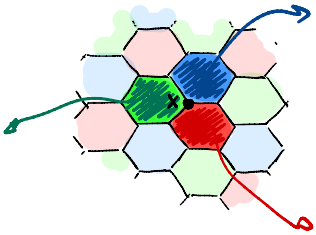


The $\overline{X}_i$ are chosen as strings of X operators on the strings designated above and the $\overline{Z}_i$ are chosen as strings of Z operators along the same string locations.

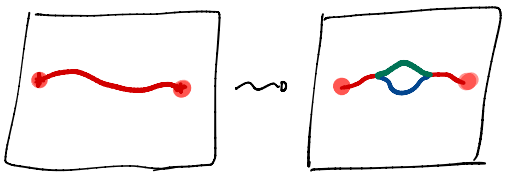
The behavior of strings in the 2D color code is richer than in the surface code. Strings can "branch out" while changing color:



We can imagine creating such a string merging point by starting with a single-qubit error, which creates an excitation triplet, and then moving the individual excitations around:



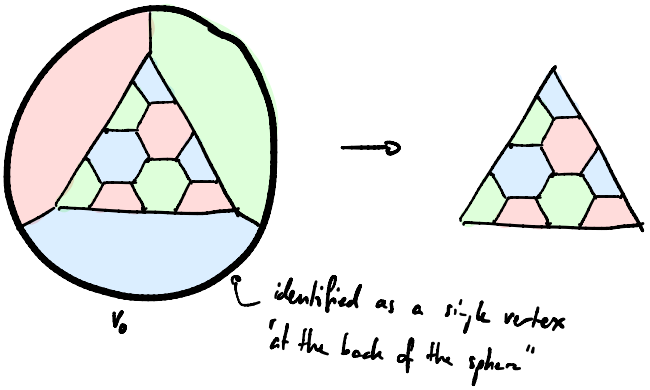
Strings can thus be deformed into more complicated objects called *string-nets*:



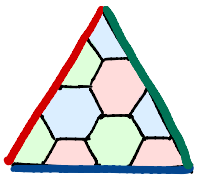
A decoder will want to account for such possibilities when associating an equivalence class of error operators to a set of observed syndromes.

The color code can be viewed as a physical model that exhibits an interesting topological phase with ties to exotic phases of matter studied in condensed matter physics.

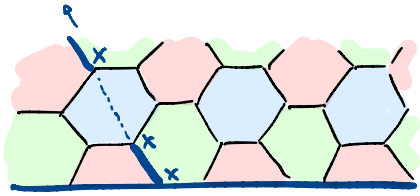
A trick to obtain a *planar* 2D color code is to tile the 2-sphere (embedded in three dimensions) with a 3-valent, 3-colorable lattice; we then remove one vertex along with its adjoining faces:



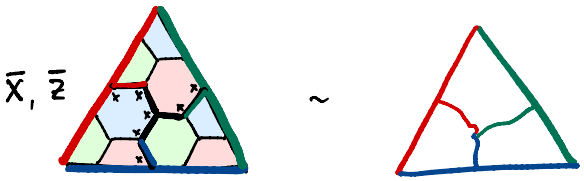
We associate to each boundary a *color*, which is that of the associated face that we erased:



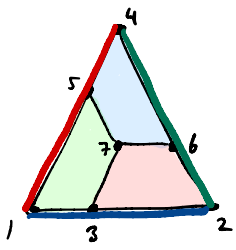
Strings can end on a boundary of the corresponding color:



Logical operators for the triangular planar color code we drew above can be chosen as follows:



In its simplest form, this is a code on seven qubits:



Writing out the stabilizer generators of the 7-qubit color code explicitly, we find:

$$X_1X_3X_5X_7\ , \quad X_2X_3X_6X_7\ , \quad X_4X_5X_6X_7\ , \tag{6.18}$$

$$Z_1Z_3Z_5Z_7\ , \quad Z_2Z_3Z_6Z_7\ , \quad Z_4Z_5Z_6Z_7\ , \tag{6.19}$$

and with logical operators

$$\overline{X} = X_3X_5X_6 \sim X_1X_2X_3X_4X_5X_6X_7\ ; \tag{6.20}$$

$$\overline{Z} = Z_3Z_5Z_6 \sim Z_1Z_2Z_3Z_4Z_5Z_6Z_7\ . \tag{6.21}$$

This is exactly the $[[7,1,3]]$ Steane code!

Color codes offer an alternative construction of topological codes with a perfect symmetry between X-type and Z-type stabilizers. As a CSS code, $H_X = H_Z$.

This symmetry between X-type and Z-type stabilizers will prove useful for fault tolerant quantum computation: it enables the color code to offer appealing schemes for performing more advanced *logical gates* (e.g. all Clifford gates). We'll return to this point in the following chapters on fault tolerance.

Color codes can be generalized to three dimensions (and beyond). In D dimensions, the lattice needs to be colored with $D+1$ different colors. The structure of higher-dimensional color codes is also somewhat richer, and the generalization of the color face operators might not commute. These codes form *subsystem codes*, which we might introduce later if we have the time.

A 2D color code can be viewed as two superimposed copies of the surface code. It also has a homological description and can be constructed using the homological procedure described above.

6.4 Further reading

Recommended reading includes great lecture notes on topological codes by Dan Browne [13].

For useful insights on the homological structure of CSS codes we point in particular to one of the original papers on the subject [44]. See also Nikolas Breuckmann's Ph.D. Thesis [9].

Valuable explanations of color code constructions are given in Refs. [10, 38, 45]. A detailed anyonic condensed matter picture applicable to color codes can be found in Refs. [46, 47].

Chapter 7

Fault Tolerance I: Code memory thresholds

So far, we've been concerned with constructing codes that have good properties in terms of their ability, in principle, to correct a high number of errors. Now, we'll revisit some of our initial assumptions with the focus on engineering schemes that can protect information in realistic hardware over extended periods of time and which can be used to perform a quantum computation.

Specifically, we'll be concerned with the following points:

- If each physical qubit suffers noise independently, then scaling up codes to many physical qubits also increases the total amount of noise the system suffers. Are we sure that using a larger code will offer better protection rather than being overwhelmed by the additional noise?
- So far, we've assumed that the recovery operation, which includes syndrome measurements, are ideal and do not suffer any noise. What happens when the recovery procedure is faulty?
- We want to run a quantum computation on the encoded qubits. How can we apply a universal set of gates on the logical qubits? What if the implementation of these gates also suffers from noise?

We'll study these questions sequentially. While we'll discuss some of these concepts in the context of general quantum error-correcting codes, we'll also put a focus on providing answers and illustrating these concepts for the surface code in particular.

7.1 Error-correcting code threshold

So far, we've quantified a code's ability to protect against errors by its *distance*. For the surface code, the distance grows as the lattice size L, meaning $d \sim L \sim \sqrt{n}$. If each qubit suffers a noise rate p, meaning that each individual qubit has a probability p of being corrupted by an error operator, then we can expect $\sim pn$ corrupted qubits after exposure to the noise. But we see that $pn \gg \sqrt{n}$ for large n, meaning that the expected number of errors can become much larger than the code's distance. Does this indicate that the surface code fails to protect information for large n?

The distance of a code is a worst-case measure that is often too pessimistic. The distance metric guarantees that strictly all errors below half the distance are, for sure, correctable.

In the surface code, a string of length $L \sim \sqrt{n}$ is a nontrivial logical operator, so the distance cannot exceed L. Yet most errors of weight $\sim \sqrt{n}$ would look like scattered errors on the surface, and would not lead the decoder to cause a logical error; they remain perfectly correctable.

Rather than the distance, we're interested in the probability that the recovery of the encoded information fails after exposure to the noise. This is the ***logical error rate*** or ***logical failure rate*** p_L.

In general, the precise definition of p_L should include any assumptions about the noise model, the decoder employed, etc. It also depends on the desired task, for instance, to simply store qubits in memory or to run a quantum computation.

Let us for now consider the task of simply storing the qubits while protecting them from noise. Then p_L can be defined as the overall probability of the decoder causing a logical error.

The code thus uses n qubits, each with failure rate p, to simulate k qubits with failure rate p_L.

If $p_L > p$, the error-correcting scheme makes the qubits worse! This situation can happen, for instance, if the physical noise rate p the qubits suffer is too high or if the error-correcting code and/or decoder are too naive.

However, if $p_L < p$, then the new qubits are better than the original physical qubits. This is the situation we'd like to engineer.

For a family of codes parametrized by their number of qubits (for instance, surface codes of various lattice sizes; alternatively, a simple code concatenated multiple times with itself), we say that the code family has a ***threshold*** if there is some noise rate p_{thres} (the *threshold*) such that for all $p < p_{\text{thres}}$, we have $p_L(p) \to 0$ as $n \to \infty$.

Often, when a code family has a threshold, the logical error rate $p_L(p)$ for $p < p_{\text{thres}}$ decays exponentially in some fixed power of n. This means that the overhead required for fault tolerance, in principle, is only $n \sim \text{poly}(\log(1/\epsilon))$ where ϵ is the desired logical error rate.

The threshold of a code family actually depends on many details and assumptions. These include:

- The specifics of the noise model;
- The chosen decoder and classical computation power;
- How syndromes are measured and how noise can induce faulty syndromes (which we have not considered yet here);
- The logical operations we want to do;
- etc.

Depending on the assumptions that went into the definition of p_L, the corresponding threshold may change significantly. Thresholds associated with some standard assumptions are sometimes given more specific names:

Assumption on the *task*: A threshold associated with the task of storing a logical qubit while protecting it from noise can be called a ***memory threshold***.

Assumptions on the noise model:

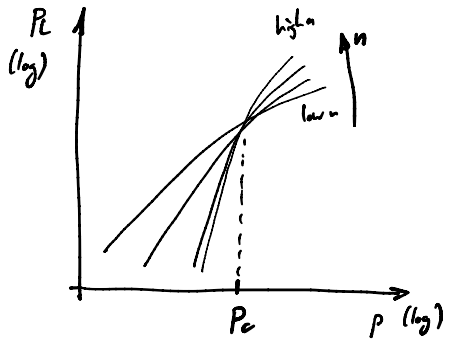
- ***code capacity threshold***: We assume that syndrome measurements are instantaneous and perfectly noiseless.

- ***phenomenological noise threshold***: We model errors in the syndrome measurements as the wrong outcome occurring with a fixed probability independently for each measured stabilizer generator; the syndrome readout is instantaneous.

- For the ***circuit-level noise threshold***, we write out explicit circuits for the measurement of stabilizers in terms of single-qubit gates, two-qubit gates, and the preparation and measurement of ancillary qubits. We assume that each step (preparation, gate, measurement) is subject to noise in a suitable noise model.

Instead of studying the asymptotic behavior in n, we can fix a code instance (we fix n), and we can ask if there is a $p_{\text{pseudothres}}$ such that $p_L(p) < p$ for all $p < p_{\text{pseudothres}}$. Such $p_{\text{pseudothres}}$ is called a ***pseudothreshold***. (It's not a *threshold* in the precise sense defined above, as it applies for a fixed n.)

When a code has a threshold, we can usually witness it by plotting the logical error rate p_L against the physical error rate p as we scale the code up:



If a code family has a threshold p_{thres}, then as long as the physical error rate p is below p_{thres}, we can in principle implement logical qubits with arbitrary small levels of noise, by picking a sufficiently large code.

7.2 The surface code has a threshold

Here, we'll prove that the surface code has a memory threshold using a simple argument (see [3]). Consider the code capacity threshold setting, with the following assumptions:

- Each qubit suffers an X error with probability p, and independently, a Z error also with probability p;
- Syndrome measurements happen instantaneously and are perfectly noiseless;
- Fast and accurate classical computation for decoding;
- We fix p_L as the probability of the event that the decoder picks a correction operation that induces a logical error. Let $F = F(\text{syndrome}(E))$ be the correction Pauli operator that is chosen by the decoder as determined by the observed syndrome, such that FE commutes with all stabilizers (all stabilizer outcomes are restored to $+1$). We define:

$$p_L = \sum_E \Pr(E) \Pr\Big[FE \in N(\mathcal{S}) \setminus \langle i, \mathcal{S}\rangle \,\Big|\, \text{syndrome}(E)\Big] , \tag{7.1}$$

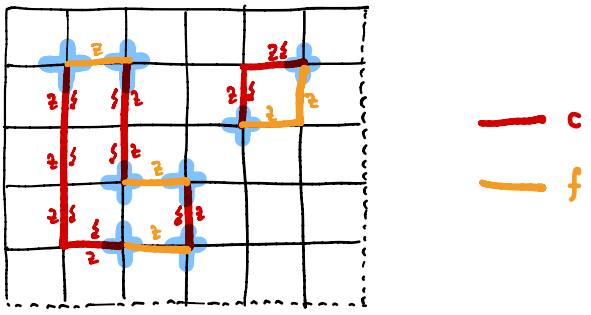
  where the sum ranges over all Pauli operators E with the fixed global phase $+1$; equivalently, it ranges over all $2n$-bitstrings in the binary symplectic representation of the Pauli operators.

Because the code is CSS, we can correct each error type separately, ignoring the other type. Let's focus first on *Z-type errors*, which are caught by the X-type star operators.

Suppose a Z-type error $E = Z^c$ occurs, where c is a 1-chain corresponding to the subset of edges on which the error applies a Z operator.

We measure all the stabilizers, leading to a syndrome vector syndrome(E). Based on this information, a decoder selects a Pauli recovery operation F to apply; if FE is a stabilizer (up to a phase), the error is successfully corrected. The correction F is a collection of Z operators that we can describe with another 1-chain f, $F = Z^f$.

Let us choose the decoder to be the minimum weight decoder: F is chosen as the correction operation $F = Z^f$ such that FE commutes with all stabilizers (restores all syndrome outcomes to $+1$) and such that f has minimal weight $|f|$. That FE commutes with all stabilizers implies that the chain $c + f$ has no boundary (or else it would leave a nontrivial syndrome outcome at that boundary), so it is a cycle: $c + f \in Z_1$. The decoder, therefore, determines the 1-chain f of smallest weight $|f|$ such that $c + f$ has no boundary:



We would like to upper bound the probability that $c + f$ is a homologically nontrivial cycle. This can only happen if $c + f$ has length at least L, the width of the surface code lattice. It turns out there is a simple counting argument that upper bounds the probability that $c + f$ contains any long closed loop, which must be the case if $c + f$ contains a homologically nontrivial loop.

Suppose $c+f$ contains a closed loop o of length ℓ. If more than half of the edges of o lie in f rather than in c, we could reduce the weight of f via $f \to f+o$; f would not be the correction operation with minimal weight, contradicting our earlier assumption. So at least half of the edges of o lie in c. Therefore, for any closed loop o of length ℓ, the chain $c+f$ can only contain o if there at least $\ell/2$ edges of o are in c.

Fix a loop o of length ℓ. We can upper bound the probability that $c+f$ contains o as follows. Consider all possible partitions of the edges of o into sets of edges s and $\bar{s} = s+o$ (i.e., $s+\bar{s}=o$ and $s, \bar{s} \subseteq o$); exactly one such $(s,\bar{s})$ will identify which edges of o belong to c and which edges belong to f. Furthermore, we just saw that we can only have $s \subseteq c$ and $\bar{s} \subseteq f$ if $|s| \geq \ell/2$, so

$$\Pr\big[o \subseteq c+f\big] = \sum_{\substack{s \subseteq o\,:\,|s| \geq \ell/2 \\ \bar{s}=s+o}} \Pr\big[s \subseteq c \text{ and } \bar{s} \subseteq f\big] \ . \tag{7.2}$$

We can upper bound each probability as $\Pr\big[s \subseteq c \text{ and } \bar{s} \subseteq f\big] \leq \Pr[s \subseteq c] \leq p^{|s|} \leq p^{\ell/2}$, since $s \subseteq c$ means that the random chain c has each edge of s affected by an error which happens with probability p independently on each edge, and since $|s| \geq \ell/2$. Because there are at most 2^ℓ subsets of o, we find

$$\Pr\big[o \subseteq c+f\big] \leq 2^\ell p^{\ell/2} \ . \tag{7.3}$$

(In this argument, the probability is taken over the error c; the correction f is a function of c, $f \equiv f(c)$; and o is completely fixed.)

The overall probability of failure $p_L^{(Z)}$ for Z-type errors can now be upper bounded as follows:

$$\begin{aligned} p_L^{(Z)} &\leq \sum_{o \text{ homologically nontrivial}} \Pr\big[o \subseteq c+f\big] \\ &\leq \sum_{o\,:\,|o| \geq L} \Pr\big[o \subseteq c+f\big] \\ &= \sum_{\ell=L}^{2L^2} M_\ell \cdot 2^\ell p^{\ell/2} \ , \end{aligned} \tag{7.4}$$

where M_ℓ is the number of loops of length ℓ and where no loop can have more than the $2L^2$ total number of edges on the lattice. To upper bound M_ℓ, we fix a starting point and a length ℓ; the loop can start in any of 4 directions and then pick any of 3 directions for all subsequent steps. There are L^2 possible starting points, so $M_\ell \leq L^2 \cdot 4\,3^{\ell-1}$. Therefore

$$p_L^{(Z)} \leq \sum_{\ell=L}^{2L^2} 4\,L^2\,3^{\ell-1}\,2^\ell\,p^{\ell/2} \leq \frac{4}{3} L^2 \sum_{\ell=L}^{2L^2} (36p)^{\ell/2} \ , \tag{7.5}$$

If we assume that $36p < 1$, or equivalently $p < 1/36$, then

$$p_L^{(Z)} \leq \frac{8}{3} L^4 (36p)^{L/2} \xrightarrow{L \to \infty} 0 \tag{7.6}$$

since the exponential decay in L dominates the poly(L) term.

We can apply exactly the same argument to correct X-type errors with the Z-type stabilizer syndromes, by reasoning on the dual lattice instead of the original lattice. Indeed, $p_L \leq p_L^{(Z)} + p_L^{(X)}$; if both $p_L^{(X)}$ and $p_L^{(Z)}$ decay to zero as $L \to \infty$, then p_L also decays to

zero.

This proves that the surface code has a threshold. At any physical error rate $p < 1/36 \approx 2.7\,\%$, the logical failure rate p_L decays to zero in the large system size limit.

It turns out that the value $1/36 \approx 2.7\,\%$ is a pessimistic estimate of the threshold value. In the following section, we'll use more powerful techniques to prove that the threshold of the surface code in this setting is higher, at $\approx 10\,\%$.

7.3 Surface code threshold from statistical mechanics

The above argument is useful to prove the *existence* of a code capacity memory threshold for the surface code, but it is a bit too crude to reliably estimate the precise threshold value. Here, we present a powerful argument that can determine the value of this threshold, based on a formal mapping of the calculation of the failure rate p_L to a problem of determining physical properties of a specific model from statistical physics.

We operate with the same assumptions as in Section 7.2 and use the same definition of the logical failure rate p_L, with uncorrelated X/Z errors and where we'll choose the decoder to be either a minimum-weight decoder or a maximum-likelihood decoder. Again, X and Z errors are decoded separately; the argument we present here applies to Z errors, but the same argument applied to the dual lattice picture yields the same threshold for correcting X errors.

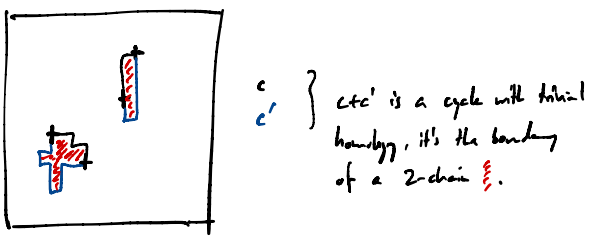
For a Z-type error, we can write $E \propto Z^c$, where c is a 1-chain describing the subset of the edges on whose qubits E acts with a Pauli-Z operator.

The probability of E occurring is

$$\Pr[c] = (1-p)^{n-|c|}p^{|c|} = (1-p)^n \left(\frac{p}{1-p}\right)^{|c|} . \tag{7.7}$$

A decoder must decide, based on the observed syndrome syndrome(E), which Pauli correction operation F to apply. The correction operation only needs to be determined up to a stabilizer, since stabilizers act trivially on the code space. Therefore, a decoder must decide whether it is better to apply a correction operation represented by a 1-chain c (or any homologically equivalent $c' \sim c$), or to apply instead a correction operation represented by a 1-chain of the form $c + z$ where z is a homologically nontrivial cycle.

For maximum likelihood decoding, we need to compute the probability of each class of stabilizer-equivalent errors and pick the class with highest overall probability. The probability of a given error class is computed by summing up the probability weights associated with all homologically equivalent chains c', i.e., all chains c' such that $c + c'$ is a closed loop with trivial homology. I.e., there is a 2-chain r such that $c + c' = \partial r$:



The probability of all errors c' that are homologically equivalent to c is then:

$$\Pr[\overline{c}] = \sum_{\substack{c': \\ \exists\, r\,:\, c+c'=\partial r}} \Pr[c'] \ . \tag{7.8}$$

In order to compute $\Pr[\overline{c}]$ and $\Pr[\overline{c+z}]$ for any given z, let's first express $\Pr[c']$ in terms of $\Pr[c]$ for any 1-chain c' that has the same syndrome as c. We express $c' = c + w$ for some 1-cycle w, noting that $w_\ell = 1$ whenever $c_\ell \neq c'_\ell$:

$$\begin{aligned}
\Pr[c'] &= \Pr[c] \prod_{\ell\,:\, c_\ell \neq c'_\ell} \left\{ \begin{array}{ll} \frac{p}{1-p} & \text{if } c_\ell = 0 \text{ and } c'_\ell = 1 \\ \frac{1-p}{p} & \text{if } c_\ell = 1 \text{ and } c'_\ell = 0 \end{array} \right\} \\
&= (1-p)^n \Big(\frac{p}{1-p}\Big)^{|c|} \prod_{\ell\,:\, w_\ell = 1} \Big(\frac{p}{1-p}\Big)^{(-1)^{c_\ell}} \\
&= (1-p)^n \exp\Big\{ \sum_{\ell \in c} \log\Big(\frac{p}{1-p}\Big) + \sum_\ell w_\ell \cdot (-1)^{c_\ell} \log\Big(\frac{p}{1-p}\Big) \Big\} \ .
\end{aligned} \tag{7.9}$$

In the last line, we expressed $|c| = \sum_{\ell \in c} 1$ where c is thought of as a subset of links ℓ (with $\ell \in c \ \Leftrightarrow \ c_\ell = 1$), and the coefficient w_ℓ in the second term ensures that the term is only nonzero when $w_\ell = 1$. We now define

$$J := \frac{1}{2} \log\Big(\frac{1-p}{p}\Big) \ ; \qquad\qquad u_\ell = 1 - 2w_\ell = \begin{cases} +1 & \text{if } w_\ell = 0 \\ -1 & \text{if } w_\ell = 1. \end{cases} \tag{7.10}$$

The symbol J will help us simplify the notation whenever terms of the form $\log(p/(1-p))$ appear. The variable u_ℓ simply denotes the sign ± 1 associated with the binary value w_ℓ, and will prove useful as we drift towards concepts in statistical mechanics. Let's continue our computation to find

$$\begin{aligned}
\Pr[c'] &= (1-p)^n \ \exp\Big\{ -J \Big[\sum_{\ell \in c} (2 - 2w_\ell) + \sum_{\ell \notin c} 2w_\ell \Big] \Big\} \\
&= (1-p)^n \ \exp\Big\{ -J \Big[\sum_{\ell \in c} (1 + u_\ell) + \sum_{\ell \notin c} (1 - u_\ell) \Big] \Big\} \\
&= (1-p)^n \, e^{-nJ} \ \exp\Big\{ J \sum_\ell (-1)^{c_\ell} u_\ell \Big\} \ .
\end{aligned} \tag{7.11}$$

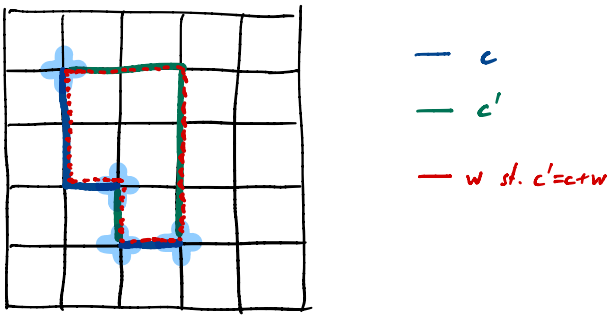
Setting $J_\ell := (-1)^{c_\ell} J$, we find

$$\Pr[c'] = \big[p(1-p)\big]^{n/2} \ \exp\Big\{ \sum_\ell J_\ell u_\ell \Big\} \ . \tag{7.12}$$

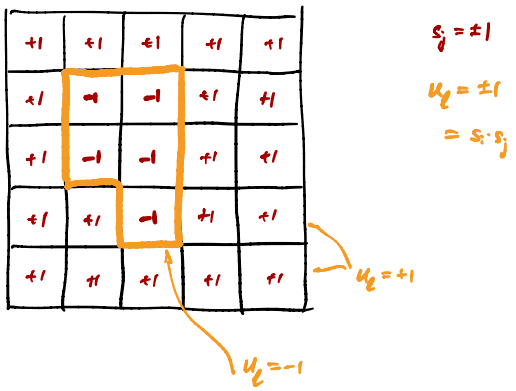
To compute $\Pr[\overline{c}]$, we need to sum up $\Pr[c']$ over all c' that differ from c by a 1-boundary cycle w:

$$\Pr[\overline{c}] = \sum_{w \in B_1} \Pr[c + w] \ . \tag{7.13}$$

The values $w_\ell \in \{0, 1\}$, or equivalently, $u_\ell \in \{+1, -1\}$, identify which edges we need to flip to map c to c':



Anticipating a connection with spin models in statistical mechanics, we place a spin $s_j \in \{+1, -1\}$ on each plaquette and set u_ℓ to be the product of the spins on the plaquettes adjoining the link ℓ, i.e., $u_\ell = s_i s_j$ where i, j are the faces that touch the link ℓ:



Of course, any spin configuration $\{s_j\}$ corresponds to a 2-chain r with $s_j = (-1)^{r_j}$, and u corresponds to the boundary of r via $u_\ell = (-1)^{w_\ell}$ and $w = \partial r$.

Now we've parametrized all $w \in B_1$ via spin configurations $\{s_j\}$. (Note a redundancy in the parametrization on the torus, as $s_j \to -s_j$ gives the same $\{u_\ell\}$.) Then

$$\Pr[\overline{c}] = C\,[p(1-p)]^{n/2} \sum_{\{s_j\}} \exp\Big\{ \sum_{\ell=\langle i\,j\rangle} J_\ell s_i s_j \Big\} \,, \tag{7.14}$$

where a constant C accounts for any redundancies in the parametrization (on the torus, $C = 1/2$), and where the notation $\ell = \langle i\,j\rangle$ refers to a link ℓ with adjoining faces i, j.

Let's write, recalling $J_\ell = (-1)^{c_\ell} J$:

$$\Pr[\overline{c}] = C\,[p(1-p)]^{n/2} \sum_{\{s_j\}} \exp\Big\{ -J\,H(\{s_j\}) \Big\} \,, \tag{7.15}$$

with

$$H(\{s_j\}) = -\sum_{\ell=\langle i\,j\rangle} \tau_\ell\, s_i s_j \;; \qquad\qquad \tau_\ell = (-1)^{c_\ell} \;. \tag{7.16}$$

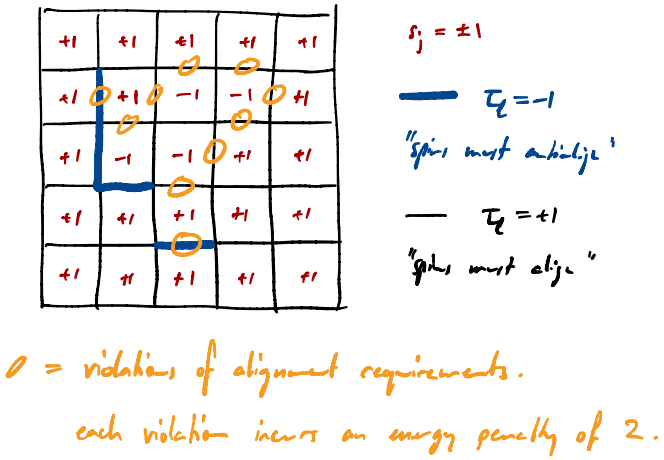
We recognize the partition function of a spin model with the Hamiltonian H, at inverse temperature $\beta = J$. The Hamiltonian H has Ising-type nearest-neighbor couplings τ_ℓ that vary on each individual link.

The connection to statistical mechanics is useful because this type of model has been extensively studied. Let's have a closer look at this model:

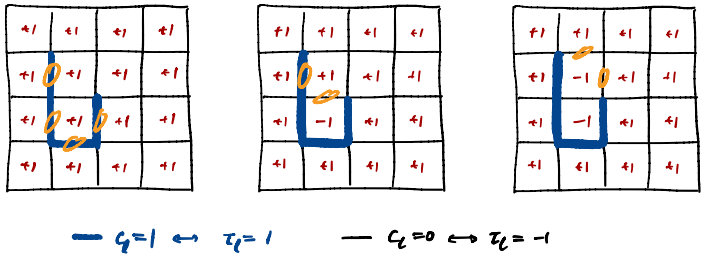
- Degrees of freedom: $\{s_j\}$ with $s_j \in \{+1, -1\}$;
- Couplings: $\tau_\ell \in \{+1, -1\}$. The couplings are treated as fixed parameters of the model for now.
- Hamiltonian: $H_\tau(\{s_j\}) = -\sum_{\ell=\langle i\, j\rangle} \tau_\ell\, s_i s_j$.

We'll occasionally write $\tau \equiv \tau(c)$ if $\tau_\ell = (-1)^{c_\ell}$ is obtained from the 1-chain c.

Spin configurations with low energy can be thought of as follows. We ask of the spins to align across every link for which $\tau_\ell > 0$, but to anti-align at every link where $\tau_\ell < 0$. Any violation of this assignment can be thought of as incurring an energy penalty for each violation. For example:



Consider the spin configuration that sets $s_j = +1$ on all sites j. It violates a number of terms determined as the size of the set $\{\tau_\ell < 0\}$ (below left). It might not be the energetically optimal configuration. The energetically optimal configurations are generally nontrivial, and there might be multiple optimal configurations (below center and right).



The physics of the model actually *only depends on the boundary of* c, which can be seen as follows. If $c' = c + w$ with $w = \partial r$ for some 2-chain r, then $\tau_\ell = (-1)^{c_\ell} \to \tau'_\ell = (-1)^{w_\ell}\, \tau_\ell$. Consider now the change of variables

$$s_j \to s'_j = \begin{cases} s_j & \text{if } j \notin r \\ -s_j & \text{if } j \in r\ . \end{cases} \tag{7.17}$$

Then we have that the term $\tau_\ell\, s_i s_j \to \tau'_\ell\, s'_i s'_j$ is invariant; indeed, flipped links $\ell \in w$ are precisely links across which exactly one of the two adjoining spins were flipped by the change of variables. Therefore, $H_\tau(s) \to H_{\tau'}(s') = H_\tau(s)$.

The physical properties of a statistical mechanical model at fixed inverse temperature β are encoded in its *partition function*:

$$Z(\beta, \tau) = \sum_{\{s_j\}} \exp\Big\{-\beta H_\tau(\{s_j\})\Big\} \ . \tag{7.18}$$

It is often more convenient to work with the *free energy*

$$F(\beta, \tau) = -\frac{1}{\beta} \log(Z(\beta, \tau)) \ . \tag{7.19}$$

Therefore: The probability of an error class $\Pr[\bar{c}]$ is determined by the partition function of the associated spin model with the couplings $\tau(c)$ and with $\beta \equiv J = -(1/2)\log(p/(1-p))$.

The logical error rate p_L involves an averaging over all possible errors, weighted by their probability of occurrence. An X-type error $E \propto X^c$ occurs with probability $\Pr[c] = (1-p)^{n-|c|}p^{|c|}$. In the spin model, E is mapped to the configuration of couplings $\tau(c)$.

The couplings can therefore be thought of as being *chosen at random*. Once they are sampled, they are fixed and not subject to thermal fluctuations. This is called *quenched disorder*.

In our spin model, the parameters τ_ℓ of the model are chosen at random in the form of quenched disorder; they individually take the value $+1$ with probability $1-p$ and the value -1 with probability p. This model is the ***random-bond Ising model***, which has been extensively studied.

The random-bond Ising model has two parameters, the bond probability p and the temperature $T = \beta^{-1}$.

At $p = 0$, all couplings are ferromagnetic (spins should align to achieve energetically favorable configurations) and there is no disorder; we recover the usual Ising model. The latter has a phase transition at a critical temperature T_{crit} between a phase where all spins are mostly aligned (*ferromagnetic phase* at $T < T_{\text{crit}}$) and a phase where thermal fluctuations dominate (the *paramagnetic phase* at $T > T_{\text{crit}}$).

At $T = 0$, on the other hand, we can break the ordered phase by increasing p instead of T. There is a critical $p_{\text{crit},0}$ up until which the system remains ordered, but beyond which the bonds start becoming too "incompatible with one another" and order breaks down. In the latter case, bonds are said to be *frustrated* and we find a *spin-glass phase*.

The random-bond Ising model's phase diagram is presented in Fig. 7.1. The phase diagram is divided into two regions, an *ordered phase* for small p and small T, where spins are mostly aligned, and a *disordered phase*, where there are no significant long-range correlations between the spins. A phase transition (solid black line above) separates the two regions.

It turns out the system exhibits some enhanced symmetry properties along the line $e^{-2\beta} = p/(1-p)$, called the ***Nishimori line***. Along this line, the thermal fluctuation probability associated with flipping a bond coincides with the relative probability of the bond flipping sign across different disorder samples.

In our mapping of the decoding problem to the random-bond Ising model, we had $\beta \equiv J = -(1/2)\log(p/(1-p))$, which is exactly the condition for the Nishimori line! Along the

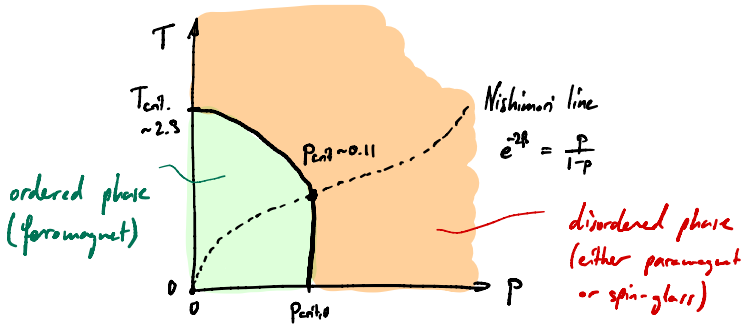


Figure 7.1: Phase diagram of the random-bond Ising model.

Nishimori line as we increase p, there is a phase transition at a critical value p_{crit} between the ordered phase and the disordered phase.

To diagnose which phase the system is in, we can inspect the *domain wall free energy*, defined as the difference in free energy caused by flipping all bonds across a homologically nontrivial cycle z:

$$\Delta_{\overline{z}}(\beta, \tau(c)) = F(\beta, \tau(c)) - F(\beta, \tau(c+z)) \ . \tag{7.20}$$

Let's average this quantity over the disorder:

$$\Big[\Delta_{\overline{z}}(\beta, \tau(c))\Big]_p \equiv \sum_c \Pr[c]\, \Delta_{\overline{z}}(\beta, \tau(c)) \ . \tag{7.21}$$

In the ordered phase (low T and low p), it turns out that $\big[\Delta_{\overline{z}}(\beta, \tau(c))\big]_p$ diverges with the system size. I.e., introducing a domain wall requires an extensive amount of energy.

In the disordered phase, $\big[\Delta_{\overline{z}}(\beta, \tau(c))\big]_p$ saturates to some finite value, so the domain wall free energy per site vanishes.

Whether the spin model's phase is ordered or disordered maps directly onto whether the logical error rate p_L goes to zero or remains finite in the large-system limit:

- If $\big[\Delta_{\overline{z}}(\beta, \tau(c))\big]_p \gg 0$ for all nontrivial loops z, then $\Pr[\overline{c+z}] \sim Z(\beta, \tau(c+z)) = \exp\{-\beta \Delta_{\overline{z}}(\beta, \tau(c))\}\, Z(\beta, \tau(c)) \ll \Pr[\overline{c}]$ so $\Pr[\overline{c}]$ dominates the other error class probabilities $\Pr[\overline{c+z}]$'s for homologically nontrivial 1-cycles z. The error class will be successfully identified by the maximum-likelihood decoder.
- If $(1/L)\, \big[\Delta_{\overline{z}}(\beta, \tau(c))\big]_p \to 0$, then $\Pr[\overline{c+z}] \sim \Pr[\overline{c}]$ for homologically nontrivial cycles z and the decoder has a significant probability of choosing the incorrect option $c+z$ rather than c.

The phase transition between the ordered phase and the disordered phase in the random-bond Ising model therefore directly translates into the regimes in which the surface code can be decoded to arbitrary accuracy in the large-system size limit, and where decoding is likely to fail. We therefore find a threshold for the surface code (subject to our assumptions listed earlier) that coincides with the ordered-disordered phase transition of the random-bond Ising model on the Nishimori line, which is at

$$p_{\text{thres}} = p_{\text{crit}} \approx 11\,\% \ . \tag{7.22}$$

A summary of the mapping between surface code decoding and the random-bond Ising model appears in Fig. 7.2.

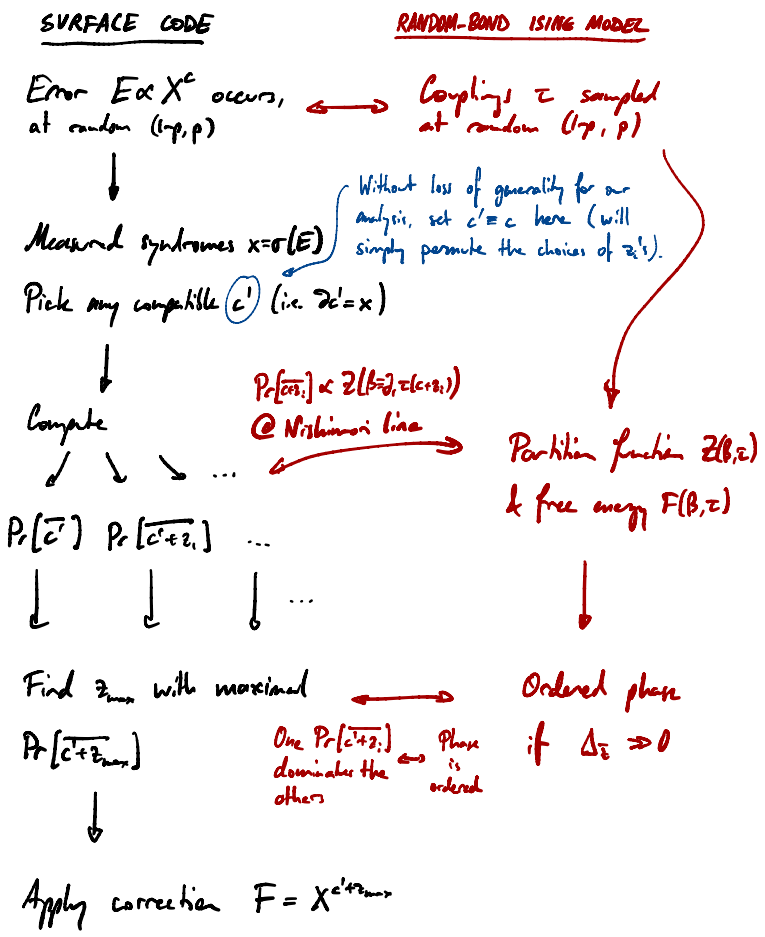


Figure 7.2: Summary of mapping the problem of decoding the surface code and the phase transitions of the random-bond Ising model.

If the error rate of the physical qubits is below the threshold value $p_{\text{crit}} \approx 11\,\%$ (for both X and Z type errors), then the surface code can be used to encode a qubit with arbitrarily low logical error rates in the setting where all the optimistic assumptions we made at the beginning of the chapter hold (perfect measurements, uncorrelated noise, maximum-likelihood decoding, etc.).

What if we used a minimum weight decoder instead of a maximum likelihood decoder? Recall that the latter can be implemented efficiently using the minimum-weight perfect matching algorithm, unlike maximum-likelihood decoding which is generically hard.

The minimal-weight error c' that is homologically equivalent to c is, in fact, the minimum energy configuration of the associated spin model. Indeed, it's the configuration that violates the fewest bonds. The relevant threshold therefore coincides with $p_{\text{crit},0}$ along the $T = 0$ line in the phase diagram of the random-bond Ising model (Fig. 7.1), recalling that the $T = 0$ state is the ground state of the model when the latter is unique. The threshold $p_{\text{crit},0}$ is believed to take a value around

$$p_{\text{crit},0} \approx 10\,\% \ . \tag{7.23}$$

7.4 Surface code threshold with measurement errors

In this section, we'll try to lift some assumptions that we made in the previous section and which were overly optimistic for realistic settings.

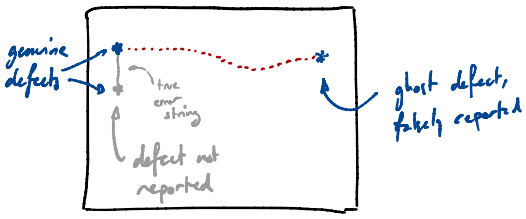
Syndrome measurements of the surface code involve measurement weight-4 Pauli operators. These typically require the application of several 2-qubit gates and involve an ancillary qubit. Certainly the resulting measurements can be faulty, if an error creeps in at any point of this procedure.

Let us first simply assume that syndrome measurements have noisy outcomes: The stabilizer is measured noiselessly and instantaneously, but the outcome is flipped with probability q.

Then, each syndrome outcome falls into one of four cases:

- a genuine defect, correctly detected (outcome -1);
- a genuine defect, not detected (outcome $+1$);
- no genuine defect, correctly reported as no defect (outcome $+1$);
- no genuine defect, but defect falsely reported ("ghost defect"; outcome -1).

A decoder that does not account for measurement errors could easily be misled into causing a logical error. Consider, for example, the following defect pattern:

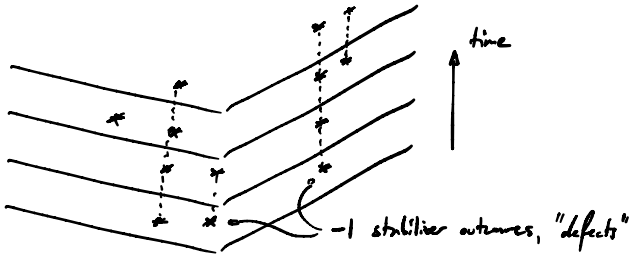


In this example, a single short error string leads to two genuine defects at its endpoints. One genuine defect is reported correctly whereas the other one fails to be detected due to a measurement error. Additionally, a ghost defect appears at a distant location because of a measurement error. Seeing two defects, a decoder might match them up, introducing a large error string (red dotted line) which is likely to lead to a logical error in future rounds of error correction.

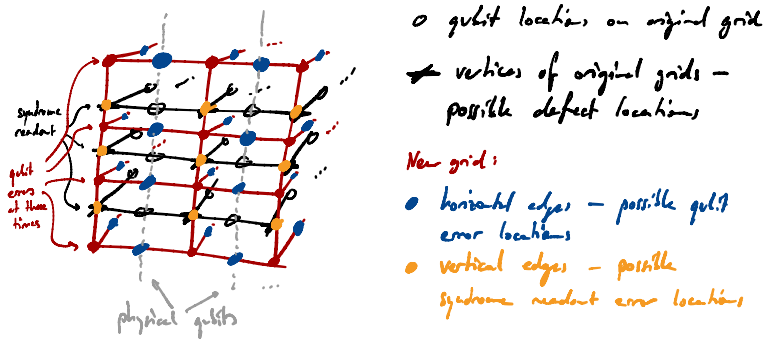
Again, we focus only on Z-type errors causing X-type defects (star operators). The analysis for X-type errors follows via a similar argument on the dual lattice.

A simple way to enhance the reliability of our picture of the errors is to measure the stabilizers multiple times. Then we'll notice if measurement outcomes are consistent over time, and we'll be able to identify the faulty measurement outcomes.

We can represent the outcomes in three directions, the third dimension representing time:

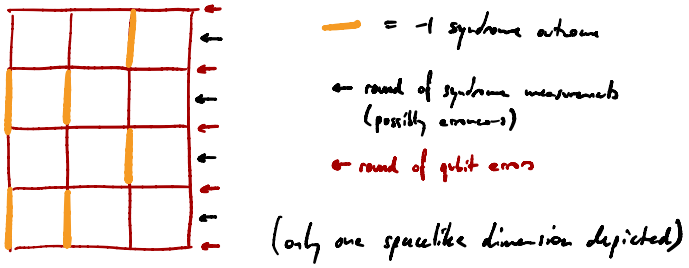


We consider a 3-dimensional square lattice that is interleaved between the timelike copies of the code, such that syndrome outcomes are associated with vertical edges. The copies of the surface codes are the black grids, and the new 3D grid is drawn in red:

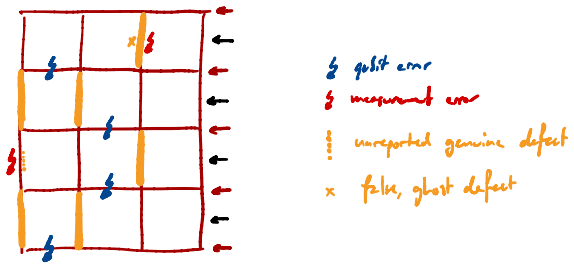


We suppose that syndrome readouts (where measurement errors can happen) are interleaved with times at which qubit errors can occur. The qubit errors happen at the horizontal edges of the new 3D lattice. Measurement errors happen during the syndrome readout at the vertices of the original horizontal grid copies, which coincides with vertical edges of the 3D lattice.

The syndrome measurement outcomes are placed on the vertical edges of the 3D lattice, providing a "history" of the possibly different outcomes in time. It is a 1-chain on the 3D lattice that contains only vertical edges:



The above pattern may have been caused, e.g., by the following configuration of qubit and measurement errors:



Observation: The chain of observed syndrome outcomes and the actual error chain (with qubit and measurement errors) must have the same boundary!

Proof. Let x be the 1-chain of observed syndrome outcomes and E be the 1-chain of both qubit or measurement errors. The 1-chain $x + c$ may contain both horizontal and vertical edges. The vertical links of $x + c$ correspond to genuine defects propagating in time (either

a correctly detected defect, or an incorrectly missing defect). The horizontal links of $x + c$ are either the creation or annihilation of a defect pair, or the propagation of a defect from one location to a neighboring location. Therefore, $x + c$ are closed loops associated with defect world lines. ∎

The measured syndrome outcomes therefore reveal the boundary of the error chain. This was already the case in 2D with perfect measurements; with measurement errors we find a very similar picture in 3D!

For a decoder to accurately diagnose the error chain (up to homological equivalence), it must determine which homology class of errors $\overline{c}$ is likeliest, with $c = x + w$ for a 1-cycle w. Here,

$$\Pr[\overline{c}] = \sum_{w \in B_1} \Pr[c + w] \ . \tag{7.24}$$

Again, $\Pr[\overline{c}]$ is determined by the partition function of a spin model now associated with a three-dimensional version of the random-bond Ising model along its Nishimori line.

Minimum-weight error decoding again corresponds to the $T = 0$ state of the associated spin model.

We usually suppose for simplicity that the measurement error probability equals the qubit error probability, $q = p$, to ensure the spin model is isotropic and to simplify the analysis.

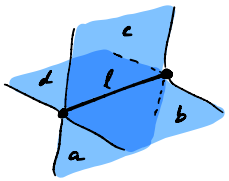
In this case, numerical evidence points to an error probability threshold of

$$p_{\mathrm{crit},0}^{(3\mathrm{D})} \approx 3\,\% \ . \tag{7.25}$$

The relevant 3D spin model is also called the *random-plaquette $\mathbb{Z}_2$-gauge model*.

In the random-plaquette $\mathcal{Z}_2$-gauge model, we can think of spins $s_j \in \{\pm 1\}$ as residing on the plaquettes of the primal lattice. The u_ℓ reside on the primal links and are determined as the boundary of the spin configuration viewed as a 2-chain.

More explicitly, we can write $u_\ell = s_a s_b s_c s_d$ with $\ell = \langle a\, b\, c\, d \rangle$ denoting a link ℓ adjoined by four plaquettes a, b, c, d:



$u_\ell = s_a \cdot s_b \cdot s_c \cdot s_d$.

In this model, introducing "antiferromagnetic links" ($\tau_\ell = -1$) means energetically favoring configurations which have an odd number of $s_a = -1$ around the links ℓ with $\tau_\ell = -1$.

On the dual lattice, we have $u_\ell \equiv u_{\tilde{p}}$ and $s_a = s_{\tilde{\ell}}$ for a dual plaquette $\tilde{p}$ and dual links $\tilde{\ell}$. Then

$$u_{\tilde{p}} = \prod_{\tilde{\ell} \in \partial \tilde{p}} s_{\tilde{\ell}} \ . \tag{7.26}$$

We can think of excitaitons as "magnetic flux tubes" that "travel" along the dual links. The $\mathbb{Z}_2$-valued "magnetic flux" through a dual plaquette $\tilde{p}$ is $u_{\tilde{p}}$.

An antiferromagnetic link ($\tau_\ell = -1$) is interpreted as a dual plaquette $\tilde{p}$ where a nontrivial flux $u_{\tilde{p}} = -1$ is energetically favorable ($\tau_{\tilde{p}} = -1$).

If a dual cube $\tilde{C}$ has an odd number of dual plaquettes $\tilde{p} \in \tilde{\partial}\tilde{C}$ with $\tau_{\tilde{p}} = -1$, then it is energetically favorable to have "magnetic flux tubes" terminate in the dual cube $\tilde{C}$. The dual cube $\tilde{C}$ can be thought of as a $\mathbb{Z}_2$ "magnetic monopole."

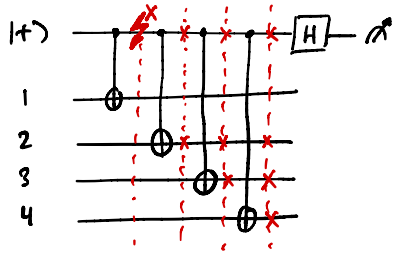
7.5 Surface code threshold with realistic error models

General results about code thresholds going beyond the assumptions we've made are difficult to compute without directly simulating the system we consider. In particular, details about measurement schedules, a carefully calibrated noise model, and a fine-tuned decoder go beyond what we can show with spin-model mappings.

For precise threshold estimates with numerical simulations, one often considers circuit-level noise, where all measurements and other operations are decomposed into hardware gates and noise is applied at each time step according to a carefully-chosen noise model.

Writing out explicitly how syndromes are measured also gives us an additional guide for improving the fault tolerance of error-correcting operations such as syndrome readout: **We must avoid, as much as possible, existing errors from spreading out into higher-weight errors.**

For instance, consider the following circuit that can read out $X_1X_2X_3X_4$ (e.g., a star operator of the surface code) using 2-qubit gates and an ancillary qubit:



Suppose the ancillary qubit suffers a single X error early during the readout ("↯" above). The remaining CNOT gates blow this error up into a weight-3 error on the data qubits!

While there exist clever schemes that enable readout of stabilizers without increasing the weight of errors, they typically require significant overheads (e.g., in the number of qubits). (We could use, for instance, the so-called *Shor* or *Steane* syndrome readout schemes, see [1, Chap. 12].) In the surface code, however, such schemes turn out not to be necessary. Instead, the order of the CNOTs in the circuit above should be optimized so multiple syndrome readouts can happen in parallel, all while minimizing the spreading of errors.

Estimates for the threshold of the surface code with circuit-level noise are around $p_{\text{thres}} \approx 0.01$.

7.6 Further reading

The mapping between decoding the surface code and the ordered phases of the random-bond Ising model was introduced by Dennis *et al.* [37]; this paper remains a primary go-to

reference to understand this mapping. The mapping is also well explained in Bombín's book chapter [38] and Fujii's review paper [5].

The argument on the existence of a threshold for the surface code is presented in Preskill's renowned lecture notes [3]; see specifically this file.

The statistical model mapping can be extended to study thresholds for the color code, see e.g. refs. [10, 48, 49] and references therein. Ref. [48] also contains a clear explanation of the different types of code thresholds.

Codes with code capacity thresholds are collected in a list in the error correction zoo.[1]

Recent developments have furthermore emphasized the necessity of viewing the quantum error correction protocol as happening over time, leading to a space-time view of quantum error correction. This picture is illustrated by the 3D spin model on which we mapped the decoding of the surface code with measurement errors. Modern views of decoding include detector error models [50].

Exciting recent demonstration of quantum error correction near or below threshold were presented in superconducting qubits [51], trapped ions [52] and in neutral atoms [53].

[1] https://errorcorrectionzoo.org/list/quantum_code_cap_threshold

Chapter 8

Fault Tolerance II: Computation on encoded states

Let's now turn to one of our core objectives of this course: How can we not only reliably store qubits in memory, but also perform some computation with them?

We'll begin by making some general statements about quantum error-correcting codes and qubit stabilizer codes, and we'll then assemble the different parts return to fault-tolerant computation with the surface code.

8.1 "Baby" fault-tolerant quantum computation: transversal gates

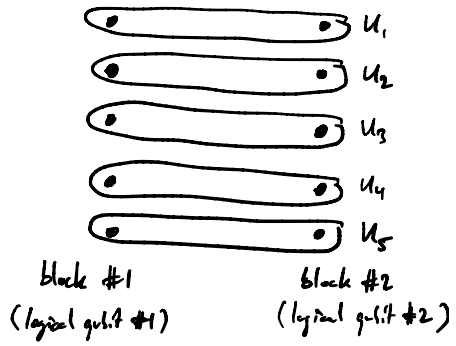
As we'll see, some logical gates are often very easy to perform on encoded states; however, implementing a universal gate set often requires significant overheads to circumvent some powerful no-go theorems.

Consider the [[7, 1, 3]] Steane code, which has the same X-type and Z-type stabilizers along with the logical operators $\overline{X} = X^{\otimes 7}$, $\overline{Z} = Z^{\otimes 7}$. If we apply $H^{\otimes 7}$, the Hadamard gate on all qubits, we swap the role of X and Z on each qubit. We remain in the code space, while $\overline{X}$ and $\overline{Z}$ are swapped! This is a *logical Hadamard operation.*

Suppose the physical Hilbert space is a tensor product of n subsystems, $\mathscr{H}_P = \mathscr{H}_{P_1} \otimes \cdots \otimes \mathscr{H}_{P_n}$, and consider an isometric encoding such that the erasure of any of the $\mathscr{H}_{P_i}$ is correctable. A ***transversal*** operation is an operation that is implemented as a single layer of independent physical operations $\mathcal{F} = \mathcal{F}_1 \otimes \cdots \otimes \mathcal{F}_n$, where each $\mathcal{F}_i$ acts only on $\mathscr{H}_{P_i}$.

While transversality depends in principle on the chosen tensor product decomposition, there is usually a standard or obvious decomposition that is implied. For instance, if the physical space consists of n qubits, transversality refers to operations acting individually on each qubit.

Suppose we have multiple instances of a $[[n, k, d]]$ qubit code, each instance encoding k logical qubits into a set of n physical qubits. (These instances are called *code blocks.*) For the definition of transversality, we usually take $\mathscr{H}_{P_i}$ as the collection of all the i-th qubits of each block. A transversal gate between multiple code blocks enables logical gates to be performed that connect the different sets of k logical qubits. A transversal gate generally looks like this:



Transversal implementations of logical gates are naturally fault-tolerant: They do not spread errors across the different subsystems. Furthermore, products of transversal gates are again transversal.

We're usually happy when we can implement a logical gate transversally.

For stabilizer codes, we've seen that logical Paulis can be implemented transversally. In particular, $\overline{Y} = i\overline{X}\overline{Z}$.

To better understand which logical gates can be implemented transversally on stabilizer codes, recall that a *Clifford unitary* U is one that conjugates Paulis to Paulis, i.e., $UPU^\dagger \in \mathsf{P}_n$ for all $P \in \mathsf{P}_n$.

Lemma 14. *Consider a stabilizer group $\mathcal{S} = \langle S_1, \ldots, S_r \rangle$. A Clifford unitary U is a logical operator if and only if it conjugates the stabilizer group $\mathcal{S}$ onto itself, i.e., $US_jU^\dagger \in \mathcal{S}$ for each S_j.*

Proof. As U is a Clifford unitary, it conjugates the $\{S_j\}$ to new Pauli operators $\{S'_j\}$, with $S'_j = US_jU^\dagger \in \mathsf{P}_n$. If U is a logical operator, i.e., $[U, \Pi] = 0$, then for any $S_i \in \mathcal{S}$ we have $S'_j\Pi = US_jU^\dagger\Pi = US_j\Pi U^\dagger = U\Pi U^\dagger = \Pi$ (recall $\Pi S_j = \Pi$). Thus $S'_j|\psi\rangle = |\psi\rangle$ for all $|\psi\rangle \in \mathcal{C}$ so S'_j must be in the code's stabilizer, $S'_j \in \mathcal{S}$.

Conversely, if $S'_j = US_jU^\dagger \in \mathcal{S}$ for all j, then for all $|\psi\rangle \in \mathcal{C}$ and for all j, we have $U|\psi\rangle = US_j|\psi\rangle = S'_jU|\psi\rangle$ so $U|\psi\rangle \in \mathcal{C}$. Therefore, U fixes the subspace Π and $[U, \Pi] = 0$. ∎

Example: The [[4, 2, 2]] code has the stabilizer group $\mathcal{S} = \{IIII, XXXX, YYYY, ZZZZ\}$ and logical Pauli operators $\overline{X}^{(1)} = XIXI$, $\overline{Z}^{(1)} = ZZII$, $\overline{X}^{(2)} = XXII$, $\overline{Z}^{(2)} = ZIZI$. The Pauli logical operators are clearly transversal Clifford gates. The Clifford gate $H^{\otimes 4}$ is also a logical operator: it conjugates $X^{\otimes 4}$ to $Z^{\otimes 4}$ and *vice versa*, leaving $\mathbb{1}^{\otimes 4}$ and $Y^{\otimes 4}$ invariant (note $HYH = -Y$ but signs cancel out over each pair of copies). The gate $H^{\otimes 4}$ swaps $\overline{X}^{(1)}$ with $\overline{Z}^{(2)}$ and $\overline{X}^{(2)}$ with $\overline{Z}^{(1)}$, so it performs logical Hadamards on the 2 qubits along with a logical SWAP.

Lemma 15. *For any CSS code, transversal CNOT's between two blocks of the code is a logical operator.*

Proof. Write the single-block code stabilizer group $\mathcal{S} = \langle \{S_j^Z\}, \{S_j^X\} \rangle$, where $\{S_j^Z\}$ are the Z-type stabilizer generators and $\{S_j^X\}$ are the X-type ones. The joint stabilizer group for the two code blocks is $\mathcal{S}_{\text{tot}} = \langle \{S \otimes I, I \otimes S\}_{S \in \mathcal{S}} \} \rangle$.

Recall CNOT acts by conjugation as $XI \to XX$, $ZI \to ZI$, $IX \to IX$, $IZ \to ZZ$. Then $\text{CNOT}^{\otimes n}$ maps $S_j^X \otimes I$ to $S_j^X \otimes S_j^X \in \mathcal{S}_{\text{tot}}$; $S_j^Z \otimes I$ to itself; $I \otimes S_j^X$ to itself; and $I \otimes S_j^Z$ to $S_j^Z \otimes S_j^Z \in \mathcal{S}_{\text{tot}}$. Since $\text{CNOT}^{\otimes n}$ maps stabilizer generators to elements of $\mathcal{S}_{\text{tot}}$, it is a logical operator. ∎

In a CSS code, we can always pick the logical Pauli-X operators to consist only of X's and I's, and the logical Z operators to contain only Z's and I's. Then, $\mathrm{CNOT}^{\otimes n}$ between two code blocks implements logical CNOT's between all pairs of encoded qubits simultaneously.

The $[[7,1,3]]$ Steane code hosts transversal logical Clifford gates. Remember

Transveral gates in the $[[7,1,3]]$ Steane code:

- $H^{\otimes 7}$: Swaps X and Z on each qubit, performs a logical Hadamard operation. (Thanks to the symmetry between X-type and Z-type stabilizers as well as the Pauli logical operators.)
- $S^{\otimes 7}$ where $S =$ phase gate $= \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$: Recall that $SXS^\dagger = Y$, $SZS^\dagger = Z$. Then

$$ZIZIZIZ \to ZIZIZIZ \text{ etc.} \tag{8.1}$$
$$XIXIXIX \to YIYIYIY = (XIXIXIX)(ZIZIZIZ) \in \mathcal{S} \text{ etc.,} \tag{8.2}$$

  so the stabilizers remain stabilizers. The logical operators are transformed as

$$\overline{Z} \to \overline{Z}, \tag{8.3}$$
$$\overline{X} = X^{\otimes 7} \to Y^{\otimes 7} = (iXZ)^{\otimes 7} = -i\overline{XZ} = -\overline{Y}\ . \tag{8.4}$$

  Therefore, $S^{\otimes 7}$ is a *logical adjoint-phase gate* $\overline{S}^\dagger$.

  Alternatively, $(S^\dagger)^{\otimes 7}$ is a transversal Clifford implementation of a logical phase gate $\overline{S}$.

- CNOTs between two code blocks implements a logical CNOT (as any CSS code).

The Hadamard, phase, and CNOT gates generate teh full multiqubit Clifford group. The Steane code can therefore implement all Clifford unitaries transversally.

The transversal Cliffords in the Steane code can be attributed to the high degree of symmetry in the stabilizer group and the logical operators:

- All stabilizer generators have *weight multiple of 4.* $\to$ Helps deal with phases since $i^4 = (-i)^4 = (-1)^4 = 1^4 = 1$.
- It's a *self-dual* CSS code ($H_X = H_Z$). $\to$ The X-type and Z-type stabilizers maps onto each other via Hadamards; helps ensure that $S^{\otimes n}$ is a logical operator.
- Logical X and Z operators *have the same support.* $\to$ ensures correct logical action of $H^{\otimes n}$ and $S^{\otimes n}$.

Some instances of the 2D color code also have these properties and can implement logical Cliffords transversally.

Exercise: Construct transversal logical Hadamard and phase gates for the 2D triangular color code studied in the previous chapter, on the hexagonal lattice. (Cf. reference [45].)

Sometimes it is possible to have a transversal implementation of a non-Clifford gate with non-Clifford single-qubit unitaries. An example of a non-Clifford single-qubit gate is the T

gate:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} . \tag{8.5}$$

Suppose we have a code where $|\overline{0}\rangle$ is a superposition of computational basis states with Hamming weight multiple of 8, and that $|\overline{1}\rangle$ is a superposition of computational basis states with Hamming weight equal to w modulo 8, with $w \neq 0$. Then $T^{\otimes n}|\overline{0}\rangle = |\overline{0}\rangle$ because each $(T|1\rangle)$ factor contributes a phase $e^{i\pi/4}$, which happens a number of times multiple of 8. Similarly, $T^{\otimes n}|\overline{1}\rangle = e^{iw\pi/4}|\overline{1}\rangle$. If $w = 1$, then $T^{\otimes n}$ implements a logical T gate.

Such a case happens for some CSS codes, such as the $[[15, 1, 3]]$ *quantum Reed-Muller code* (where $w = -1$). The stabililizer generators of this code can be chosen as:

$$\begin{gathered} ZZZZIIIIIIIIIII\ , \\ ZZIIZZIIIIIIII\ , \\ ZIZIZIZIIIIIII\ , \\ ZZIIIIIIZZIIII\ , \\ ZIZIIIIIZIZIII\ , \\ ZIIIZIIIZIIIZII\ , \\ ZZZZZZZZIIIIIII\ , \\ ZZZZIIIIZZZZIII\ , \\ ZZIIZZIIZZIIZZI\ , \\ ZIZIZIZIZIZIZIZ\ , \\ XXXXXXXXIIIIIII\ , \\ XXXXIIIIXXXXIII\ , \\ XXIIXXIIXXIIXXI\ , \\ XIXIXIXIXIXIXIX\ . \end{gathered} \tag{8.6}$$

In fact, this code happens to be a 3D color code.

Can we find a code with a transversal T gate and also transversal Cliffords? Such a code would support universal quantum computation with transversal gates!

Theorem 16 (Eastin-Knill)**.** *There exists no code that supports a transversal, universal set of logical gates.*

Proof. The set of all transversal gates is the compact Lie group $\mathcal{T} = U(d_1) \otimes \cdots \otimes U(d_n)$, where $d_i = \dim(\mathscr{H}_{P_i})$, with Lie algebra

$$\mathfrak{t} = \{A_1 + \cdots + A_n\ ,\ A_i = A_i^\dagger \text{ operator on } \mathscr{H}_{P_i}\}\ . \tag{8.7}$$

The subset $\mathcal{G}$ of $\mathcal{T}$ consisting of logical operators is a closed subgroup of $\mathcal{T}$ (a product of transversal logical operators remains a transversal logical operator, as well as the inverse of a transversal logical operator), so it is itself a compact Lie group. The group $\mathcal{G}$ has Lie algebra

$$\mathfrak{g} = \left\{T_1 + \cdots + T_n\ :\ [T_1 + \cdots + T_n, \Pi] = 0\,,\ T_i = T_i^\dagger \text{ acts on } \mathscr{H}_{P_i}\right\}\ . \tag{8.8}$$

But the erasure of each $\mathscr{H}_i$ is correctable; since T_i acts on $\mathscr{H}_i$, we have $\Pi T_i \Pi \propto \Pi$, so $\Pi(T_1 + \cdots + T_n)\Pi \propto \Pi$. Therefore, for all $\theta \in \mathbb{R}$, we have

$$\Pi e^{-i\theta\,(T_1+\cdots+T_n)}\Pi = \Pi e^{-i\theta\,\Pi(T_1+\cdots+T_n)\Pi}\Pi = (\text{phase})\,\Pi\ . \tag{8.9}$$

We find that the logical action of all gates generated by $\mathfrak{g}$ act trivially on the code space! Therefore, all gates in the connected component of $\mathcal{G}$ containing $\mathbb{1}$ act trivially on the code space. The logical action of $\mathcal{G}$ is therefore a discrete group, and it certainly cannot coincide with $U(d_L)$. ∎

Unfortunately, we cannot use transversal gates for universal fault-tolerant quantum computation.

In the following sections, we'll explore how to implement fault-tolerant computation using schemes such as *magic state distillation* that go beyond transversal operations.

Before moving on, let's cover two quick remarks on operations that are closely related to transversal operations.

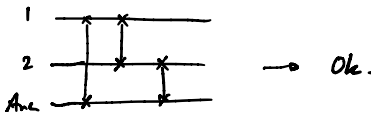
Recall that any CSS code has code words that are an even superposition of computational basis states corresponding to classical codes, $|u + C_X^\perp\rangle = \sum_{v \in C_X^\perp} |u + v\rangle$ with $u \in C_Z$, with logical basis states labeled by elements of $C_Z/C_X^\perp$. A *measurement* in the computational basis of all the physical qubits collapses the state onto $u + v$ with a random element $v \in C_X^\perp$. Decoding C_Z ensures we catch measurement errors, and the remaining syndrome of $C_X^\perp$ ($\subseteq C_Z$) reveals the logical computational basis state. **Therefore, any CSS code supports *transversal measurement* in the logical computational basis.**

Sometimes, logical operations can be performed by simply swapping qubits. E.g. in the $[[4, 2, 2]]$ code, swapping the 2nd and 3rd qubits performs a logical SWAP:

$$SWAP_{2,3}: \quad \overline{X}_1 \to \overline{X}_2 \; ; \quad \overline{X}_2 \to \overline{X}_1 \; ; \quad \overline{Z}_1 \to \overline{Z}_2 \; ; \quad \overline{Z}_2 \to \overline{Z}_1 \; . \tag{8.10}$$

Swapping qubits can increase the weight of existing errors if not implemented carefully, which affects fault tolerance.

- Physically swapping two qubits by physically moving them around (e.g., moving individual trapped atoms around) → OK.
- Physical gate generated by $XX + YY + ZZ$ within the computational subspace → Not OK, errors will spread.
- Ancillary-qubit-assisted SWAP (ancillary qubit initial state is unimportant):



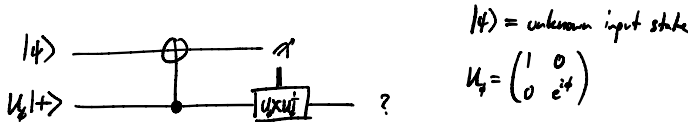
8.2 Universal logical computation with magic states

Magic state computation relies on two essential ingredients:

(1) *Gate teleportation*: Given access to some special state $|\text{magic}\rangle$, we can apply a non-Clifford gate onto some other unknown state $|\psi\rangle$ using only Clifford operations (gates and measurements).

(2) *Magic state distillation*: A scheme to prepare an encoded version of $|\text{magic}\rangle$ to be used with gate teleportation.

Consider the following circuit:



Suppose $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. After the CNOT gate, we find

$$(CNOT_{2,1})|\psi\rangle_1 \otimes (U_\phi|\psi\rangle_2) = \frac{1}{\sqrt{2}}(\alpha|00\rangle + \beta e^{i\phi}|01\rangle + \beta|10\rangle + \alpha e^{i\phi}|11\rangle) \ . \tag{8.11}$$

If the measurement of qubit #1 gives $|0\rangle$, the second qubit collapses onto the state $|\psi'\rangle \propto \alpha|0\rangle + \beta e^{i\phi}|1\rangle = U_\phi|\psi\rangle$. If the measurement gives $|1\rangle$, then the state of the second qubit after the correction becomes $|\psi\rangle' \propto (UXU^\dagger)(\beta|0\rangle + \alpha e^{i\phi}|1\rangle) = \alpha|0\rangle + \beta e^{i\phi}|1\rangle = U_\phi|\psi\rangle$. In all cases, the circuit prepares the state $U_\phi|\psi\rangle$ without knowing $|\psi\rangle$ a priori, but knowing U_ϕ, and given $U_\phi|+\rangle$ as an input.

This circuit applies U_ϕ onto the input state, without having to apply U_ϕ directly if given $U_\phi|\psi\rangle$ as an input! We might still have to apply $(U_\phi X U_\phi^\dagger)$, though.

There are gates U_ϕ such that U_ϕ is not a Clifford unitary, but $U_\phi X U_\phi^\dagger$ is! E.g., the T gate $T = \left(\begin{smallmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{smallmatrix}\right)$. The gate T is not a Clifford unitary, but $TXT^\dagger = \left(\begin{smallmatrix} 0 & e^{i\pi/4} \\ e^{i\pi/4} & 0 \end{smallmatrix}\right) = \frac{1}{\sqrt{2}}(X+Y)$, which is a Clifford gate.

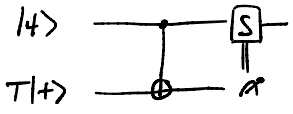
Exercise: Show that $\frac{1}{\sqrt{2}}(X+Y) = HS^\dagger HSH$, proving that the operator is a Clifford unitary.

Provided access to a copy of $T|+\rangle$, we can apply T on any unknown qubit using only Clifford operations.

More generally, gate teleportation can be used to implement any gate of the k-th level of the Clifford hierarchy $\mathcal{C}_k$, while consuming a suitable input state and assuming we can perform gates in $\mathcal{C}_{k-1}$. The ***k-th level of the Clifford hierarchy*** is defined recursively as

$$\mathcal{C}_k = \{U \ : \ UPU^\dagger \in \mathcal{C}_{k-1} \ \forall \ P \in \mathsf{P}_n\} \ ; \qquad\qquad \mathcal{C}_1 = \mathsf{P}_n \ . \tag{8.12}$$

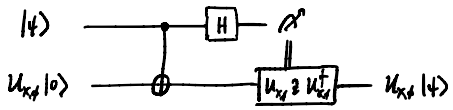
For the T gate, another convenient gate teleportation circuit is the following:



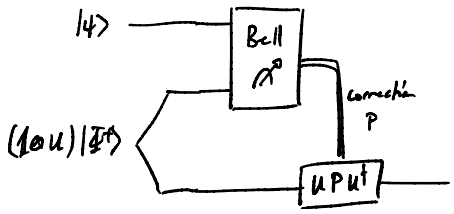
This circuit is convenient because the correction operation is a simple phase gate.

Exercise: Check that this circuit always outputs $T|\psi\rangle$, up to a global phase.

We can use other circuits to teleport gates that are not diagonal. E.g., if U is diagonal in the X basis, $[U_{X,\phi}, X] = 0$, we can use an analogous circuit:



For a general U that does not commute with either Z or X, we can always use the following scheme, based on the protocol of quantum teleportation:



Involving a Bell state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and performing a Bell measurement on the first two qubits.

The additional input state $U_\phi|+\rangle$, $U_{X,\phi}|0\rangle$, or $(\mathbb{1} \otimes U)|\Phi^+\rangle$ above have the "magical" property of enabling the application of a gate with a gate set that is less powerful. Such states are referred to as ***magic states***.

To apply a T gate, the corresponding magic state is

$$|T\rangle := T|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle) \ . \tag{8.13}$$

Suppose we had a reliable way of preparing an encoded state $|T\rangle$ in a code that supports transversal Clifford operations. We could do universal computation on the encoded states using transversal gates for the Cliffords and producing a magic state whenever we need to apply a T gate.

The general strategy to prepare magic states encoded in the desired code is to begin with noisy magic states and run them though a process called *distillation* to obtain fewer but more accurate magic states.

There are several magic state distillation protocols that are known. A general understanding of what can be done in terms of magic state distillation is still lacking; the field is still under active research.

Let's now study a particular scheme for magic state distillation, based on codes that have a transversal T gate. We've seen that some codes happen to have transversal implementations of the logical T gate, such as the $[[15, 1, 3]]$ quantum Reed-Muller code. Such codes can be leveraged to distill magic states, even though the code in which we're running our computation might be completely unrelated to the code with the transversal T gate.

Suppose we have qubits encoded in a code with transversal Cliffords and let's fix a $[[n_T, 1, d_T]]$ code that has a transversal implementation of a logical T gate. The idea is to use n_T imperfect encoded magic states to distill a single, more accurate encoded magic state. We proceed as follows:

1. (1) Prepare a logical $|+\rangle$ state in the $[[n_T, 1, d_T]]$ code with each of the n_T qubits encoded in our base code;

2. (2) Use n_T additional qubits encoded in our base code, initialized in some imperfect magic states, to apply $T^{\otimes n_T}$ onto the n_T qubits of the encoded $|+\rangle$ state using gate teleportation. If the magic states and Clifford gates where perfect, we would have a state $T|+\rangle$ encoded in the $[[n_T, 1, d_T]]$ code.

At this point, there are likely errors in our state since the original n_T magic states were faulty.

We could error-correct the $[[n_T, 1, d_T]]$ code to recover a noiseless magic state, provided there were $\leq (d_T - 1)/2$ faults in the original magic states.

Actually, it suffices to *detect* errors, which we can do up to weight $\leq d_T - 1$; if we detect any error, we throw the state away and start again. We catch more errors in this fashion. We then:

1. (3) Run error detection. If an error is detected, we start again.

Finally, we need to decode the $[[n_T, 1, d_T]]$ code to get a "raw" magic state encoded only in our base code.

1. (4) Decode the $[[n_T, 1, d_T]]$ code.

If the original, noisy magic states have a noise rate p, then the new ones have noise $O(p^{d_T})$.

In the case of the $[[15, 1, 3]]$ quantum Reed-Muller code, applying $T^{\otimes 15}$ implements, in fact, a logical $T^\dagger$. Therefore, we need to include a logical S correction immediately after step (2), recalling $ST^\dagger = T$. The application of the S gate is simple because it has a transversal, Clifford implementation in the $[[n_T, 1, d_T]]$ code; indeed, applying ST on each qubit of the $[[n_T, 1, d_T]]$ code is the same as directly applying $T^\dagger$ on each of those qubits, which applies the logical T as desired.

This protocol can still be heavily optimized. It can even be written as a protocol on five qubits.

Other magic state distillation protocols offer different distillation factors. E.g. $15 \to 1$, $5 \to 1$, $116 \to 12$. We can also distill one type of magic state (e.g., for a T gate) to another type of magic state (e.g., for a CCZ gate): 512 ($|T\rangle$) $\to$ 10 ($|\text{CCZ}\rangle$), etc.

Magic state distillation is generally an expensive process. That's one reason why we often worry about the number of T gates that are present in a circuit we'd like to run.

8.3 Fault-tolerant quantum computation with the surface code

We can now begin assembling the elements required to construct a fault-tolerant scheme to perform quantum computation with the surface code.

We suppose we have a large 2D array of qubits that are available. Qubits are encoded on small surface code patches:

We've seen that, as long as our hardware qubits are below a relevant threshold, we can engineer arbitrarily good logical qubits.

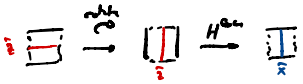
We can extend or deform individual patches as we like, by including more stabilizers in the code:

Exercise: Convince yourself how one can extend, deform, or compress patches by including additional stabilizers or removing stabilizers from consideration, following Refs. [37, 42].

A single qubit can be measured by measuring all its qubits in the computational basis, since it is a CSS code.

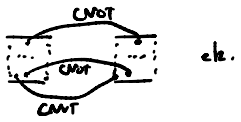
The logical X and Z operators for each qubit can be applied with a transversal operation; they are strings of operators across the patch. But instead of applying Pauli operations directly on the hardware qubits, it is usually simpler to track an overall Clifford transformation in software; we talk about "updating the Pauli frame," as if we changed the logical "coordinate system" (choice of logical Pauli operators) in which we operate.

Implementing a logical Hadamard H is somewhat annoying, unless we can rotate all qubits physically (which we usually cannot do):



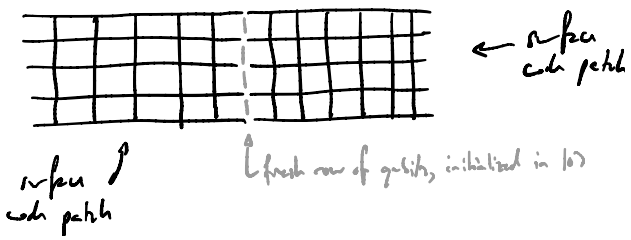
But again, H is a Clifford operation, so we can also track its effect in software, as long as we are able to update correspondingly all subsequent gates we'd like to apply to the qubits.

As a CSS code, the surface code has a transversal CNOT between two patches, implemented by applying CNOTs on pairs of qubits, one from each patch:



But this implementation, even if it is transversal, is often impractical, as it requires applying gates between distant qubits. Thankfully, there is another, more practical way of implementing CNOT gates.

We introduce a new operation between patches: ***lattice merging***. Let's consider two patches aligned over a pair of rough boundaries, and let's suppose we have a fresh row of qubits between these patches:



In a *lattice merge* procedure along the rough boundary (*rough lattice merge*), we perform the following steps:

(1) We initialize the fresh qubits in $|0\rangle$;

(2) We measure new bulk surface code star stabilizers across the earlier boundaries, involving the new fresh qubits;

(3) Perform error correction.

(As we perform error correction, we need to measure the stabilizers multiple times to counter the effect of measurement errors.)

The product of all the new stabilizers we measured in this fashion is $\overline{X}_1 \otimes \overline{X}_2$! The lattice merging operation *performs a logical measurement* of $\overline{X}_1 \otimes \overline{X}_2$ between the two patches.

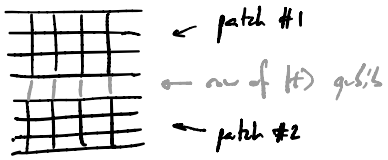
The result is a new patch of a different width. If the initial states were $|\psi\rangle_1 \otimes |\phi\rangle_2$, with $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and $|\phi\rangle = \alpha'|0\rangle + \beta'|1\rangle$, then the new patch is left in the logical state

$$|\psi\rangle_1 \otimes |\phi\rangle_2 \xrightarrow{\text{lattice merge}} \alpha|\phi\rangle + (-1)^m \beta(X|\phi\rangle) = \alpha'|\psi\rangle + (-1)^m \beta'(X|\psi\rangle) \ , \qquad (8.14)$$

where m is the outcome of the measurement of $\overline{X}_1 \otimes \overline{X}_2$, obtained as the product of all the new stabilizer outcomes.

Observe that there is no risk of introducing a logical Z error because of the way we might have messed up the X stabilizer outcomes along the vertical measurement path, because logical Z operators run horizontally from rough boundary to rough boundary.

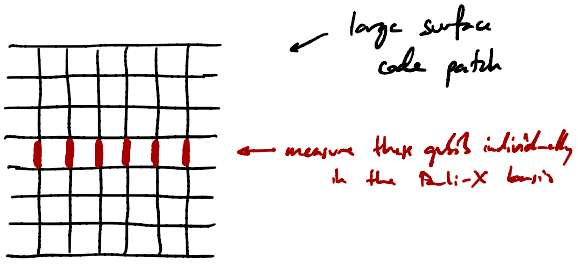
Similarly, we can perform a *smooth lattice merge* along a smooth boundary with an additional row of qubits in the $|+\rangle$ state, and where we measure the new plaquettes that are formed by aligning the patches with the fresh row of qubits:



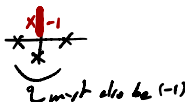
The operation performed after the smooth lattice merge (including any corrections/redefinition of logical operators) is a logical $\overline{Z}_1\overline{Z}_2$ measurement.

We also introduce an operation called ***lattice splitting***. This operation splits a large surface code patch into two smaller patches.

The operation of *smooth lattice splitting* consists in measuring a row of qubits connecting two rough boundaries in the X basis, creating two new smooth boundaries:



Error correction proceeds on the individual patches, but we update the expected value of the new boundary stabilizers to (-1) whenever the random outcome of an X qubit measurement was -1:



(Remember, the original state is an eigenstate of all original star operators.)

Logical Z operators (horizontal Z strings) are not affected by these measurements, so they still reveal the same value both above and below the lattice split. The original logical X straddles the split, indicating that we've entangled the two new qubits. One can verify that the operation performed by the smooth lattice splitting is:

$$\alpha|0\rangle + \beta|1\rangle \quad \xrightarrow{\text{smooth split}} \quad \alpha|00\rangle + \beta|11\rangle \ . \tag{8.15}$$

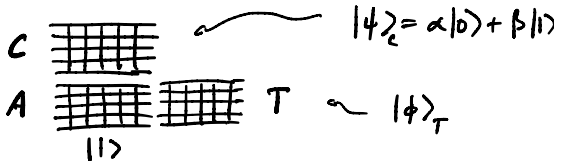
Similarly, a *rough lattice split* performs the operation

$$\alpha|+\rangle + \beta|-\rangle \quad \xrightarrow{\text{rough split}} \quad \alpha|++\rangle + \beta|--\rangle \ . \tag{8.16}$$

The lattice split and merge operations are collectively referred to as ***lattice surgery*** operations.

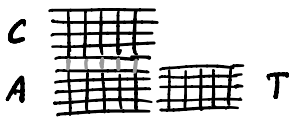
Lattice surgery operations can be employed, for instance, to perform a CNOT gate between two patches:

(1) Say we have two patches C and T encoding two qubits to act as the control and the target qubits of a CNOT gate. We add an ancillary qubit, encoded in a patch A, and prepared in the logical state $|+\rangle$. We arrange the patches as follows:



Suppose the initial state of C is $|\psi\rangle_C = \alpha|0\rangle + \beta|1\rangle$, and let $|\phi\rangle_T$ be the initial state of T.

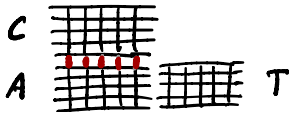
(2) We perform a *smooth lattice merge* between C and A:



The new combined patch (CA) is left in the state

$$\alpha|0\rangle_{(CA)} + \beta|1\rangle_{(CA)} \tag{8.17}$$

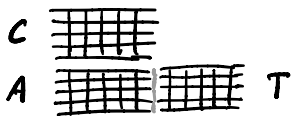
(3) Perform a *smooth lattice split* of C and A:



The state of C and A is now entangled:

$$\alpha|00\rangle_{CA} + \beta|11\rangle_{CA} \ . \tag{8.18}$$

(4) Perform a *rough lattice merge* between A and T:



The state of C and (AT) is now

$$\begin{aligned} & (\alpha|00\rangle_{CA} + \beta|11\rangle_{CA}) \otimes |\phi\rangle_T \\ [2ex] \quad & \xrightarrow{\text{rough merge}} \quad \alpha|0\rangle_C|\phi\rangle_{(AT)} + \beta|1\rangle(X|\phi\rangle_{(AT)}) = \mathrm{CNOT}(|\psi\rangle \otimes |\phi\rangle) \ . \end{aligned} \tag{8.19}$$

In the steps above, we've implicitly assumed all necessary correction operations and/or redefinition of the logical operators have been applied.

The result produces the output of the CNOT onto C and a larger merged patch (AT). We can reduce the size of the latter patch if we need the result on the original patch location T.

In summary, lattice operations (resizing, moving, etc.) along with lattice surgery operations provide fault-tolerant implementations of all *Clifford gates*.

A provision of encode *magic states*, obtained via magic state distillation, enables universal fault-tolerant quantum computation.

8.4 Further reading

We point in particular to Chapters 11 and 13 of Gottesman's book [1].

The original Eastin-Knill paper is a great read [54]. Further restrictions on logical computation include restrictions on logical gates that can be implemented with short-depth circuits on topological codes (Bravyi-König) [55] and robust versions of the Eastin-Knill theorem for approximate quantum error correction [56].

We point to Ref.[57] for deeper details on surface code lattice surgery operations. Proposals for fault-tolerant computation with the surface code include Refs. [41, 42]. We point to Ref. [48] for proposals for fault-tolerant quantum computation with the color code.

Magic state distillation is also a very active topic of research. See for instance Refs. [58, 59] that optimize magic state distillation protocols and identifies families of codes with transversal T gates.

There are alternative methods of applying non-Clifford gates. These include a method with subsystem codes based on a 3D color code (e.g. [45]), braiding anyons in a suitable topological state of matter (cf. EC Zoo), and concatenating codes with different types of transversal gates [60].

Chapter 9

Bosonic codes

This course has been mostly focused on qubit codes, given that a lot is known about them. The surface code is a prominent example and has been really extensively studied!

Can we construct interesting codes using other quantum systems as elementary constituents? *Qudit codes*, with a local dimension $q \geq 3$, have a rich structure that we won't have the time to explore in this lecture. Instead, we'll focus now on building codes with continuous-variable systems we find commonly across quantum devices: *quantum harmonic oscillators*.

9.1 Quantum bosonic modes.

A bosonic mode corresponds to an infinite-dimensional Hilbert space. Some mathematical subtleties related to this fact will be glossed over in favor of clarity and simplicity.

The Hilbert space is $L^2(\mathbb{R} \to \mathbb{C})$, i.e., the space of square-integrable, complex-valued functions over the real line.

The *position "basis"* is formed of *position "eigenstates"* $|x\rangle$, used to express a general state $|\psi\rangle$ as

$$|\psi\rangle = \int_{\mathbb{R}} dx\, |x\rangle\langle x\,|\,\psi\rangle = \int_{\mathbb{R}} dx\, \psi(x)\, |x\rangle \ , \tag{9.1}$$

where $\psi(x) = \langle x\,|\,\psi\rangle$ is the position representation of $|\psi\rangle$ (the wave function).

The position states obey, as a "basis,"

$$\langle x\,|\,y\rangle = \delta(x-y) \ ; \qquad\qquad \int dx\, |x\rangle\langle x| = \mathbb{1} \ , \tag{9.2}$$

where $\delta(\cdot)$ is the Dirac delta "function."

The *position operator* is

$$\hat{x} = \int_{\mathbb{R}} dx\, x\, |x\rangle\langle x| \ . \tag{9.3}$$

The *momentum "states"* $|p\rangle$ are related by a Fourier transform to the position "states":

$$|p\rangle = \frac{1}{\sqrt{2\pi}} \int dx\, e^{ipx}\, |x\rangle \ . \tag{9.4}$$

The momentum states also obey

$$\langle p|p'\rangle = \delta(p-p') \ ; \qquad \int dp\, |p\rangle\langle p| = \mathbb{1} \ . \tag{9.5}$$

The position and momentum operators obey the *fundamental commutation relation*

$$[\hat{x},\hat{p}] = i \quad (\equiv i\mathbb{1}) \ . \tag{9.6}$$

The *harmonic oscillator* is given by the Hamiltonian

$$\hat{H} = \frac{\omega}{2}(\hat{x}^2+\hat{p}^2) \ . \tag{9.7}$$

In your basic quantum mechanics course, you've seen that $\hat{H}$ is diagonal in the basis of *Fock states* $|n\rangle$, with $n=0,1,\ldots$, which are eigenstates of the *number operator* $\hat{n} = \frac{1}{2}(\hat{p}^2+\hat{x}^2-1)$. The number operator is more conveniently written as

$$\hat{n} = \hat{a}^\dagger\hat{a} \ ; \qquad \hat{a} = \frac{1}{\sqrt{2}}(\hat{x}+i\hat{p}) \ . \tag{9.8}$$

The $\hat{a},\hat{a}^\dagger$ are the *bosonic mode operators*, or *annihilation and creation operators*; they obey

$$\hat{a}|n\rangle = \sqrt{n}\,|n-1\rangle \ ; \qquad \hat{a}^\dagger|n\rangle = \sqrt{n+1}\,|n+1\rangle \ . \tag{9.9}$$

In particular, $\hat{a}|0\rangle = 0$, where $|0\rangle$ is the *vacuum state* and the *ground state* of the quantum harmonic oscillator $\hat{H}$. We obtain all other Fock states starting from $|0\rangle$ by acting iteratively with $\hat{a}^\dagger$.

The $\hat{x}$ and $\hat{p}$ operators generate *shifts in momentum and position space*:

$$\begin{aligned} e^{-ia\hat{p}}\,|x\rangle = |x+a\rangle \ ; \quad & e^{ib\hat{x}}\,|x\rangle = e^{ibx}\,|x\rangle \ ; \\ e^{ia\hat{p}}\,|p\rangle = e^{-iap}\,|p\rangle \ ; \quad & e^{ib\hat{x}}\,|p\rangle = |p+b\rangle \qquad (a,b\in\mathbb{R}) \, . \end{aligned} \tag{9.10}$$

We can group both types of shifts together and define the *displacement operator*:

$$\hat{D}(a,b) = e^{-ia\hat{p}+ib\hat{x}} \ . \tag{9.11}$$

(One often encounters the displacement operator in terms of a complex number $\alpha\in\mathbb{C}$, encoding the a,b above by $\alpha = a\sqrt{2}+ib\sqrt{2}$ and yielding the alternative expression $\hat{D}(\alpha) = e^{\alpha\hat{a}^\dagger-\alpha^*\hat{a}}$. We'll come back to this form of the displacement operator later in this chapter.)

To help us visualize bosonic states, we define the *Wigner function* of any given state $\hat{\rho}$ as

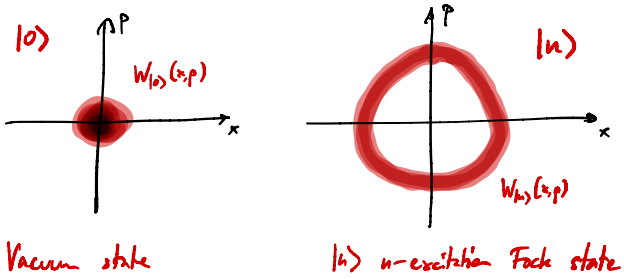
$$\begin{aligned} W_{\hat{\rho}} &= \frac{1}{4\pi^2}\int dadb\, e^{-i(xb-pa)}\ \mathrm{tr}\Big\{\hat{\rho}\,\hat{D}(a,b)\Big\} \\ [1ex] &= \text{Fourier transform of } \mathrm{tr}\{\hat{\rho}\hat{D}(a,b)\} \\ [1ex] &= \frac{1}{2\pi}\int dy\, e^{ipy}\,\langle x-\tfrac{y}{2}\,|\,\hat{\rho}\,|\,x+\tfrac{y}{2}\rangle \ . \end{aligned} \tag{9.12}$$

The Wigner function reveals the distributions of ρ in the $\hat{x}$ as well as in the $\hat{p}$ bases, by

"tracing out" the other variable:

$$\langle x|\hat{\rho}|x\rangle = \int dp\, W(x,p)\ ; \qquad \langle p|\hat{\rho}|p\rangle = \int dx\, W(x,p)\ . \tag{9.13}$$

The Wigner function is useful to visualize states. For example:



Bosonic modes commonly suffer one of the following errors. Error processes can be described in terms of jump operators $\{\hat{K}_j\}$ in a Lindbladian

$$\mathcal{L}[\hat{\rho}] = \kappa \sum_j (\hat{K}_j \hat{\rho} \hat{K}_j^\dagger - \frac{1}{2}\{\hat{K}_j^\dagger \hat{K}_j, \hat{\rho}\})\ , \tag{9.14}$$

where κ is a fixed noise rate, or in terms of a quantum channel $\mathcal{E}$ with Kraus operators which we think of as error operators that can be applied to the system.

- *loss* causes excited states to relax towards the ground state, losing photons. The simple jump operator is the annihilation operator $\hat{a}$, with Lindbladian $\mathcal{L}[\hat{\rho}] = \kappa(\hat{a}\hat{\rho}\hat{a}^\dagger - \frac{1}{2}\{\hat{a}^\dagger\hat{a}, \hat{\rho}\})$.
- *displacements* are generated by $\hat{x}$ and $\hat{p}$ operators, or, more generally, are implemented by $\hat{D}(a,b)$. If a, b are random and distributed according to a Gaussian distribution, the noise corresponds to *thermal noise*.
- *dephasing* errors occur if $H \propto \hat{a}^\dagger\hat{a}$ is applied for some (typically small) random amount of time, causing loss of coherence of the state in the Fock basis. Equivalently, the environment proceeds in a weak measurement of the energy of the system, causing loss in coherence.
- Some multimode codes inspired by qubit codes can correct a set of errors that affect exclusively a small subset of the modes, similarly to common qubit error models. We can think, for instance, about correcting the *erasure of one of the modes*.

9.2 Bosonic stabilizer codes

What is the bosonic equivalent of the Pauli X and Z operators, which we used to build stabilizer groups? Good candidates are the *bosonic position and momentum shift operators* defined in (9.10). To see why the bosonic shift operators are natural bosonic equivalents of the Pauli X and Z operations, observe that they naturally generalize the following qubit operations, where X shifts to the next computational basis state (modulo 2) and Z applies a computational-basis-state-dependent phase:

$$\begin{aligned} X|c\rangle &= |c \oplus 1\rangle\ ; \qquad Z|c\rangle = e^{i\pi c}|c\rangle\ ; \\ X|\pm_j\rangle &= e^{i\pi j}|\pm_j\rangle\ ; \quad Z|\pm_j\rangle = |\pm_{j\oplus 1}\rangle \qquad (c, j \in \{0,1\})\ , \end{aligned} \tag{9.15}$$

employing the notation $|\pm_j\rangle = H|j\rangle = \left\{ \begin{smallmatrix} |+\rangle \ (j=0) \\ |-\rangle \ (j=1) \end{smallmatrix} \right\}$ for the Hadamard basis states.

The bosonic shift operators are unitaries that we'll use to build stabilizer groups.

If the logical encoded quantum system is finite-dimensional, we'll call the code a *digital stabilizer code*. If we encode another mode (or collection of modes), it's an *analog stabilizer code*.

Several analog stabilizer codes are simple analogs of qubit codes. Say, the *quantum repetition code*, in the position states, is:

$$|x\rangle \quad \to \quad |x\rangle \otimes |x\rangle \otimes |x\rangle \ . \tag{9.16}$$

Code states are $+1$ eigenstates of $e^{-i\theta\hat{x}_1}e^{i\theta\hat{x}_2}$ and of $e^{-i\theta\hat{x}_2}e^{i\theta\hat{x}_3}$ (for $\theta \in \mathbb{R}$), which generate a stabilizer group.

The code can correct shifts in position of any of the individual modes and of arbitrary magnitude:

1. Measure $\hat{x}_1 - \hat{x}_2$ and $\hat{x}_2 - \hat{x}_3$, which commute;
2. From the real-valued measurement outcomes, determine which mode was shifted and the shift magnitude;
3. Apply a suitable shift to the affected mode to correct the error.

Let's inspect the Knill-Laflamme conditions. If $E_{a,i} \propto e^{-ia\hat{p}_i}$ and $\Pi = \int dx \, |x,x,x\rangle\langle x,x,x|$, and fixing $i = 1$, $i' = 2$ for convenience, we find:

$$\begin{aligned} \Pi E_{a,i}^\dagger E_{a,i} \Pi &\propto \int dxdy \, |x,x,x\rangle\langle y,y,y| \, \langle x,x,x| \, e^{-i(a\hat{p}_i - a'\hat{p}_{i'})} \, |y,y,y\rangle \\ &= \int dxdy \, |x,x,x\rangle\langle y,y,y| \, \langle x,x,x| \, y+a, y-a', y\rangle \\ &= \int dxdy \, |x,x,x\rangle\langle y,y,y| \, \delta(y+a-x) \, \delta(y-a'-x) \, \delta(x-y) \ . \\ &= \delta(a) \, \delta(a') \, \Pi \quad \propto \Pi \ . \end{aligned} \tag{9.17}$$

Here, we already hit upon difficulties dealing with infinite quantities. The argument above is likely to have made a mathematically-minded reader uneasy, for multiple good reasons:

- How did we get "raw" Dirac deltas pop out, while we started with what looks like a valid operator?
- Even Π is not well defined to start off with! For a project, we should have $\Pi^2 = \Pi$, but we find $\Pi^2 = \int dxdy \, |x,x,x\rangle\langle y,y,y|\langle x,x,x| \, y,y,y\rangle = \int dxdy \, |x,x,x\rangle\langle y,y,y| \, [\delta(x-y)]^3 = [\delta(0)]^2\Pi$! This is due to the fact that the states used to define these codes are not normalizable.

We're usually fine as long as we keep in mind the operational interpretation of the quantities we write down, and remember that states like $|x\rangle$ are ideal, infinite-energy, nonnormalizable states that would have to be approximated physically with finite-energy valid quantum states.

By concatenating two copies of the $1 \to 2$ analog quantum repetition code, we find a bosonic version of the $[[4,1,2]]$ code, which we'll call a $[[4,1,2]]_{\mathbb{R}}$ bosonic analog code:

$$\begin{aligned} |x\rangle \quad \to \quad |\psi_x\rangle &= \Big[\int dy\, e^{ixy}\, |y,y\rangle\Big]^{\otimes 2} \\ &= \int dy \int dz\, e^{ix(y+z)}\, |y,y,z,z\rangle \ . \end{aligned} \tag{9.18}$$

This code can *detect any error* on a single mode. Equivalently, it can correct the *erasure* of any single mode.

The stabilizer group of the $[[4,1,2]]_{\mathbb{R}}$ analog code is generated by the displacements $e^{i\theta(\hat{x}_1-\hat{x}_2)}$, $e^{i\theta(\hat{x}_3-\hat{x}_4)}$, and $e^{-i\theta(\hat{p}_1+\hat{p}_2-\hat{p}_3-\hat{p}_4)}$:

$$\mathcal{S} = \Big\langle e^{i\theta(\hat{x}_1-\hat{x}_2)}\ ,\ e^{i\theta(\hat{x}_3-\hat{x}_4)}\ ,\ e^{-i\theta(\hat{p}_1+\hat{p}_2-\hat{p}_3-\hat{p}_4)} \Big\rangle \qquad (\theta \in \mathbb{R})\,. \tag{9.19}$$

Since the stabilizer group is continuous, it makes sense to study the infinitesimal generators of the group, called *nullifiers*. They are mutually commuting. The codewords are $+1$ eigenstates of the stabilizers, so they must be 0-eigenstates of the nullifiers: *nullifiers annihilate codewords.*

The nullifiers of the $[[4,1,2]]_{\mathbb{R}}$ code are spanned by

$$\{\hat{x}_1 - \hat{x}_2\,,\ \hat{x}_3 - \hat{x}_4\,,\ \hat{p}_1 + \hat{p}_2 - \hat{p}_3 - \hat{p}_4\}\ . \tag{9.20}$$

We can similarly construct analog versions of qubit codes:

- A $[[9,1,3]]_{\mathbb{R}}$ version of the $[[9,1,3]]$ Shor code;
- A $[[5,1,3]]_{\mathbb{R}}$ version of the $[[5,1,3]]$ perfect code;
- An analog surface code;
- etc.

The way analog stabilizer codes ensure the stabilizer elements commute is by ensuring the nullifiers commute. Can we get stabilizer generators to commute without requiring the commutation of corresponding nullifiers?

Displacements corresponding to noncommuting generators may may still commute for particular shift values. Consider a single mode, on which $\hat{x}$ and $\hat{p}$ shifts obey the *Weyl group commutation relations*:

$$e^{-ia\hat{p}} e^{ib\hat{x}} = e^{iab} e^{ib\hat{x}} e^{-ia\hat{p}}\ . \tag{9.21}$$

We find that $[e^{-ia\hat{p}}, e^{ib\hat{x}}] = 0$ if $ab = 0 \mod 2\pi$, despite $\hat{x}$ and $\hat{p}$ not commuting.

For instance, we can choose $S_1 = e^{i\sqrt{2\pi N}\hat{x}}$ and $S_2 = e^{-i\sqrt{2\pi N}\hat{p}}$ to generate a stabilizer group

$$\mathcal{S}_{\text{GKP}} = \langle S_1, S_2\rangle = \big\{S_1^k S_2^\ell\ :\ k,\ell \in \mathbb{Z}\big\}\ . \tag{9.22}$$

This group is discrete, so it does not have nullifiers.

We see that $\mathcal{S}_{\text{GKP}}$ stabilizes a logical N-dimensional qudit, defined by logical operators $\overline{Z} = S_1^{1/N} = e^{i\sqrt{2\pi/N}\hat{x}}$ and $\overline{X} = S_2^{1/N} = e^{-i\sqrt{2\pi/N}\hat{p}}$. We can verify that $\overline{Z}$ and $\overline{X}$

individually commute with all stabilizers and satisfy $\overline{XZ} = e^{i2\pi/N}\overline{ZX}$, as required by qudit X and Z Weyl operators (also known as *modular qudit operators*). For a qubit ($N = 2$), and we find as expected that $\overline{X}$ and $\overline{Z}$ anticommute, $\overline{XZ} = -\overline{ZX}$.

This code is known as the ***Gottesman-Kitaev-Preskill (GKP) code***, in honor of its discoverers. In one of its simplest forms (with $N = 2$), it encodes one qubit into a single bosonic mode.

The code states of the GKP code must have both a position representation $\psi(x)$ that is $\sqrt{2\pi N}$-periodic as well as a momentum representation $\tilde{\psi}(p)$, the Fourier transform of $\psi(x)$, that is also $\sqrt{2\pi N}$-periodic. (Indeed, this periodicity is enforced by the chosen stabilizers.) We know that a Dirac delta comb has this property, and indeed it gives the position (and momentum representation of the GKP code states. For $N = 2$, we find:

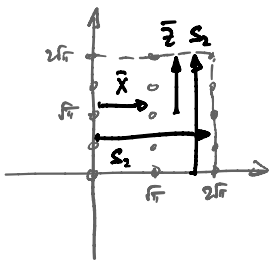
$$\begin{aligned}
|\overline{0}_{\mathrm{GKP}}\rangle &= \sum_{\ell\in\mathbb{Z}} |x{=}(2\ell)\sqrt{\pi}\rangle && \propto \quad \sum_{\ell'\in\mathbb{Z}} |p{=}\ell'\sqrt{\pi}\rangle \ ; \\
|\overline{1}_{\mathrm{GKP}}\rangle &= \sum_{\ell\in\mathbb{Z}} |x{=}(2\ell+1)\sqrt{\pi}\rangle && \propto \quad \sum_{\ell'\in\mathbb{Z}} (-1)^{\ell'} |p{=}\ell'\sqrt{\pi}\rangle \ ; \\
|\overline{+}_{\mathrm{GKP}}\rangle &\propto \sum_{\ell\in\mathbb{Z}} |x{=}\ell\sqrt{\pi}\rangle && \propto \quad \sum_{\ell'\in\mathbb{Z}} |p{=}(2\ell')\sqrt{\pi}\rangle \ ; \\
|\overline{-}_{\mathrm{GKP}}\rangle &\propto \sum_{\ell\in\mathbb{Z}} (-1)^{\ell} |x{=}\ell\sqrt{\pi}\rangle && \propto \quad \sum_{\ell'\in\mathbb{Z}} |p{=}(2\ell'+1)\sqrt{\pi}\rangle \ .
\end{aligned} \tag{9.23}$$

The Wigner function of GKP states has a distinctive grid structure. For $|\overline{0}\rangle_{\mathrm{GKP}}$ and $|\overline{1}\rangle_{\mathrm{GKP}}$ with $N = 2$, we find:

$$W_{|\bar{j}_{\mathrm{GKP}}\rangle}(x,p) \propto \sum_{k,\ell\in\mathbb{Z}} (-1)^{k\cdot\ell}\, \delta\Big(p - \frac{\sqrt{\pi}}{2}\, k\Big)\, \delta\Big(x - \sqrt{\pi}(\ell + j)\Big) \ . \tag{9.24}$$

The Wigner functions of the GKP-encoded qubit Pauli eigenstates are depicted in Fig. 9.1.

The action of logical operators on the Wigner function can be visualized as:



GKP codes can protect against small displacements. The smallest displacements that act nontrivially on the code space are shifts by $\sqrt{\pi}$ in either quadrature. GKP codes can thus correct shifts up to $\frac{\sqrt{\pi}}{2}$.

The ideal GKP code words have infinite energy and are not normalizable. To realize them physically, we replace the position "eigenstates" $|x\rangle$ by thin Gaussians $\sim e^{-x^2/(2\Delta^2)}$, and we apply a global Gaussian envelope $\sim e^{-\kappa^2 x^2/2}$ to damp off peaks at positions far away from the origin:

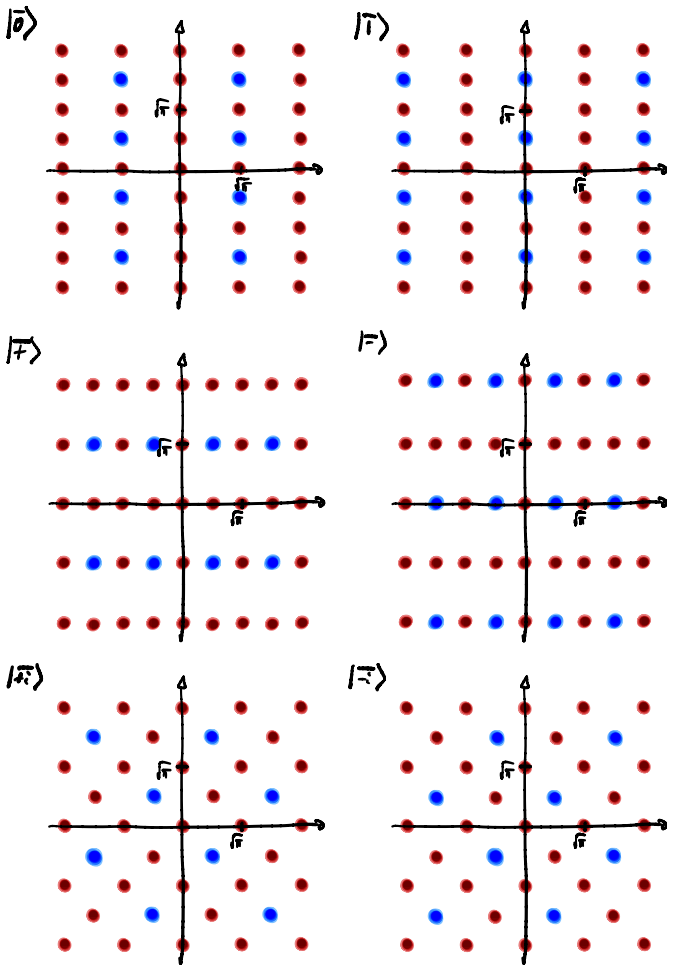
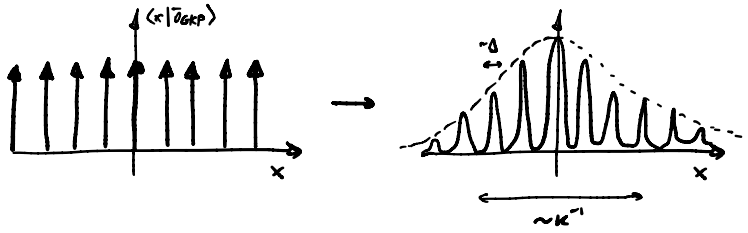


Figure 9.1: Wigner function representation of the GKP encoded states associated with the logical qubit states $|0\rangle, |1\rangle$ (Z eigenstates), $|\pm\rangle$ (X eigenstates), and $|\pm i\rangle$ (Y eigenstates).



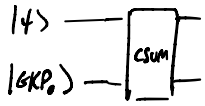
Part of the difficulty of engineering GKP code states is the need to measure not $\hat{x}$ and $\hat{p}$, but rather *the values of* $\hat{x}$ *and* $\hat{p}$ *modulo* $2\sqrt{\pi}$. Techniques to do so include using an ancillary qubit/qudit/mode to perform controlled-displacements.

More generally, GKP codes have a beautiful description in terms of lattices in $\mathbb{R}^{2n}$, representing the phase space of n bosonic modes, with a symplectic structure.

GKP states can also be used to construct an encoding that protects a *logical bosonic mode*. Consider the 2-mode CSUM gate defined through the following action by conjugation on the position and momentum operators of each mode:

$$\begin{aligned} \hat{x}_1 &\to \hat{x}_1 \ ; && (= \ \mathrm{CSUM}\, \hat{x}_1\, \mathrm{CSUM}^\dagger) \\ \hat{p}_1 &\to \hat{p}_1 - \hat{p}_2 \ ; \\ \hat{x}_2 &\to \hat{x}_1 + \hat{x}_2 \ ; \\ \hat{p}_2 &\to \hat{p}_2 \ . \end{aligned} \tag{9.25}$$

We define an encoding of a logical mode $|\psi\rangle$ as:



where $|\mathrm{GKP}_0\rangle = \sum_{\ell\in\mathbb{Z}} |\ell\sqrt{2\pi}\rangle$ is the canonical GKP state/grid state.

This code is called a ***GKP-stabilizer code***, whose stabilizer group is

$$\begin{aligned} \mathcal{S} &= \mathrm{CSUM}\, \mathcal{S}_{\mathrm{GKP}}\, \mathrm{CSUM}^\dagger \\ &= \left\langle \mathrm{CSUM}\, e^{-i\hat{p}_2\sqrt{2\pi}}\, \mathrm{CSUM}^\dagger \, , \ \mathrm{CSUM}\, e^{i\hat{x}_2\sqrt{2\pi}}\, \mathrm{CSUM}^\dagger \right\rangle \ . \end{aligned} \tag{9.26}$$

Suppose the noise effects small displacements δx_1, δp_1, δx_2, and δp_2 in the quadratures, distributed as Gaussians $\Pr[\delta x_i] \sim e^{-(\delta x_i)^2/(2\sigma^2)}$, $\Pr[\delta p_i] \sim e^{-(\delta p_i)^2/(2\sigma^2)}$. After the encoding, noise, and after we apply an additional $\mathrm{CSUM}^\dagger$ to decode the logical mode, the four quadratures transform as

$$\begin{aligned} &\hat{x}_1 \to \hat{x}_1 + \delta x_1 \ ; && \hat{x}_2 \to \hat{x}_2 + \delta x_2 - \delta x_1 \ ; \\ &\hat{p}_1 \to \hat{p}_1 + \delta p_1 + \delta p_2 \ ; && \hat{p}_2 \to \hat{p}_2 + \delta p_2 \ . \end{aligned} \tag{9.27}$$

We measure the stabilizers by measuring $\hat{x}_2$, $\hat{p}_2$ modulo $\sqrt{2\pi}$ after applying $\mathrm{CSUM}^\dagger$. We measure, assuming that δx_i, δp_i are small,

$$x_{\mathrm{obs}} = \delta x_2 - \delta x_1 \ ; \qquad\qquad p_{\mathrm{obs}} = \delta p_2 \ . \tag{9.28}$$

We can remove the effect of δp_2 on the logical mode by applying a shift $-\delta p_2$ on the first mode, such that

$$\hat{p}_1 \xrightarrow{\text{enc, noise, CSUM}^\dagger} \hat{p}_1 + \delta p_1 + \delta p_2 \xrightarrow{\text{correct } -\delta p_2} \hat{p}_1 + \delta p_1 \ . \tag{9.29}$$

The effect of δp_1 remains; we don't achieve any noise reduction in the momentum shift on the logical mode, but the noise is also not amplified.

We can apply a shift $x \to x - x_{\text{obs}}/2$ on the first mode, such that the logical $\hat{x}$ transforms overall as

$$\hat{x}_1 \xrightarrow{\text{enc, noise, CSUM}^\dagger} \hat{x}_1 + \delta x_1 \xrightarrow{\text{correct } -x_{\text{obs}}/2} \hat{x}_1 + \frac{\delta x_1 + \delta x_2}{2} \ . \tag{9.30}$$

We're left with the position displacement error $(\delta x_1 + \delta x_2)/2$. But if we assume that δx_1, δx_2 are independent random variables, then $(\delta x_1 + \delta x_2)/2$ has *half the variance* of the individual variables x_1 and x_2. I.e., the noise magnitude decreased from σ to $\sigma/2$.

With a smarter choice of encoding gates and modes, we can achieve noise reduction in both quadratures.

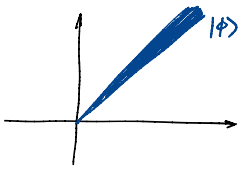
9.3 Bosonic Fock-state codes

While GKP codes are naturally described with position or momentum states, certain bosonic codes have some rotational symmetry in phase space and are more conveniently described in terms of Fock states.

A useful set of states that transforms naturally under phase space rotations are the *phase "states:"*

$$|\phi\rangle = \frac{1}{\sqrt{2\pi}} \sum_{n \geq 0} e^{i\phi n} |n\rangle \ . \tag{9.31}$$

Similarly to position "states," phase "states" cannot be normalized. Furthermore, phase "states" are not quite orthogonal. Phase "states" occupy a ray in phase space:



A ***number-phase code*** encodes a qubit (or qudit) into evenly-spaced sequences of phase states. For instance, for a qubit and $N = 3$ phase states per code word, we can set

$$\begin{aligned} |\overline{0}_{\text{num-ph}}\rangle &= \frac{1}{\sqrt{3}}\Big(|\phi{=}0\rangle + |\phi{=}\tfrac{2\pi}{3}\rangle + |\phi{=}\tfrac{4\pi}{3}\rangle\Big) \ ; \\ |\overline{1}_{\text{num-ph}}\rangle &= \frac{1}{\sqrt{3}}\Big(|\phi{=}\tfrac{\pi}{3}\rangle + |\phi{=}\pi\rangle + |\phi{=}\tfrac{5\pi}{3}\rangle\Big) \ . \end{aligned} \tag{9.32}$$

The number-phase code has a useful representation of its $|\overline{\pm}\rangle$ code words in terms of Fock

states:

$$\begin{aligned} |\overline{+}_{\text{num-ph}}\rangle &= \frac{1}{\sqrt{2}}\Big(|\overline{0}_{\text{num-ph}}\rangle + |\overline{1}_{\text{num-ph}}\rangle\Big) \quad \propto \quad \sum_{n\geq 0}|6n\rangle \ ; \\ |\overline{-}_{\text{num-ph}}\rangle &= \frac{1}{\sqrt{2}}\Big(|\overline{0}_{\text{num-ph}}\rangle - |\overline{1}_{\text{num-ph}}\rangle\Big) \quad \propto \quad \sum_{n\geq 0}|6n+3\rangle \ ; \end{aligned} \tag{9.33}$$

Intuitively, number-phase codes can correct small shifts in phase (but careful! $|\phi\rangle$'s are not orthogonal) and a small number of jumps in excitation levels ($|n\rangle \to |n \pm 1\rangle$).

Recall that a *coherent state* $|\alpha\rangle$ is defined as an eigenstate of the mode annihilation operator with complex eigenvalue $\alpha \in \mathbb{C}$:

$$\hat{a}|\alpha\rangle = \alpha|\alpha\rangle \ . \tag{9.34}$$

Recall some properties of coherent states:

- Fock basis representation:

$$|\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n\geq 0} \frac{\alpha^n}{\sqrt{n!}} |n\rangle \ ; \tag{9.35}$$

- As displaced vacuum:

$$|\alpha\rangle = \hat{D}(\alpha)\,|0\rangle \ ; \qquad \hat{D}(\alpha) = e^{\alpha\hat{a}^\dagger - \alpha^*\hat{a}} = \hat{D}\Big(\frac{\mathrm{Re}(\alpha)}{\sqrt{2}}, \frac{\mathrm{Im}(\alpha)}{\sqrt{2}}\Big) \ ; \tag{9.36}$$

- Time evolution with the Hamiltonian $H = \omega\hat{n} = \omega\hat{a}^\dagger\hat{a}$:

$$e^{-iHt}|\alpha\rangle = |\alpha\, e^{-i\omega t}\rangle \ ; \tag{9.37}$$

- The Wigner function of $|\alpha\rangle$ is a Gaussian centered at $(x = \mathrm{Re}(\alpha)\sqrt{2}, p = \mathrm{Im}(\alpha)\sqrt{2})$.
- Overlap between two coherent states:

$$|\langle\alpha\,|\beta\rangle| = e^{-\frac{1}{2}|\alpha-\beta|^2} \ . \tag{9.38}$$

If we use coherent states instead of the phase states in the number-phase code, we obtain a ***cat code***. Given α, n, a qubit is encoded as

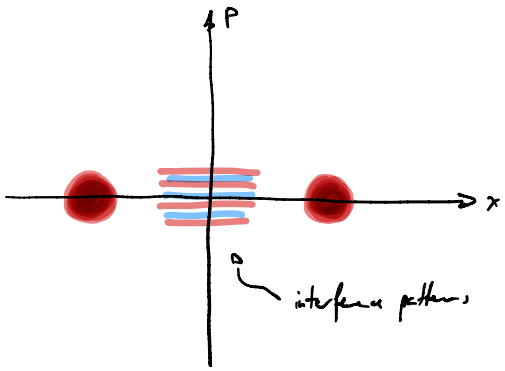
$$|\overline{j}_{\text{cat},\approx}\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |\alpha\, e^{i\frac{\pi}{N}(2k+j)}\rangle \ . \tag{9.39}$$

These code words are not exactly orthogonal, since coherent states are not orthogonal! It can be more convenient to define the cat code states in the Hadamard basis as

$$|\overline{\mp}_{\text{cat}}\rangle = \frac{1}{2}\Big(|\overline{0}_{\text{cat},\approx}\rangle \pm |\overline{1}_{\text{cat},\approx}\rangle\Big) \ , \tag{9.40}$$

now yielding truly orthogonal states.

In phase space ($N = 2$), cat states look as follows:



Cat codes can protect against dephasing errors, which we can check by expanding the dephasing error $e^{i\theta\hat{n}} \approx \mathbb{1} + i\theta\hat{n}$ and inspecting the Knill-Laflamme conditions:

$$\Pi\hat{n}\Pi = |\alpha|^2\Pi + O(|\alpha|^2 e^{-|\alpha|^2}) Z_{\text{cat}} \ , \tag{9.41}$$

where $Z_{\text{cat}} = |\overline{+}_{\text{cat}}\rangle\langle\overline{+}_{\text{cat}}| - |\overline{-}_{\text{cat}}\rangle\langle\overline{-}_{\text{cat}}|$ is the logical Z operator of the cat-code-encoded qubit.

Because the coherent states are not perfectly orthogonal, there is still an exponentially suppressed (in $|\alpha|^2$) term effecting a logical Z. But dephasing cannot produce a logical X error.

Cat code words can be engineered by dissipative systems with a Lindbladian of the form

$$\mathcal{L}[\rho] = J\rho J^\dagger - \frac{1}{2}\{J^\dagger J, \rho\} \ , \qquad\qquad J = \hat{a}^{2N} - \alpha^{2N} \ . \tag{9.42}$$

Steady states are then precisely those $|\psi\rangle$ for which $J|\psi\rangle = 0$, i.e., $(\hat{a}^{2N} - \alpha^{2N})|\psi\rangle = 0$, which single out the cat code words.

Another way of constructing a normalized (or physical) version of number-phase codes is to introduce nonuniform coefficients in the Fock state representation, and to make sure the coefficients vanish past some cut-off maximal excitation number.

A ***binomial code*** is defined by the following code words, for fixed N, D:

$$\begin{aligned} |\overline{0}_{\text{bin}}\rangle &= \frac{1}{2^{D/2}} \sum_{m \text{ even}} \sqrt{\binom{D+1}{m}} |Nm\rangle \ ; \\ |\overline{1}_{\text{bin}}\rangle &= \frac{1}{2^{D/2}} \sum_{m \text{ odd}} \sqrt{\binom{D+1}{m}} |Nm\rangle \ . \end{aligned} \tag{9.43}$$

A simple example is with $N = 2$ and $D = 1$:

$$|\overline{0}_{\text{bin}}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |4\rangle) \ ; \qquad\qquad |\overline{1}_{\text{bin}}\rangle = |2\rangle \ . \tag{9.44}$$

This code detects a Fock state shift (an excitation up or down, although not both), and a dephasing error $\hat{n}$. The code can thus correct *one loss* $\hat{a}$.

9.4 Further reading

- Victor's review paper [61]

- GKP original paper [62]

- Girvin lecture notes [12]

- General Gaussian quantum information [63]

- Noise in bosonic modes, channel capacities [64]

- other types of bosonic codes - e.g. spherical codes [65]

- Physical realization in cavity QED [66]

- GKP and symplectic lattices. [67] - Fault tolerance in GKP and GKP geometry [68]

Bibliography

[1] Daniel Gottesman, *Surviving as a Quantum Computer in a Classical World, 2024 Draft* (2024).

[2] Victor V Albert and Philippe Faist, *The Error Correction Zoo*.

[3] John Preskill, *Quantum Information and Quantum Computation. Lecture Notes, Caltech*.

[4] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information* (2010).

[5] Keisuke Fujii, Quantum computation with topological codes: from qubit to topological fault-tolerance (2015), arXiv:1504.01444.

[6] Daniel Gottesman, An introduction to quantum error correction and fault-tolerant quantum computation (2009), arXiv:0904.2557.

[7] Joschka Roffe, Quantum error correction: an introductory guide, Contemporary Physics **60**, 226–245 (2019), arXiv:1907.11157.

[8] Daniel Gottesman, Stabilizer codes and quantum error correction (1997), arXiv:quant-ph/9705052.

[9] Nikolas P. Breuckmann, PhD thesis: Homological quantum codes beyond the toric code (2018), arXiv:1802.01520.

[10] Aleksander Marek Kubica, *The ABCs of the Color Code: A Study of Topological Quantum Codes as Toy Models for Fault-Tolerant Quantum Computation and Quantum Phases Of Matter*, Ph.D. thesis, California Institute of Technology (2018).

[11] Tomas Jochym-O'Connor, *Novel Methods in Quantum Error Correction*, Ph.D. thesis, University of Waterloo (2016).

[12] Steven M. Girvin, Introduction to quantum error correction and fault tolerance, SciPost Physics Lecture Notes , 70 (2023), arXiv:2111.08894.

[13] Dan Browne, *Topological Codes and Computation. Lecture Notes, Univ. Innsbruck* (2014).

[14] Andrew Cleland, An introduction to the surface code, SciPost Physics Lecture Notes , 49 (2022).

[15] John Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018), arXiv:1801.00862.

[16] Frank Arute *et al.*, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505–510 (2019), arXiv:1910.11333.

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

[18] Ying Li and Simon C. Benjamin, Efficient variational quantum simulator incorporating active error minimization, Physical Review X **7**, 021050 (2017), arXiv:1611.09301.

[19] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta, Error mitigation for short-depth quantum circuits, Physical Review Letters **119**, 180509 (2017), arXiv:1612.02058.

[20] Christophe Piveteau, David Sutter, Sergey Bravyi, Jay M. Gambetta, and Kristan Temme, Error mitigation for universal gates on encoded qubits, Physical Review Letters **127**, 200505 (2021), arXiv:2103.04915.

[21] Yihui Quek, Daniel Stilck França, Sumeet Khatri, Johannes Jakob Meyer, and Jens Eisert, Exponentially tighter bounds on limitations of quantum error mitigation, Nature Physics **20**, 1648–1658 (2024), arXiv:2210.11505.

[22] Lorenza Viola and Seth Lloyd, Dynamical suppression of decoherence in two-state quantum systems, Physical Review A **58**, 2733–2744 (1998), arXiv:quant-ph/9803057.

[23] Harrison Ball *et al.*, Software tools for quantum control: improving quantum computer performance through noise and error suppression, Quantum Science and Technology **6**, 044011 (2021), arXiv:2001.04060.

[24] Emanuel Knill and Raymond Laflamme, Theory of quantum error-correcting codes, Physical Review A **55**, 900–911 (1997), arXiv:quant-ph/9604034.

[25] Benjamin Schumacher and M. A. Nielsen, Quantum data processing and error correction, Physical Review A **54**, 2629–2635 (1996), arXiv:quant-ph/9604022.

[26] Cédric Bény and Ognyan Oreshkov, General conditions for approximate quantum error correction and near-optimal recovery channels, Physical Review Letters **104**, 120501 (2010), arXiv:0907.5391.

[27] Patrick Hayden and Andreas Winter, Weak decoupling duality and quantum identification, IEEE Transactions on Information Theory **58**, 4914–4929 (2012), arXiv:1003.4994.

[28] Cédric Bény, Achim Kempf, and David W. Kribs, Quantum error correction on infinite-dimensional hilbert spaces, Journal of Mathematical Physics **50**, 10.1063/1.3155783 (2009), arXiv:0811.0421.

[29] Cédric Bény, Achim Kempf, and David W. Kribs, Quantum error correction of observables, Physical Review A **76**, 042303 (2007), arXiv:0705.1574.

[30] Sergey Bravyi and Barbara Terhal, A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes, New Journal of Physics **11**, 043029 (2009), arXiv:0810.1983.

[31] Joschka Roffe, David R. White, Simon Burton, and Earl Campbell, Decoding across the quantum low-density parity-check code landscape, Physical Review Research **2**, 043423 (2020), arXiv:2005.07016.

[32] Min-Hsiu Hsieh and François Le Gall, Np-hardness of decoding quantum error-correction codes, Physical Review A **83**, 052331 (2011), arXiv:1009.1319.

[33] Kao-Yueh Kuo and Chung-Chin Lu, On the hardness of decoding quantum stabilizer codes under the depolarizing channel, in *2012 International Symposium on Information Theory and its Applications* (2012) p. 208–211.

[34] Pavithran Iyer and David Poulin, Hardness of decoding quantum stabilizer codes (2013), arXiv:1310.3235.

[35] A. Yu. Kitaev, Quantum error correction with imperfect gates (1997).

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

[37] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill, Topological quantum memory, Journal of Mathematical Physics **43**, 4452–4505 (2002), arXiv:quant-ph/0110143.

[38] H. Bombin, An introduction to topological quantum codes (2013), arXiv:1311.0277.

[39] Arthur Pesah, *Interactive Introduction to the Surface Code. Blog post*.

[40] Sebastian Krinner *et al.*, Realizing repeated quantum error correction in a distance-three surface code, Nature **605**, 669–674 (2022), arXiv:2112.03708.

[41] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland, Surface codes: Towards practical large-scale quantum computation, Physical Review A **86**, 032324 (2012), arXiv:1208.0928.

[42] Daniel Litinski, A game of surface codes: Large-scale quantum computing with lattice surgery, Quantum **3**, 128 (2019), arXiv:1808.02892.

[43] Bei Zeng, Xie Chen, Duan-Lu Zhou, and Xiao-Gang Wen, *Quantum Information Meets Quantum Matter* (Springer New York, 2019) arXiv:1508.02595.

[44] Sergey Bravyi and Matthew B. Hastings, Homological product codes (2013), arXiv:1311.0885.

[45] Aleksander Kubica and Michael E. Beverland, Universal transversal gates with color codes: A simplified approach, Physical Review A **91**, 032330 (2015), arXiv:1410.0069.

[46] Markus S. Kesselring, Fernando Pastawski, Jens Eisert, and Benjamin J. Brown, The boundaries and twist defects of the color code and their applications to topological quantum computation, Quantum **2**, 101 (2018), arXiv:1806.02820.

[47] Markus S. Kesselring, Julio C. Magdalena de la Fuente, Felix Thomsen, Jens Eisert, Stephen D. Bartlett, and Benjamin J. Brown, Anyon condensation and the color code, PRX Quantum **5**, 010342 (2024), arXiv:2212.00042.

[48] Andrew J. Landahl, Jonas T. Anderson, and Patrick R. Rice, Fault-tolerant quantum computing with color codes (2011), arXiv:1108.5738.

[49] Aleksander Kubica, Michael E. Beverland, Fernando Brandão, John Preskill, and Krysta M. Svore, Three-dimensional color code thresholds via statistical-mechanical mapping, Physical Review Letters **120**, 180501 (2018), arXiv:1708.07131.

[50] Peter-Jan H. S. Derks, Alex Townsend-Teague, Ansgar G. Burchards, and Jens Eisert, Designing fault-tolerant circuits using detector error models (2024), arXiv:2407.13826.

[51] Rajeev Acharya *et al.*, Quantum error correction below the surface code threshold (2024), arXiv:2408.13687.

[52] A. Paetznick *et al.*, Demonstration of logical qubits and repeated error correction with better-than-physical error rates (2024), arXiv:2404.02280.

[53] Dolev Bluvstein *et al.*, Logical quantum processor based on reconfigurable atom arrays, Nature **626**, 58–65 (2023), arXiv:2312.03982.

[54] Bryan Eastin and Emanuel Knill, Restrictions on transversal encoded quantum gate sets, Physical Review Letters **102**, 110502 (2009), arXiv:0811.4262.

[55] Sergey Bravyi and Robert König, Classification of topologically protected gates for local stabilizer codes, Physical Review Letters **110**, 170503 (2013), arXiv:1206.1609.

[56] Philippe Faist, Sepehr Nezami, Victor V. Albert, Grant Salton, Fernando Pastawski, Patrick Hayden, and John Preskill, Continuous symmetries and approximate quantum error correction, Physical Review X **10**, 041018 (2020), arXiv:1902.07714.

[57] Dominic Horsman, Austin G Fowler, Simon Devitt, and Rodney Van Meter, Surface code quantum computing by lattice surgery, New Journal of Physics **14**, 123011 (2012), arXiv:1111.4022.

[58] Sergey Bravyi and Jeongwan Haah, Magic-state distillation with low overhead, Physical Review A **86**, 052329 (2012), arXiv:1209.2426.

[59] Daniel Litinski, Magic state distillation: Not as costly as you think, Quantum **3**, 205 (2019), arXiv:1905.06903.

[60] Tomas Jochym-O'Connor and Raymond Laflamme, Using concatenated quantum codes for universal fault-tolerant quantum gates, Physical Review Letters **112**, 010505 (2014), arXiv:1309.3310.

[61] Victor V. Albert, Bosonic coding: introduction and use cases (2022), arXiv:2211.05714.

[62] Daniel Gottesman, Alexei Kitaev, and John Preskill, Encoding a qubit in an oscillator, Physical Review A **64**, 012310 (2001), arXiv:quant-ph/0008040.

[63] Christian Weedbrook, Stefano Pirandola, Raúl García-Patrón, Nicolas J. Cerf, Timothy C. Ralph, Jeffrey H. Shapiro, and Seth Lloyd, Gaussian quantum information, Reviews of Modern Physics **84**, 621–669 (2012), arXiv:1110.3234.

[64] Kyungjoo Noh, Victor V. Albert, and Liang Jiang, Quantum capacity bounds of gaussian thermal loss channels and achievable rates with Gottesman-Kitaev-Preskill codes, IEEE Transactions on Information Theory **65**, 2563–2582 (2019), arXiv:1801.07271.

[65] Shubham P. Jain, Joseph T. Iosue, Alexander Barg, and Victor V. Albert, Quantum spherical codes, Nature Physics **20**, 1300–1305 (2024), arXiv:2302.11593.

[66] P. Campagne-Ibarcq *et al.*, Quantum error correction of a qubit encoded in grid states of an oscillator, Nature **584**, 368–372 (2020), arXiv:1907.12487.

[67] Jonathan Conrad, Jens Eisert, and Francesco Arzani, Gottesman-Kitaev-Preskill codes: A lattice perspective, Quantum **6**, 648 (2022), arXiv:2109.14645.

[68] Jonathan Conrad, Ansgar G. Burchards, and Steven T. Flammia, Lattices, gates, and curves: GKP codes as a Rosetta stone (2024), arXiv:2407.03270.