PHYS 7810

Quantum Error Correction

Andrew Lucas[1,*] and ChatGPT/Codex 5.6

[1]Department of Physics and Center for Theory of Quantum Matter,
University of Colorado, Boulder, CO 80309, USA

These are AI-generated transcriptions (with light human edits) of human-developed handwritten lecture notes into more readable LaTeX. These lecture notes do not contain references.

Contents

* andrew.j.lucas@colorado.edu

1 Quantum channels

1.1 How to encode classical information

Classical information is usually transmitted as a sequence of bits. The environment may corrupt some of those bits before the message reaches its destination: see Figure 1.1. The basic idea of error correction is to encode information redundantly before exposing it to noise.

Example 1.1: The simplest example is the three-bit repetition code:

$$0 \longmapsto 000, \qquad 1 \longmapsto 111. \tag{1.1}$$

Here the bit on the left is the *logical bit*, while the three bits on the right are the *physical bits*. A majority-vote decoder assigns

$$\begin{aligned} &000,\ 001,\ 010,\ 100 \longmapsto 0, \\ &111,\ 110,\ 101,\ 011 \longmapsto 1. \end{aligned} \tag{1.2}$$

Thus the desired logic is schematically

$$\text{decoder}(\text{noise}(\text{information})) = \text{information}\,. \tag{1.3}$$

Modern communication networks use more sophisticated versions of this idea (i.e. more sophisticated codes). We'll describe them later in Section **??**.

No code is perfect: some errors are decodable, while sufficiently large errors can change the logical information. For example, $0 \longmapsto 000 \longmapsto 001 \longmapsto 0$ is a decodable error, whereas $0 \longmapsto 000 \longmapsto 011 \longmapsto 1$ is an undecodable, or logical, error. For the three-bit repetition code, every single-bit flip is correctable, but two or more bit flips can make the majority vote return the wrong logical bit. In Section 6, the quality of a code will be quantified by the smallest physical error capable of producing a logical error.

1.2 Why encoding quantum information is harder

A logical qubit is a normalized superposition

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \qquad |\alpha|^2 + |\beta|^2 = 1. \tag{1.4}$$

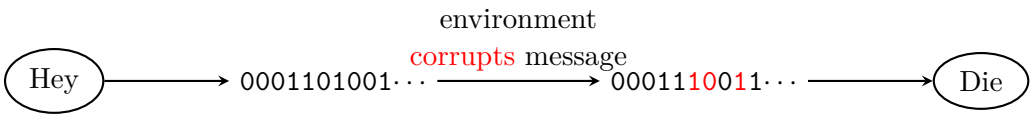

Figure 1.1: A classical "text message", which is first encoded into bits, can be altered by noise during transmission.

Unlike a classical bit, a qubit contains continuous parameters α and β. We must preserve the entire superposition, not merely distinguish the basis states $|0\rangle$ and $|1\rangle$.

1.2.1 Challenge 1: the no-cloning theorem

Theorem 1.2: There is no linear operation Q that clones every qubit according to

$$Q(\alpha|0\rangle + \beta|1\rangle) = (\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle) \tag{1.5}$$

for arbitrary α and β.

Proof. If Q clones the computational-basis states, then

$$Q|0\rangle = |0\rangle \otimes |0\rangle, \qquad Q|1\rangle = |1\rangle \otimes |1\rangle. \tag{1.6}$$

Linearity would therefore imply

$$Q(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle \otimes |0\rangle + \beta|1\rangle \otimes |1\rangle. \tag{1.7}$$

By contrast, the desired cloned state expands as

$$(\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle) = \alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle, \tag{1.8}$$

which is generically different from Eq. (1.7). Thus quantum redundancy cannot be created by simply copying an unknown state. □

1.2.2 Challenge 2: bit and phase errors

The Pauli operators act on the computational basis as

$$X|0\rangle = |1\rangle, \qquad X|1\rangle = |0\rangle, \qquad Z|0\rangle = |0\rangle, \qquad Z|1\rangle = -|1\rangle. \tag{1.9}$$

For three qubits, subscripts specify the qubit on which an operator acts; for example,

$$X_1 = X \otimes \mathbb{I} \otimes \mathbb{I}, \qquad Z_2 = \mathbb{I} \otimes Z \otimes \mathbb{I}. \tag{1.10}$$

Consider the encoded superposition $\alpha|000\rangle + \beta|111\rangle$. This is basically taking the three-bit repetition code and trying to make it quantum. A bit flip on the first qubit gives

$$X_1(\alpha|000\rangle + \beta|111\rangle) = \alpha|100\rangle + \beta|011\rangle, \tag{1.11}$$

which moves the state into a distinguishable error space. A phase flip gives

$$Z_1(\alpha|000\rangle + \beta|111\rangle) = \alpha|000\rangle - \beta|111\rangle, \tag{1.12}$$

which remains inside the same two-dimensional codespace and is therefore undetectable by this repetition code.

A measurement can also act as an error. If an eavesdropper measures Z_1, then

$$\alpha\,|000\rangle + \beta\,|111\rangle \xrightarrow{\text{measure } Z_1} \begin{cases} |000\rangle\,, & \text{with probability } |\alpha|^2, \\ |111\rangle\,, & \text{with probability } |\beta|^2. \end{cases} \tag{1.13}$$

The relative phase and coherent superposition have been destroyed.

1.3 Open quantum systems

To describe generic quantum error processes, we need the mathematical framework of open quantum systems. For a closed system, a pure state is represented by a vector. For a single qubit,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}. \tag{1.14}$$

An open system is generally described by a mixed state, or density matrix,

$$\rho = a\,|0\rangle\langle 0| + b\,|1\rangle\langle 1| + c\,|0\rangle\langle 1| + \overline{c}\,|1\rangle\langle 0| = \begin{pmatrix} a & c \\ \overline{c} & b \end{pmatrix}. \tag{1.15}$$

In these lectures, we use $\overline{c}$ to denote the complex conjugate of c. A valid density matrix satisfies

$$\rho = \rho^\dagger, \qquad \operatorname{spec}(\rho) \subseteq [0,1], \qquad \operatorname{tr}(\rho) = 1. \tag{1.16}$$

Since ρ is Hermitian, it admits a spectral decomposition

$$\rho = \sum_n p_n\,|\psi_n\rangle\langle\psi_n|\,, \qquad \langle\psi_n|\psi_{n'}\rangle = \delta_{nn'}. \tag{1.17}$$

We may interpret this as being in the pure state $|\psi_n\rangle$ with probability p_n. This decomposition is not unique if one does not insist that the states in the ensemble be orthogonal.

Equivalently, one often states the density-matrix condition as $\rho \succeq 0$ together with $\operatorname{tr}\rho = 1$. Positivity and unit trace automatically place every eigenvalue in the interval $[0,1]$.

1.4 Projective measurements

Suppose an observable A has eigenvalue a, and let P_a project onto the corresponding eigenspace, so that

$$AP_a = aP_a. \tag{1.18}$$

If outcome a is observed, Born's rule gives the conditional post-measurement state

$$\rho_a = \frac{P_a \rho P_a}{\mathrm{tr}(P_a \rho P_a)} \tag{1.19}$$

with probability

$$p_a = \mathrm{tr}(P_a \rho P_a). \tag{1.20}$$

The conditional update in Eq. (1.19) is nonlinear because of its normalization. If the outcome is measured and then discarded, however, the averaged state is

$$\rho_{\text{ignore}} = \sum_a p_a \rho_a = \sum_a \mathrm{tr}(P_a \rho P_a) \frac{P_a \rho P_a}{\mathrm{tr}(P_a \rho P_a)} = \sum_a P_a \rho P_a =: \mathcal{N}_A(\rho), \tag{1.21}$$

which is linear. A useful model of noise is therefore: the environment measures A, but we never learn the outcome.

Because $P_a^2 = P_a$ and the trace is cyclic, $p_a = \mathrm{tr}(P_a \rho P_a) = \mathrm{tr}(P_a \rho)$. The form in Eq. (1.20) makes the cancellation in Eq. (1.21) especially transparent.

Example 1.3: Start from

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \qquad \rho_0 = |\psi\rangle\langle\psi| = \begin{pmatrix} |\alpha|^2 & \alpha\overline{\beta} \\ \overline{\alpha}\beta & |\beta|^2 \end{pmatrix}. \tag{1.22}$$

If the environment measures Z and discards the outcome, then

$$\mathcal{N}_Z(\rho_0) = \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix}. \tag{1.23}$$

The off-diagonal coherences vanish, and the state is now mixed.

1.5 Quantum channels act on density matrices

A quantum channel is the most general physical – i.e. linear – operation that maps density matrices to density matrices:

$$\mathcal{C} : \{\text{density matrices}\} \longrightarrow \{\text{density matrices}\}. \tag{1.24}$$

It must satisfy three requirements.

1. It is trace preserving:

$$\mathrm{tr}(\mathcal{C}(\rho)) = \mathrm{tr}(\rho) = 1. \tag{1.25}$$

2. It is linear on probabilistic mixtures:

$$\mathcal{C}(p_1\rho_1 + p_2\rho_2) = p_1\mathcal{C}(\rho_1) + p_2\mathcal{C}(\rho_2). \tag{1.26}$$

This is required because a mixed state can represent classical uncertainty about which state was prepared. Also, although we introduced $\mathcal{C}$ as a map from density matrices to density matrices, linearity extends it to the full operator space. This extension is used below when $\mathcal{C}$ acts on operators such as $|\alpha\rangle\langle\alpha'|$ in (1.37).

3. It is completely positive. If the system S is entangled with an arbitrary reference system R, then

$$(\mathcal{C}_S \otimes \mathbb{I}_R)(\rho_{SR}) \succeq 0 \tag{1.27}$$

for every positive state ρ_{SR}.

The last requirement is essential because a system of interest may be entangled with degrees of freedom that are not explicitly included in its description.

Amazingly, all quantum channels (on finite-dimensional Hilbert spaces, which is basically what we'll be focusing on) can be completely classified by the following theorem.

Theorem 1.4 (Kraus representation)**:** In finite dimensions, $\mathcal{C}$ is completely positive and trace preserving if and only if it can be written as

$$\mathcal{C}(\rho) = \sum_i K_i \rho K_i^\dagger, \qquad \sum_i K_i^\dagger K_i = \mathbb{I}. \tag{1.28}$$

The operators K_i are called Kraus operators.

Proof. First suppose Eq. (1.28) holds. Trace preservation follows from cyclicity of the trace:

$$\mathrm{tr}(\mathcal{C}(\rho)) = \mathrm{tr}\left(\rho \sum_i K_i^\dagger K_i\right) = \mathrm{tr}(\rho\mathbb{I}) = 1. \tag{1.29}$$

For any positive ρ_{SR} and any vector $|\Psi_{SR}\rangle$,

$$\begin{aligned} \langle\Psi_{SR}|\,(\mathcal{C}_S \otimes \mathbb{I}_R)(\rho_{SR})\,|\Psi_{SR}\rangle &= \sum_i \langle\Psi_{SR}|\,(K_i \otimes \mathbb{I}_R)\rho_{SR}(K_i \otimes \mathbb{I}_R)^\dagger\,|\Psi_{SR}\rangle \\ &= \sum_i \langle\Psi_i|\,\rho_{SR}\,|\Psi_i\rangle \geq 0, \qquad |\Psi_i\rangle := (K_i^\dagger \otimes \mathbb{I}_R)\,|\Psi_{SR}\rangle\,. \end{aligned} \tag{1.30}$$

Thus $\mathcal{C}$ is completely positive.

Conversely, suppose $\mathcal{C}$ is CPTP. Take $\dim R = \dim S$ and choose bases $\{|\alpha\rangle_S\}$ and $\{|\alpha\rangle_R\}$.

Define the Bell-like vector

$$|\Omega\rangle = \sum_{\alpha} |\alpha\rangle_S \otimes |\alpha\rangle_R \,, \qquad \rho_\Omega = |\Omega\rangle\langle\Omega| \,. \tag{1.31}$$

Complete positivity implies that

$$\sigma_{SR} := (\mathcal{C}_S \otimes \mathbb{I}_R)(\rho_\Omega) \succeq 0. \tag{1.32}$$

Therefore the positive operator σ_{SR} can be decomposed as

$$\sigma_{SR} = \sum_i |\sigma_i\rangle\langle\sigma_i| \,. \tag{1.33}$$

Expanding ρ_Ω in the chosen basis gives

$$\sigma_{SR} = \sum_{\alpha,\alpha'} \mathcal{C}(|\alpha\rangle\langle\alpha'|) \otimes |\alpha\rangle\langle\alpha'| \,. \tag{1.34}$$

For an arbitrary $|\psi\rangle = \sum_\alpha \psi_\alpha |\alpha\rangle$, define its basis-wise conjugate in R by

$$\left|\overline{\psi}\right\rangle_R := \sum_\alpha \overline{\psi_\alpha} |\alpha\rangle_R \,, \tag{1.35}$$

and define

$$|k_i(\psi)\rangle_S := (\mathbb{I}_S \otimes \left\langle\overline{\psi}\right|_R) |\sigma_i\rangle_{SR} \,. \tag{1.36}$$

Then Eqs. (1.33) and (1.34) imply

$$\sum_i |k_i(\psi)\rangle\langle k_i(\psi)| = (\mathbb{I}_S \otimes \left\langle\overline{\psi}\right|_R)\sigma_{SR}(\mathbb{I}_S \otimes \left|\overline{\psi}\right\rangle_R) = \sum_{\alpha,\alpha'} \psi_\alpha \overline{\psi_{\alpha'}} \mathcal{C}(|\alpha\rangle\langle\alpha'|) = \mathcal{C}(|\psi\rangle\langle\psi|). \tag{1.37}$$

The conjugation in Eq. (1.35) makes $|k_i(\psi)\rangle$ a linear function of $|\psi\rangle$. Hence there is an operator K_i such that

$$|k_i(\psi)\rangle = K_i |\psi\rangle \,. \tag{1.38}$$

Equation (1.37) therefore yields

$$\mathcal{C}(|\psi\rangle\langle\psi|) = \sum_i K_i |\psi\rangle\langle\psi| K_i^\dagger. \tag{1.39}$$

By linearity, this extends to every density matrix:

$$\mathcal{C}(\rho) = \sum_i K_i \rho K_i^\dagger. \tag{1.40}$$

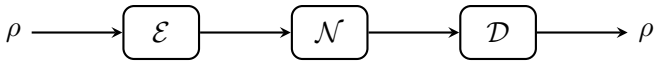

Figure 1.2: The encoder, noise, and decoder are all quantum channels. Remember that the order of operations in the quantum channels in (1.43) is read from right to left!

Finally, trace preservation implies

$$\sum_i K_i^\dagger K_i = \mathbb{I}, \tag{1.41}$$

as in the forward direction. □

The operator σ_{SR} in Eq. (1.32) is the Choi matrix of the channel, up to the normalization convention for $|\Omega\rangle$. The proof is constructive: a decomposition of the Choi matrix into rank-one positive terms directly produces a set of Kraus operators.

1.6 Looking forward: the goal of quantum error correction

Let ρ denote the logical quantum data. An encoder $\mathcal{E}$ maps it into a larger physical Hilbert space, a noise channel $\mathcal{N}$ acts, and a decoder $\mathcal{D}$ attempts to recover the logical state. The goal is

$$\mathcal{D}(\mathcal{N}(\mathcal{E}(\rho))) = \rho. \tag{1.42}$$

Equivalently, for every noise channel in the chosen correctable family,

$$\mathcal{D} \circ \mathcal{N} \circ \mathcal{E} = \mathcal{I} \tag{1.43}$$

on the logical state space: pictorially, see Figure 1.2.

The essence of QEC can be foreshadowed in three steps.

1. **Encode logical information in a subspace.** For the three-qubit repetition code, the encoder $\mathcal{E}$ acts as

$$\mathcal{E} : \alpha\,|0\rangle + \beta\,|1\rangle \longmapsto \alpha\,|000\rangle + \beta\,|111\rangle\,, \tag{1.44}$$

so the codespace is

$$\mathcal{H}_{\text{code}} = \text{span}\{|000\rangle\,, |111\rangle\}. \tag{1.45}$$

Single bit flips move the two logical basis states into distinct basis states:

$$\begin{aligned} |000\rangle &\xrightarrow{X_1,X_2,X_3} \{|100\rangle\,, |010\rangle\,, |001\rangle\}, \\ |111\rangle &\xrightarrow{X_1,X_2,X_3} \{|011\rangle\,, |101\rangle\,, |110\rangle\}. \end{aligned} \tag{1.46}$$

Entanglement is needed in order to preserve a generic superposition while distributing the logical information among several physical qubits. For generic nonzero α and β, the encoded state in Eq. (1.44) is entangled. The three-qubit repetition code nevertheless corrects an arbitrary single-qubit X error but no Z error; later codes combine complementary checks so that both kinds of error become detectable.

2. **Correctable noise $\mathcal{N}$ moves the state into distinguishable error spaces.** The error diagnosis should identify the occupied error space without revealing the logical state.

3. **Decode.** The decoder $\mathcal{D}$ infers the error and reverses it, returning the state to the codespace and recovering the logical data.

Problems

Problem 1.1 (Photon loss error)**:** Consider an atomic qubit where $|0\rangle$ is a ground state, while $|1\rangle$ is an excited state. These states are chosen such that $|1\rangle$ may decay via spontaneous emission of a photon to $|0\rangle$. A "microscopic" account of this decay process, using quantum channels, proceeds as follows. Let $|\circ\rangle$ denote the no-photon state while $|\gamma\rangle$ denotes the photon state. Assuming that there is no photon to start with, the state of the system after some time becomes:

$$U\left[\alpha|0\circ\rangle + \beta|1\circ\rangle\right] = \alpha|0\circ\rangle + \beta\left[\sqrt{1-p}|1\circ\rangle + \sqrt{p}|0\gamma\rangle\right] \tag{1.47}$$

where p is the probability of a decay event, and α, β are generic wave function coefficients. We will prefer to work in the language of open quantum systems, where we trace out the photon states. Show by explicit calculation that after tracing out the photon,

$$\operatorname*{tr}_{\text{photon}}\left[U\rho\otimes|\circ\rangle\langle\circ|U^\dagger\right] =: \mathcal{N}(\rho) =: K_1\rho K_1^\dagger + K_2\rho K_2^\dagger \tag{1.48}$$

where $\rho = |\psi\rangle\langle\psi|$, with $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, is the initially pure state of the atomic qubit alone, and the Kraus operators are

$$K_1 = |0\rangle\langle 0| + \sqrt{1-p}|1\rangle\langle 1|, \tag{1.49a}$$

$$K_2 = \sqrt{p}|0\rangle\langle 1|. \tag{1.49b}$$

Verify that $\mathcal{N}$ is a valid quantum channel.

2 Shor code

Recall that the goal of quantum error correction is

$$(\mathcal{D} \circ \mathcal{N} \circ \mathcal{E})(\rho) = \rho, \tag{2.1}$$

where $\mathcal{E}$ encodes the logical qubit, $\mathcal{N}$ describes the noise or error, and $\mathcal{D}$ decodes the state. It is often useful to work entirely within the physical Hilbert space. If

$$\rho_L = \mathcal{E}(\rho) \in \mathcal{C} \tag{2.2}$$

is an encoded state in the codespace, then a recovery channel $\mathcal{R}$ should obey

$$(\mathcal{R} \circ \mathcal{N})(\rho_L) = \rho_L. \tag{2.3}$$

We will study the simplest quantum code that detects and corrects single-qubit Pauli X- and Z-type errors.

2.1 The three-qubit repetition code for X errors

For three physical qubits, the Hilbert space is

$$\mathcal{H} = (\mathbb{C}^2)^{\otimes 3} = \text{span}\{|000\rangle, |001\rangle, \ldots, |111\rangle\}. \tag{2.4}$$

2.1.1 The codespace and error syndromes

The repetition-code subspace is

$$\mathcal{C} = \text{span}\{|000\rangle, |111\rangle\}, \qquad |0_L\rangle := |000\rangle, \qquad |1_L\rangle := |111\rangle. \tag{2.5}$$

Thus a generic encoded qubit is

$$|\psi_L\rangle = \alpha |000\rangle + \beta |111\rangle. \tag{2.6}$$

A single Pauli X error acts as

$$X_1 |\psi_L\rangle = \alpha |100\rangle + \beta |011\rangle \in \mathcal{B}_1, \tag{2.7a}$$
$$X_2 |\psi_L\rangle = \alpha |010\rangle + \beta |101\rangle \in \mathcal{B}_2, \tag{2.7b}$$
$$X_3 |\psi_L\rangle = \alpha |001\rangle + \beta |110\rangle \in \mathcal{B}_3. \tag{2.7c}$$

The corresponding orthogonal sectors are

$$\mathcal{B}_1 = \text{span}\{|100\rangle, |011\rangle\}, \tag{2.8a}$$

$$\mathcal{B}_2 = \mathrm{span}\{|010\rangle, |101\rangle\}, \tag{2.8b}$$
$$\mathcal{B}_3 = \mathrm{span}\{|001\rangle, |110\rangle\}, \tag{2.8c}$$

and therefore

$$\mathcal{H} = \mathcal{C} \oplus \mathcal{B}_1 \oplus \mathcal{B}_2 \oplus \mathcal{B}_3. \tag{2.9}$$

The decoding strategy is to determine which of these four sectors contains the state and then apply X_j if the state lies in $\mathcal{B}_j$. Crucially, measuring the sector does not determine α or β.

Let $P_{\mathcal{C}}$ denote the projector onto the codespace. On computational-basis states,

$$P_{\mathcal{C}} |x_1 x_2 x_3\rangle = \begin{cases} |x_1 x_2 x_3\rangle, & x_1 = x_2 = x_3, \\ 0, & \text{otherwise.} \end{cases} \tag{2.10}$$

Let $P_{\mathcal{B}_j}$ be the projectors onto the three error sectors, defined analogously. The recovery channel is

$$\mathcal{R}(\rho) = P_{\mathcal{C}} \rho P_{\mathcal{C}} + \sum_{j=1}^{3} X_j P_{\mathcal{B}_j} \rho P_{\mathcal{B}_j} X_j. \tag{2.11}$$

This is a valid quantum channel. Indeed, it has Kraus operators

$$K_0 = P_{\mathcal{C}}, \qquad K_j = X_j P_{\mathcal{B}_j} \quad (j = 1, 2, 3), \qquad \mathcal{R}(\rho) = \sum_{j=0}^{3} K_j \rho K_j^\dagger, \tag{2.12}$$

and they satisfy

$$\sum_{j=0}^{3} K_j^\dagger K_j = P_{\mathcal{C}}^2 + \sum_{j=1}^{3} P_{\mathcal{B}_j} X_j^2 P_{\mathcal{B}_j} = P_{\mathcal{C}} + \sum_{j=1}^{3} P_{\mathcal{B}_j} = \mathbb{I}. \tag{2.13}$$

2.1.2 Stabilizer measurements and the error syndrome

Definition 2.1: A *stabilizer* is an observable S that obeys $S|\psi\rangle = |\psi\rangle$ for every $|\psi\rangle \in \mathcal{C}$. We will usually specify a small set of independent stabilizer generators. We will revisit this concept more seriously in Section 7.

For the three-qubit repetition code, take

$$S_1 = Z_1 Z_2, \qquad S_2 = Z_2 Z_3. \tag{2.14}$$

Both logical basis states have eigenvalue $+1$ under both operators:

$$\begin{aligned} S_1 |000\rangle &= (+1)^2 |000\rangle = |000\rangle, & S_2 |000\rangle &= |000\rangle, \\ S_1 |111\rangle &= (-1)^2 |111\rangle = |111\rangle, & S_2 |111\rangle &= |111\rangle. \end{aligned} \tag{2.15}$$

Checking the remaining computational-basis states gives

sector	S_1	S_2	correction
$\mathcal{C}$	$+1$	$+1$	$\mathbb{I}$
$\mathcal{B}_1$	-1	$+1$	X_1
$\mathcal{B}_2$	-1	-1	X_2
$\mathcal{B}_3$	$+1$	-1	X_3

. (2.16)

The two stabilizer outcomes identify the error sector without resolving the logical state. Their ordered pair is called the *syndrome*. In particular, the codespace projector can be written as

$$P_{\mathcal{C}} = \frac{\mathbb{I} + S_1}{2} \frac{\mathbb{I} + S_2}{2}. \tag{2.17}$$

Thus measuring the stabilizers implements the sector measurement needed by the recovery channel.

2.1.3 Logical operators and detectable errors

Logical Pauli operators may be chosen as

$$X_L = X_1 X_2 X_3, \qquad Z_L = Z_1. \tag{2.18}$$

Their action on the encoded state is

$$X_L \left|\psi_L\right\rangle = \beta \left|000\right\rangle + \alpha \left|111\right\rangle, \tag{2.19a}$$

$$Z_L \left|\psi_L\right\rangle = \alpha \left|000\right\rangle - \beta \left|111\right\rangle. \tag{2.19b}$$

The first swaps $|0_L\rangle$ and $|1_L\rangle$, while the second applies a phase to $|1_L\rangle$. We must not measure X_L or Z_L, because doing so would measure and generally destroy the encoded quantum information.

The key property is that the logical operators commute with the stabilizers:

$$[X_L, S_j] = [Z_L, S_j] = 0, \qquad j = 1, 2. \tag{2.20}$$

For example,

$$X_L S_1 = X_1 X_2 X_3 Z_1 Z_2 = (XZ) \otimes (XZ) \otimes X = (-ZX) \otimes (-ZX) \otimes X = S_1 X_L. \tag{2.21}$$

Here we used the single-qubit anticommutation relation

$$XZ = -ZX. \tag{2.22}$$

It can be checked directly on the computational basis:

$$\begin{aligned} XZ\,|0\rangle &= X\,|0\rangle = |1\rangle\,, & ZX\,|0\rangle &= Z\,|1\rangle = -\,|1\rangle\,, \\ XZ\,|1\rangle &= -X\,|1\rangle = -\,|0\rangle\,, & ZX\,|1\rangle &= Z\,|0\rangle = |0\rangle\,. \end{aligned} \tag{2.23}$$

An error need not commute with the logical operators. For example,

$$[X_1, Z_L] \neq 0. \tag{2.24}$$

It is detectable because it fails to commute with at least one stabilizer.

Proposition 2.2: Suppose an error E obeys

$$ES_j = \eta_j S_j E, \qquad \eta_j \in \{+1, -1\}. \tag{2.25}$$

Then, on any codeword, measurement of S_j returns the deterministic outcome η_j.

Proof. For $|\psi\rangle \in \mathcal{C}$, we have $S_j\,|\psi\rangle = |\psi\rangle$. Therefore

$$S_j E\,|\psi\rangle = \eta_j E S_j\,|\psi\rangle = \eta_j E\,|\psi\rangle\,. \tag{2.26}$$

Thus the errored state is an eigenstate of S_j with eigenvalue η_j. □

For example,

$$X_1 S_1 = -S_1 X_1, \qquad X_1 S_2 = S_2 X_1, \tag{2.27}$$

so an X_1 error has syndrome $(-1, +1)$. This matches what we saw earlier in (2.16). Linearity makes the result valid for every superposition in the codespace.

2.2 The three-qubit repetition code for Z errors

A complementary repetition code detects phase errors rather than bit flips. Its stabilizers and logical operators are

$$S_1 = X_1 X_2, \qquad S_2 = X_2 X_3, \qquad X_L = X_1, \qquad Z_L = Z_1 Z_2 Z_3. \tag{2.28}$$

The roles of X and Z can be exchanged by the Hadamard gate,

$$H = \frac{X + Z}{\sqrt{2}}, \qquad H^\dagger X H = Z, \qquad H^\dagger Z H = X. \tag{2.29}$$

Thus applying $H^{\otimes 3}$ maps the bit-flip repetition code to the phase-flip repetition code. We choose the logical basis

$$|0_L\rangle = \frac{1}{2}(|000\rangle + |011\rangle + |101\rangle + |110\rangle), \tag{2.30a}$$

$$|1_L\rangle = \frac{1}{2}(|100\rangle + |010\rangle + |001\rangle + |111\rangle). \tag{2.30b}$$

This convention also includes a logical Hadamard. In particular,

$$H^{\otimes 3}|000\rangle = \frac{|0_L\rangle + |1_L\rangle}{\sqrt{2}}. \tag{2.31}$$

For example, the stabilizer and logical actions are

$$S_1|0_L\rangle = \frac{1}{2}(|110\rangle + |101\rangle + |011\rangle + |000\rangle) = |0_L\rangle, \tag{2.32a}$$

$$X_L|0_L\rangle = \frac{1}{2}(|100\rangle + |111\rangle + |001\rangle + |010\rangle) = |1_L\rangle. \tag{2.32b}$$

A phase error produces a detectable pattern of relative signs:

$$Z_1|0_L\rangle = \frac{1}{2}(|000\rangle + |011\rangle - |101\rangle - |110\rangle). \tag{2.33}$$

Indeed, the errored state is orthogonal to the original codeword:

$$\langle 0_L| Z_1 |0_L\rangle = \frac{1}{4}(1 + 1 - 1 - 1) = 0. \tag{2.34}$$

Equivalently, the anticommutation of Z_1 with S_1 gives a nontrivial syndrome.

2.3 Code concatenation and the nine-qubit Shor code

We now combine the two repetition codes to obtain a genuinely quantum code that protects against both X- and Z-type errors. Begin with the phase-flip codeword written as three outer qubits,

$$|0_L\rangle_{\text{outer}} = \frac{1}{2}(|0\rangle\otimes|0\rangle\otimes|0\rangle + |0\rangle\otimes|1\rangle\otimes|1\rangle + |1\rangle\otimes|0\rangle\otimes|1\rangle + |1\rangle\otimes|1\rangle\otimes|0\rangle). \tag{2.35}$$

Replace each outer qubit by a logical qubit of the bit-flip repetition code:

$$|0\rangle \longmapsto |000\rangle, \qquad |1\rangle \longmapsto |111\rangle. \tag{2.36}$$

This is described more formally in Section 17.3; here we will focus on a more intuitive and direct description. The result is the nine-qubit Shor code,

$$|0_L\rangle = \frac{1}{2}(|000\,000\,000\rangle + |000\,111\,111\rangle + |111\,000\,111\rangle + |111\,111\,000\rangle), \tag{2.37a}$$

$$|1_L\rangle = \frac{1}{2}(|111\,000\,000\rangle + |000\,111\,000\rangle + |000\,000\,111\rangle + |111\,111\,111\rangle). \tag{2.37b}$$

Writing out the codewords becomes cumbersome very quickly. The stabilizer description is much more economical.

2.3.1 Stabilizers of the Shor code

The two outer phase-flip stabilizers become products of logical X operators on neighboring three-qubit blocks, while each inner block retains its two Z-type stabilizers. A set of eight stabilizer generators is

$$\begin{aligned} S_1 &= X_1X_2X_3X_4X_5X_6, & S_2 &= X_4X_5X_6X_7X_8X_9, \\ S_3 &= Z_1Z_2, & S_4 &= Z_2Z_3, \\ S_5 &= Z_4Z_5, & S_6 &= Z_5Z_6, \\ S_7 &= Z_7Z_8, & S_8 &= Z_8Z_9. \end{aligned} \tag{2.38}$$

The six generators $S_3, \ldots, S_8$ stabilize the three inner bit-flip-code blocks—qubits 1–3, 4–6, and 7–9—and ensure that each block is an encoded inner qubit, with basis states $|000\rangle$ and $|111\rangle$. The other two generators come from the two X-type stabilizers of the outer phase-flip code. After each outer qubit is replaced by a three-qubit block, its X is replaced by the logical X of that inner block, namely XXX; applying the first outer stabilizer to the first two blocks ($X \otimes X \otimes \mathbb{I}$ as seen by the outer code, following the tensor product notation in (2.35)) gives S_1, while applying the second to the last two blocks ($\mathbb{I} \otimes X \otimes X$) gives S_2.

The logical operators transform to

$$X_L = X_1X_2X_3, \qquad Z_L = Z_1Z_4Z_7. \tag{2.39}$$

They obey

$$[X_L, S_j] = [Z_L, S_j] = 0, \qquad [S_j, S_{j'}] = 0, \qquad j, j' = 1, \ldots, 8. \tag{2.40}$$

For every pair consisting of an X-type generator and a Z-type generator, the supports overlap on either zero or two qubits. Each overlap contributes one minus sign when the operators are exchanged, so the total sign is always $+1$. This is a general property we return to in Section 7.

Because all eight stabilizers commute, they can be measured simultaneously to diagnose errors without measuring the logical operators. Again, we'll revisit this more carefully in Section 7.

2.3.2 Correcting bit-flip errors

To detect X errors, measure the six inner-block stabilizers

$$Z_1Z_2, \quad Z_2Z_3, \quad Z_4Z_5, \quad Z_5Z_6, \quad Z_7Z_8, \quad Z_8Z_9. \tag{2.41}$$

Within each block, the syndrome is exactly the one in Eq. (2.16); hence one can detect and correct at most one X error in each block. For example, the computational-basis state $|001\,000\,010\rangle$ has

$$\begin{aligned}(Z_2Z_3)\,|001\,000\,010\rangle &= -\,|001\,000\,010\rangle\,,\\ (Z_7Z_8)\,|001\,000\,010\rangle &= -\,|001\,000\,010\rangle\,,\\ (Z_8Z_9)\,|001\,000\,010\rangle &= -\,|001\,000\,010\rangle\,.\end{aligned} \tag{2.42}$$

All other Z-stabilizers have $+1$ eigenvalues in this state. The first syndrome locates the flip on qubit 3, while the last two locate the flip on qubit 8.

(2.42) is not analyzing the syndrome on a codeword; it just illustrates how the strategy from Section 2.1 would generalize. We should really think of detecting the error $E = X_3X_8$ because $\{E, S_4\} = \{E, S_7\} = \{E, S_8\} = 0$, following Proposition 2.2.

2.3.3 Correcting phase-flip errors

To detect a Z error, measure the two outer stabilizers S_1 and S_2. For a phase error in the first block,

$$Z_jS_1 = -S_1Z_j, \qquad Z_jS_2 = S_2Z_j, \qquad j = 1, 2, 3. \tag{2.43}$$

The syndrome identifies the block but not the particular qubit within that block. This is sufficient because the three errors act identically on the codespace. For example, if $|\psi\rangle \in \mathcal{C}$, then

$$Z_1\,|\psi\rangle = Z_1(Z_1Z_2)\,|\psi\rangle = Z_2\,|\psi\rangle\,. \tag{2.44}$$

The same reasoning gives

$$Z_1\,|\psi\rangle = Z_2\,|\psi\rangle = Z_3\,|\psi\rangle\,, \qquad Z_4\,|\psi\rangle = Z_5\,|\psi\rangle = Z_6\,|\psi\rangle\,, \qquad Z_7\,|\psi\rangle = Z_8\,|\psi\rangle = Z_9\,|\psi\rangle\,. \tag{2.45}$$

More generally, if S stabilizes the codespace, then E and ES have the same action on every codeword. Errors related in this way are called *stabilizer-equivalent*; a syndrome need only identify their equivalence class. It is enough to use Z_1, Z_4, and Z_7 as the three representative phase-error corrections. Their syndromes are

representative error	Z_1	Z_4	Z_7
(S_1, S_2)	$(-1, +1)$	$(-1, -1)$	$(+1, -1)$

. (2.46)

2.3.4 Combined bit-flip and phase-flip errors

Finally,

$$Y_j = -\mathrm{i}Z_jX_j. \tag{2.47}$$

A Y_j error therefore produces both the X_j-type and Z_j-type syndrome information. Detecting and correcting those two components separately corrects the Y_j error as well.

Problems

Problem 2.1: Suppose that in the Shor code we measure error syndrome

$$(S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8) = (+1, -1, -1, +1, +1, +1, -1, +1). \tag{2.48}$$

Provide three different candidate unitary operations we could apply to a quantum state to return it to the codespace. Why is it unimportant which choice we apply?

Problem 2.2: Replace the repetition code from Section 2.1 with the following 5-qubit bit-flip repetition code, where the codespace is $\text{span}(|00000\rangle, |11111\rangle)$.

A: Write down a set of 4 stabilizers together with logical operators. It suffices to guess the right structure here, following Section 2.1, and check that it works.

B: Write down the stabilizers for a 5-qubit phase-flip repetition code.

C: Build a concatenated quantum code to correct against both phase-flip and bit-flip errors, generalizing the strategy from Section 2.3. How many physical qubits does it have? How many logical qubits? Write down stabilizers and logical operators.

D: Can you think of any advantages of this code, compared to the Shor code (Section 2.3)?

3 Knill-Laflamme conditions

We have seen a specific example of a quantum code. We now develop a general, abstract theory of quantum error correction. The abstraction is worthwhile: relatively simple conditions will guarantee protection against all quantum errors in a prescribed error space.

3.1 Abstract setup

Let $\mathcal{H}$ be a discrete Hilbert space, let $\mathcal{C} \subseteq \mathcal{H}$ be the codespace that we wish to protect, and let P denote the orthogonal projector onto $\mathcal{C}$. Thus a state supported in the codespace obeys

$$\rho = P\rho P. \tag{3.1}$$

The goal is to find a recovery channel $\mathcal{R}$ such that

$$(\mathcal{R} \circ \mathcal{N})(\rho) = \rho \tag{3.2}$$

for every channel of the form

$$\mathcal{N}(\rho) = \sum_i K_i \rho K_i^\dagger, \qquad K_i \in \text{span}\{E_a\}. \tag{3.3}$$

The important point is that the Kraus operators need not themselves be chosen from a discrete list. For example, the error space may contain both X_j and Z_ℓ, and hence also every linear combination (e.g. $\alpha X_j + \beta Z_\ell$).

Theorem 3.1 (Knill–Laflamme conditions)**:** Let $\{E_a\}$ be a basis for a linear space of errors. This error space is correctable on $\mathcal{C}$ if and only if there is a Hermitian matrix α such that

$$PE_a^\dagger E_b P = \alpha_{ab} P \qquad \text{for every } a, b. \tag{3.4}$$

In particular,

$$\alpha_{ab} = \overline{\alpha_{ba}}. \tag{3.5}$$

Example 3.2: As a simple sanity check, let's see how this works for the 3-qubit bit-flip repetition code (Section 2.1). Here we claim that the set of correctable errors is $\{\mathbb{I}, X_1, X_2, X_3\}$. To see why this is a set of correctable errors, let's do a few quick calculations: using that

$$P = \frac{\mathbb{I}+S_1}{2}\frac{\mathbb{I}+S_2}{2} = \frac{\mathbb{I}+Z_1Z_2}{2}\frac{\mathbb{I}+Z_2Z_3}{2}, \tag{3.6}$$

we can check (3.4) for a few different errors in the allowed error set:

$$PX_1^\dagger \mathbb{I} P = X_1 \frac{\mathbb{I}-S_1}{2}\frac{\mathbb{I}+S_2}{2}\cdot\frac{\mathbb{I}+S_1}{2}\frac{\mathbb{I}+S_2}{2} = 0, \tag{3.7a}$$

$$PX_1^\dagger X_2 P = X_1 \frac{\mathbb{I}-S_1}{2}\frac{\mathbb{I}+S_2}{2}\cdot\frac{\mathbb{I}-S_1}{2}\frac{\mathbb{I}-S_2}{2}X_2 = 0, \tag{3.7b}$$

and so on. Notice that the stabilizer technology is very useful, again! Theorem 3.1 shows us that we can correct against any superposition of such errors, such as $\cos\theta X_1 + \sin\theta X_2$. This is far more powerful than what we were first thinking when we designed this code!

The rest of the lecture proves Theorem 3.1 and pauses along the way to explain its physical content. First, we may change the basis of the error space by diagonalizing α, and then rescale every basis vector corresponding to a nonzero eigenvalue. We therefore split the resulting basis into operators $E_{\widetilde{a}}$ and $E_{\bar{a}}$ such that

$$PE_{\widetilde{a}}^\dagger E_{\widetilde{b}} P = \delta_{\widetilde{a}\widetilde{b}} P, \tag{3.8a}$$

$$PE_{\bar{a}}^\dagger E_b P = 0 \qquad \text{for every } b. \tag{3.8b}$$

The tilded indices label the nonzero block of α, while the barred indices label its zero block. Some barred errors may annihilate the codespace; we return to them below.

The rescaling above is well defined because α is automatically positive semidefinite. Indeed, for $F = \sum_a v_a E_a$, the Knill–Laflamme condition gives

$$PF^\dagger F P = \left(\sum_{a,b} \overline{v_a}\alpha_{ab} v_b\right) P \geq 0. \tag{3.9}$$

Thus the nonzero eigenvalues of α are positive and can be normalized to one.

3.2 Sufficiency: constructing a recovery channel

The first part of the proof of Theorem 3.1 shows that (3.4) is sufficient to construct $\mathcal{R}$.

3.2.1 Orthogonal error sectors

Lemma 3.3: For each normalized error $E_{\widetilde{a}}$, there is a unitary $U_{\widetilde{a}}$ such that

$$E_{\widetilde{a}} P = U_{\widetilde{a}} P. \tag{3.10}$$

If

$$P_{\widetilde{a}} := U_{\widetilde{a}} P U_{\widetilde{a}}^\dagger, \tag{3.11}$$

then the rotated codespaces are mutually orthogonal:

$$P_{\widetilde{b}} P_{\widetilde{a}} = \delta_{\widetilde{a}\widetilde{b}} P_{\widetilde{a}}. \tag{3.12}$$

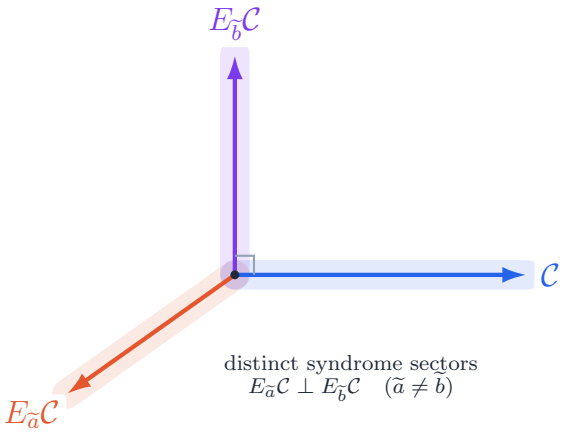

Figure 3.1: Each distinct detectable error $E_{\tilde{a}}$ takes the codespace $\mathcal{C}$ into an orthogonal subspace.

Proof. For $|\psi\rangle, |\phi\rangle \in \mathcal{C}$, Eq. (3.8a) implies

$$\langle\psi|\phi\rangle = \langle\psi| P |\phi\rangle = \langle\psi| P E_{\tilde{a}}^\dagger E_{\tilde{a}} P |\phi\rangle = \langle E_{\tilde{a}}\psi | E_{\tilde{a}}\phi\rangle . \tag{3.13}$$

Hence $E_{\tilde{a}}P$ is an isometry on the codespace. Extending this isometry to the rest of the finite-dimensional Hilbert space gives a unitary $U_{\tilde{a}}$ obeying Eq. (3.10).

Using both Eq. (3.10) and its adjoint, we find

$$P_{\tilde{b}} P_{\tilde{a}} = U_{\tilde{b}} P U_{\tilde{b}}^\dagger U_{\tilde{a}} P U_{\tilde{a}}^\dagger = U_{\tilde{b}} (P E_{\tilde{b}}^\dagger E_{\tilde{a}} P) U_{\tilde{a}}^\dagger = \delta_{\tilde{a}\tilde{b}} U_{\tilde{a}} P U_{\tilde{a}}^\dagger = \delta_{\tilde{a}\tilde{b}} P_{\tilde{a}}, \tag{3.14}$$

which gives (3.12). □

We include the no-error operator and choose

$$E_{\tilde{0}} = \mathbb{I}, \qquad U_{\tilde{0}} = \mathbb{I}, \qquad P_{\tilde{0}} = P. \tag{3.15}$$

Each detectable error therefore maps the codespace to a different orthogonal syndrome sector, exactly as in the Shor code (Section 2.3). As illustrated in Figure 3.1, each class of correctable error $E_{\tilde{a}}$ kicks us into orthogonal subspaces $E_{\tilde{a}}\mathcal{C}$; projector $P_{\tilde{a}}$ is a projector onto only this subspace.

3.2.2 The decoder

Define the projector onto all states not reached by the normalized error operators:

$$P_\perp := \mathbb{I} - \sum_{\tilde{a}} P_{\tilde{a}}. \tag{3.16}$$

The recovery channel is

$$\mathcal{R}(\rho) = \sum_{\widetilde{a}} U_{\widetilde{a}}^{-1} P_{\widetilde{a}} \rho P_{\widetilde{a}} U_{\widetilde{a}} + P_\perp \rho P_\perp. \tag{3.17}$$

Operationally, the projectors measure which syndrome sector $\widetilde{a}$ is occupied, and $U_{\widetilde{a}}^{-1}$ rotates that sector back to the codespace. The last term specifies an arbitrary harmless action on states that the correctable errors cannot reach.

The Kraus operators of Eq. (3.17) are

$$L_{\widetilde{a}} = U_{\widetilde{a}}^{-1} P_{\widetilde{a}}, \qquad L_\perp = P_\perp. \tag{3.18}$$

They satisfy

$$\sum_{\widetilde{a}} L_{\widetilde{a}}^\dagger L_{\widetilde{a}} + L_\perp^\dagger L_\perp = \sum_{\widetilde{a}} P_{\widetilde{a}} U_{\widetilde{a}} U_{\widetilde{a}}^{-1} P_{\widetilde{a}} + P_\perp = \sum_{\widetilde{a}} P_{\widetilde{a}} + P_\perp = \mathbb{I}, \tag{3.19}$$

so $\mathcal{R}$ is a quantum channel.

Lemma 3.4 (Correction of the normalized basis errors)**:** For every $\widetilde{a}$ and every $\rho = P\rho P$,

$$\mathcal{R}(E_{\widetilde{a}} \rho E_{\widetilde{a}}^\dagger) = \rho. \tag{3.20}$$

Proof. The corrupted state must begin in the codespace. For every $\widetilde{b}$,

$$U_{\widetilde{b}}^{-1} P_{\widetilde{b}} E_{\widetilde{a}} P = P U_{\widetilde{b}}^\dagger E_{\widetilde{a}} P = P E_{\widetilde{b}}^\dagger E_{\widetilde{a}} P = \delta_{\widetilde{a}\widetilde{b}} P. \tag{3.21}$$

The $P_\perp$ term vanishes, and therefore

$$\mathcal{R}(E_{\widetilde{a}} P \rho P E_{\widetilde{a}}^\dagger) = \sum_{\widetilde{b}} U_{\widetilde{b}}^{-1} P_{\widetilde{b}} E_{\widetilde{a}} P \rho P E_{\widetilde{a}}^\dagger P_{\widetilde{b}} U_{\widetilde{b}} = \sum_{\widetilde{b}} \delta_{\widetilde{a}\widetilde{b}} P \rho P = P \rho P = \rho, \tag{3.22}$$

which gives (3.20). □

3.2.3 Arbitrary channels in the error span

Lemma 3.5 (Linearity of correctable errors)**:** The same recovery channel corrects every channel

$$\mathcal{N}(\rho) = \sum_i K_i \rho K_i^\dagger \tag{3.23}$$

whose Kraus operators obey

$$K_i \in \mathrm{span}\{E_{\widetilde{a}}\}. \tag{3.24}$$

Proof. By linearity, it is enough first to take one operator

$$K = \sum_{\widetilde{a}} k_{\widetilde{a}} E_{\widetilde{a}}. \tag{3.25}$$

Repeating the calculation above gives

$$\mathcal{R}(KP\rho PK^\dagger) = \sum_{\widetilde{b}} PE_{\widetilde{b}}^\dagger KP\rho PK^\dagger E_{\widetilde{b}} P = \sum_{\widetilde{a},\widetilde{a}',\widetilde{b}} k_{\widetilde{a}} \overline{k_{\widetilde{a}'}} \delta_{\widetilde{a}\widetilde{b}} \delta_{\widetilde{a}'\widetilde{b}} P\rho P = \left(\sum_{\widetilde{a}} |k_{\widetilde{a}}|^2 \right) P\rho P. \tag{3.26}$$

Consequently, $(\mathcal{R} \circ \mathcal{N})(P\rho P)$ is proportional to $P\rho P$. Since both $\mathcal{R}$ and $\mathcal{N}$ are channels, and (3.26) applies to every K_i in (3.23), their composition is trace preserving, so the proportionality constant is one:

$$(\mathcal{R} \circ \mathcal{N})(P\rho P) = P\rho P. \tag{3.27}$$

This completes the proof. □

3.2.4 Why a continuum of errors reduces to discrete syndromes

The preceding calculation is fast, but its physical content is crucial. Consider a coherent error acting on $|\psi\rangle \in \mathcal{C}$: for example,

$$(\alpha \mathbb{I} + \beta E_{\widetilde{1}} + \gamma E_{\widetilde{2}}) |\psi\rangle = \alpha |\psi\rangle + \beta E_{\widetilde{1}} |\psi\rangle + \gamma E_{\widetilde{2}} |\psi\rangle . \tag{3.28}$$

The three terms lie in orthogonal syndrome sectors. A syndrome measurement therefore returns

sector	probability	post-measurement state
P	$\lvert\alpha\rvert^2$	$\lvert\psi\rangle$
$P_{\widetilde{1}}$	$\lvert\beta\rvert^2$	$E_{\widetilde{1}} \lvert\psi\rangle$
$P_{\widetilde{2}}$	$\lvert\gamma\rvert^2$	$E_{\widetilde{2}} \lvert\psi\rangle$

(3.29)

for normalized coefficients. Lemma 3.4 then rotates the observed sector back to the codespace.

In this respect, quantum error correction resembles digital classical error correction: the decoder only needs to fix a discrete set of syndromes. The continuum of possible coherent errors changes the probabilities of those syndromes, rather than requiring a continuum of distinct correction operations. This is illustrated in Figure 3.2.

3.2.5 Errors that annihilate the codespace

Lemma 3.6 (Null directions in the error space)**:** The recovery above still works when the error space includes operators $E_{\bar{a}}$ obeying Eq. (3.8b).

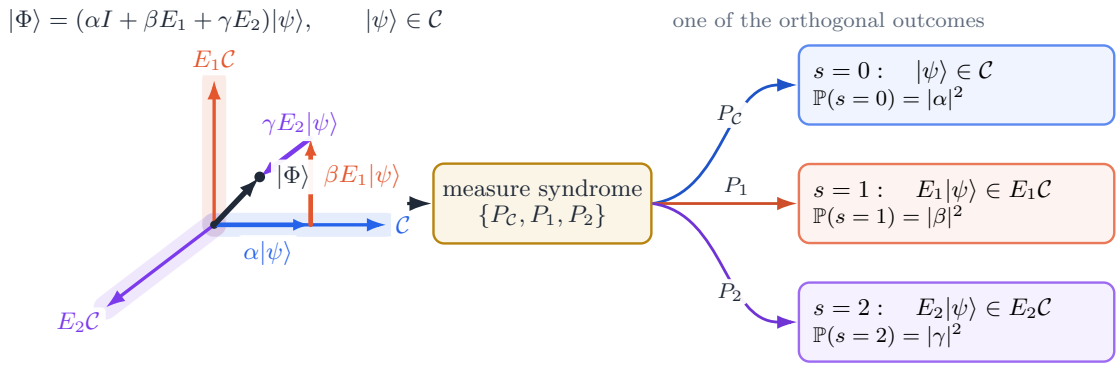

Figure 3.2: Measuring which syndrome sector we are in collapses the quantum state and "digitizes" a continuum of errors into a discrete set, which we can then correct using $\mathcal{R}$.

Proof. Taking $b = \bar{a}$ in Eq. (3.8b) gives

$$0 = PE_{\bar{a}}^\dagger E_{\bar{a}} P = (E_{\bar{a}} P)^\dagger (E_{\bar{a}} P), \tag{3.30}$$

so

$$E_{\bar{a}} P = 0. \tag{3.31}$$

For a general operator in the enlarged span,

$$K = \sum_{\widetilde{a}} k_{\widetilde{a}} E_{\widetilde{a}} + \sum_{\bar{a}} k_{\bar{a}} E_{\bar{a}}, \tag{3.32}$$

we therefore have

$$KP = \left(\sum_{\widetilde{a}} k_{\widetilde{a}} E_{\widetilde{a}} \right) P. \tag{3.33}$$

The problem reduces to Lemma 3.5. □

Example 3.7: A common null error has the form

$$\widetilde{E}(S - \mathbb{I}), \tag{3.34}$$

where S is a stabilizer, because $(S - \mathbb{I})P = 0$. For example, the Shor code corrects Z_1, Z_2, Z_3, although these errors are not distinguishable within the first three-qubit block. Since $Z_1 Z_2$ is a stabilizer,

$$Z_2 = Z_1 + Z_1(Z_1 Z_2 - \mathbb{I}). \tag{3.35}$$

The second term annihilates the codespace, so Z_2 has the same action on codewords as the representative Z_1.

3.3 Necessity: exact recovery implies Knill–Laflamme

The preceding lemmas prove that the Knill–Laflamme conditions imply quantum error correction. It remains to prove the converse.

3.3.1 Finishing the proof of Theorem 3.1

Lemma 3.8 (Necessity of the Knill–Laflamme conditions)**:** Suppose a channel

$$\mathcal{N}(\rho) = \sum_i K_i \rho K_i^\dagger \tag{3.36}$$

is exactly corrected on $\mathcal{C}$ by some recovery $\widetilde{\mathcal{R}}$:

$$(\widetilde{\mathcal{R}} \circ \mathcal{N})(P\rho P) = P\rho P. \tag{3.37}$$

Then there are constants α_{ij} such that

$$P K_i^\dagger K_j P = \alpha_{ij} P. \tag{3.38}$$

Proof. By Theorem 1.4, we may write the recovery channel as

$$\widetilde{\mathcal{R}}(\sigma) = \sum_\mu D_\mu \sigma D_\mu^\dagger. \tag{3.39}$$

The exact-recovery assumption becomes

$$\sum_{\mu,i} D_\mu K_i P \rho P K_i^\dagger D_\mu^\dagger = P\rho P. \tag{3.40}$$

We first claim that

$$D_\mu K_i P = c_{\mu i} P \tag{3.41}$$

for constants $c_{\mu i}$. To see this, choose $|\psi\rangle \in \mathcal{C}$ and any $|\phi\rangle \perp |\psi\rangle$. Applying Eq. (3.40) to $\rho = |\psi\rangle\langle\psi|$ gives

$$0 = \langle\phi| \left[(\widetilde{\mathcal{R}} \circ \mathcal{N} - \mathcal{I})(|\psi\rangle\langle\psi|)\right] |\phi\rangle = \sum_{\mu,i} |\langle\phi| D_\mu K_i |\psi\rangle|^2 - |\langle\phi|\psi\rangle|^2 = \sum_{\mu,i} |\langle\phi| D_\mu K_i |\psi\rangle|^2 . \tag{3.42}$$

Every nonnegative summand must vanish. Hence $D_\mu K_i |\psi\rangle$ is parallel to $|\psi\rangle$ for every codeword $|\psi\rangle$.

The proportionality factor cannot depend on the codeword. If $A = D_\mu K_i$ obeys $A|\psi\rangle = c_\psi|\psi\rangle$ and $A|\chi\rangle = c_\chi|\chi\rangle$ for two linearly independent codewords, then applying the same statement to their sum gives

$$c_\psi|\psi\rangle + c_\chi|\chi\rangle = c_{\psi+\chi}(|\psi\rangle + |\chi\rangle), \tag{3.43}$$

which forces $c_\psi = c_\chi = c_{\psi+\chi}$. This proves Eq. (3.41).

Finally, because $\widetilde{\mathcal{R}}$ is trace preserving, Theorem 1.4 implies that

$$\sum_\mu D_\mu^\dagger D_\mu = \mathbb{I}. \tag{3.44}$$

Using Eqs. (3.41) and (3.44), we obtain

$$PK_i^\dagger K_j P = \sum_\mu PK_i^\dagger D_\mu^\dagger D_\mu K_j P = \sum_\mu \overline{c_{\mu i}} c_{\mu j} P = \alpha_{ij} P, \tag{3.45}$$

where

$$\alpha_{ij} := \sum_\mu \overline{c_{\mu i}} c_{\mu j}. \tag{3.46}$$

This is precisely the Knill–Laflamme condition. □

Together with the sufficiency construction, this completes the proof of Theorem 3.1.

3.3.2 Physical interpretation

The necessity proof has a simple interpretation. If there were an operator associated with the error process whose expectation value differed between two orthogonal codewords, then measuring that operator—or allowing the environment to record the same information—would distinguish the logical states. In the limiting case of perfect distinguishability, the logical superposition would collapse as

$$\alpha|\psi\rangle + \beta|\phi\rangle \longrightarrow \begin{cases} |\psi\rangle, & \text{with probability } |\alpha|^2, \\ |\phi\rangle, & \text{with probability } |\beta|^2. \end{cases} \tag{3.47}$$

No recovery operation can reconstruct the lost relative phase.

Equivalently, in any orthonormal logical basis $\{|m_L\rangle\}$, the Knill–Laflamme condition reads

$$\langle m_L| E_a^\dagger E_b |n_L\rangle = \alpha_{ab}\delta_{mn}. \tag{3.48}$$

Thus the error process has identical diagonal matrix elements on every logical state and no off-diagonal matrix elements between distinct logical states: it cannot learn which encoded state was

present. We can therefore correct only errors that do not reveal logical information. This is the content of the Knill–Laflamme conditions.

Problems

Problem 3.1 (Correcting photon loss errors): Suppose that we built the Shor code and/or the 3-qubit repetition codes (Section 2) out of 9 atomic qubits described in Problem 1.1. Which of these codes can correct against an arbitrary single-qubit photon loss error? Explain your answer. Does the error correction strategy depend on the value of p from (1.49)?

Problem 3.2 (Storing a logical qubit inside a single qudit): Suppose that we have an experimental system with access to a single physical qudit with q internal levels, i.e. the Hilbert space is

$$\mathbb{C}^q = \text{span}\{|0\rangle, |1\rangle, \ldots, |q-1\rangle\}. \tag{3.49}$$

There is a natural generalization of Pauli X and Z operators on qubits:

$$X|j\rangle = |j+1 \ (\text{mod } q)\rangle, \tag{3.50a}$$
$$Z|j\rangle = \mathrm{e}^{2\pi \mathrm{i} j/q}|j\rangle. \tag{3.50b}$$

A short calculation shows that X and Z are unitary matrices, obeying $X^q = Z^q = I$ and

$$XZ = ZX\mathrm{e}^{-2\pi \mathrm{i}/q}. \tag{3.51}$$

This setup arises in the context of bosonic GKP codes, which can be realized in superconducting circuits. There are beautiful codes, which I call "torus codes", that allow you to store logical qubits inside the larger physical qudit. In this problem we will look at two such codes. The first such code uses $q = 8$, and imposes stabilizers $S_1 = Z^4$ and $S_2 = X^4$.

A: Check that $[S_1, S_2] = 0$. Show that the codespace is two-dimensional with logical codewords

$$|0_{\mathrm{L}}\rangle = \frac{|0\rangle + |4\rangle}{\sqrt{2}}, \tag{3.52a}$$
$$|1_{\mathrm{L}}\rangle = \frac{|2\rangle + |6\rangle}{\sqrt{2}}. \tag{3.52b}$$

Let P be the projector onto $\text{span}\{|0_{\mathrm{L}}\rangle, |1_{\mathrm{L}}\rangle\}$. Find a set of logical Pauli operators X_{L} and Z_{L}, obeying $X_{\mathrm{L}}^2 P = Z_{\mathrm{L}}^2 P = P$ and $X_{\mathrm{L}} Z_{\mathrm{L}} P = -Z_{\mathrm{L}} X_{\mathrm{L}} P$. Show that $[X_{\mathrm{L}}, S_1] = [X_{\mathrm{L}}, S_2] = [Z_{\mathrm{L}}, S_1] = [Z_{\mathrm{L}}, S_2] = 0$.

B: Show that one set of correctable errors is $\text{span}\{I, X, Z, XZ\}$. Is this choice unique? Why or why not?

C: Explicitly write down the recovery channel $\mathcal{R}$. Give a clear physical explanation for what it does and how you could implement it, in principle, experimentally.

D: Show that X and X^{-1} cannot both be corrected, and explain why this is so.

Now, let's discover a code that can correct all of the "smallest errors" in the system: namely, we'd like the correctable error set to be exactly $\text{span}(I, X, X^{-1}, Z, Z^{-1}) = \text{span}(E_a)$. We also do not want any of these errors to be degenerate: we demand $\alpha_{ab} = \delta_{ab}$.

E: Prove that you need $q \geq 10$ for this to be possible.

F: It is possible to find such a code using $q = 10$, with a single stabilizer of the form $S = X^a Z^b$, which has the property that $S^5 = I$, and the eigenvalues of S are the five roots of unity $e^{2\pi i j/5}$ for $j = 0, \ldots, 4$. Find a valid choice of a and b, being sure to explain why the choice works. As part of your answer, you should explain conceptually how to decode this code, although you don't need to explicitly write down $\mathcal{R}$. Find logical operators for this code.

4 1d Ising model as the repetition code

Our goal is to construct codes whose performance improves as the number n of physical bits or qubits grows. A larger code should protect against more errors, and we would also like to understand whether decoding can remain manageable. Analogies with physics provide useful insight into both questions. We begin with a minimal scalable example whose connection to statistical mechanics foreshadows many later ideas.

4.1 The scalable repetition code is the Ising ferromagnet

The length-n classical repetition code encodes one logical bit as

$$0_L \longleftrightarrow 00\cdots0, \qquad 1_L \longleftrightarrow 11\cdots1. \tag{4.1}$$

Previously we used $n = 3$, but there is no obstruction to taking n much larger.

Map the binary symbols to Ising spins by

$$0 \longleftrightarrow +1, \qquad 1 \longleftrightarrow -1. \tag{4.2}$$

Then the two codewords become the two ferromagnetic configurations

$$00\cdots0 \longleftrightarrow (+,+,\ldots,+), \qquad 11\cdots1 \longleftrightarrow (-,-,\ldots,-). \tag{4.3}$$

Let $z_i \in \{+1,-1\}$ be classical spin variables and consider the open one-dimensional Ising Hamiltonian

$$H_0(z) = -\sum_{i=1}^{n-1} z_i z_{i+1}, \tag{4.4}$$

as illustrated in Figure 4.1. A ground state must obey

$$z_i z_{i+1} = 1 \qquad \text{for every } i = 1,\ldots,n-1. \tag{4.5}$$

Multiplying by z_{i+1} and using $z_{i+1}^2 = 1$ gives

$$z_i = z_{i+1} \qquad \text{for every } i, \tag{4.6}$$

which is precisely the repetition-code constraint.

From the physics perspective, the two ground states are related by the global $\mathbb{Z}_2$ transformation

$$(z_1, z_2, \ldots, z_n) \longmapsto (-z_1, -z_2, \ldots, -z_n). \tag{4.7}$$

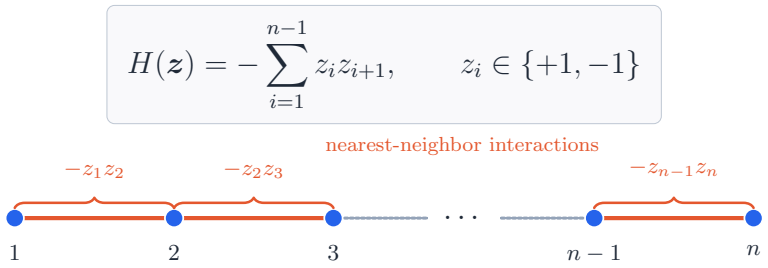

Figure 4.1: The repetition code with n classical bits can be mapped to the Ising model on a 1d chain of length n.

The Hamiltonian is invariant because

$$z_i z_{i+1} \longmapsto (-z_i)(-z_{i+1}) = z_i z_{i+1}. \tag{4.8}$$

Each individual ground state is not invariant under this symmetry: the two states are exchanged. In the thermodynamic language, the ferromagnetic ground states spontaneously break the global $\mathbb{Z}_2$ symmetry.

4.2 Independent bit flips and majority-vote decoding

We now return to the error-correction perspective and ask what is gained by using $n \gg 3$ physical bits. Consider an independent error model in which

$$z_i \longmapsto \eta_i z_i, \tag{4.9}$$

with independent random variables $\eta_i \in \{+1, -1\}$ distributed as

$$\mathbb{P}(\eta_i = +1) = 1 - p, \qquad \mathbb{P}(\eta_i = -1) = p. \tag{4.10}$$

Thus p is the probability that bit i flips. Define the indicator

$$e_i := \frac{1 - \eta_i}{2} = \begin{cases} 0, & \text{no error,} \\ 1, & \text{error.} \end{cases} \tag{4.11}$$

The total number of errors is

$$N := \sum_{i=1}^{n} e_i. \tag{4.12}$$

Its expectation value is

$$\mathbb{E}[N] = \sum_{i=1}^{n} \mathbb{E}[e_i] = n\big[(1-p)\cdot 0 + p \cdot 1\big] = pn. \tag{4.13}$$

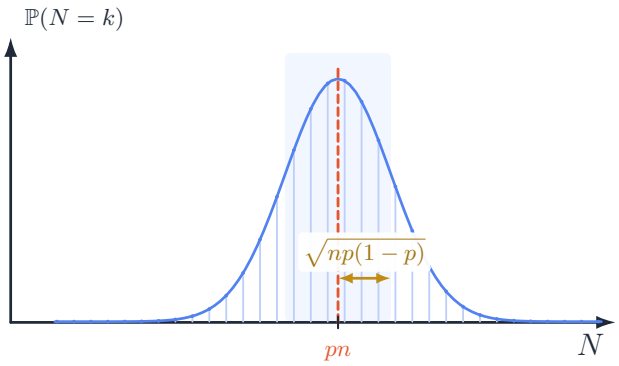

Figure 4.2: The independent errors make N a tightly peaked random variable in the large n limit.

Because $e_i^2 = e_i$ and distinct e_i, e_j are independent,

$$\mathbb{E}[N^2] = \sum_{i,j=1}^{n} \mathbb{E}[e_i e_j] = \sum_{i=1}^{n} \mathbb{E}[e_i] + 2\sum_{i<j} \mathbb{E}[e_i]\mathbb{E}[e_j] = pn + n(n-1)p^2 = (pn)^2 + np(1-p). \quad (4.14)$$

Therefore

$$\mathrm{Var}(N) := \mathbb{E}[N^2] - \mathbb{E}[N]^2 = np(1-p). \quad (4.15)$$

The distribution of N is consequently concentrated near pn, with a characteristic width

$$\sqrt{\mathrm{Var}(N)} = \sqrt{p(1-p)n} \ll pn \qquad (n \longrightarrow \infty) \quad (4.16)$$

for fixed $p > 0$: see Figure 4.2.

Suppose now that

$$0 < p < \frac{1}{2}, \quad (4.17)$$

and decode by majority vote.

Proposition 4.1: For independent bit flips with $p < 1/2$, the probability that majority-vote decoding succeeds approaches one as $n \to \infty$.

Proof. The decoder can fail only if at least half of the bits flip. Hence

$$\mathbb{P}(\text{decoder fails}) = \mathbb{P}\left(N \geq \frac{n}{2}\right) \leq \mathbb{P}\left(|N - pn| \geq \left(\frac{1}{2} - p\right) n\right). \quad (4.18)$$

Chebyshev's inequality and Eq. (4.15) give

$$\mathbb{P}(\text{decoder fails}) \leq \frac{\mathbb{E}[(N-pn)^2]}{\left[(\frac{1}{2}-p)n\right]^2} = \frac{p(1-p)}{n(\frac{1}{2}-p)^2} \longrightarrow 0 \qquad (n \longrightarrow \infty). \tag{4.19}$$

Thus the decoder almost surely works as $n \to \infty$. □

For even n, an exact tie can be declared a failure; the event $N \geq n/2$ used above is therefore a conservative failure event. Taking n odd removes the tie without changing the asymptotic conclusion.

4.3 Faulty measurements of the physical bits

A first apparent problem is that the physical-bit measurements may themselves be faulty. Model the observed spin as

$$z_i \longmapsto \theta_i \eta_i z_i, \tag{4.20}$$

where η_i is the intrinsic bit-flip error and θ_i is a measurement error, with

$$\mathbb{P}(\eta_i = -1) = p_e, \qquad \mathbb{P}(\theta_i = -1) = p_m. \tag{4.21}$$

The observed bit is wrong precisely when exactly one of η_i, θ_i is negative. Its effective error probability is therefore

$$p := \mathbb{P}(\theta_i \eta_i = -1) = p_e(1-p_m) + p_m(1-p_e). \tag{4.22}$$

The same majority-vote argument works whenever $p < 1/2$. In particular, this condition is achieved if

$$p_e < \frac{1}{2}, \qquad p_m < \frac{1}{2}. \tag{4.23}$$

Thus independent measurement faults do not alter Proposition 4.1.

4.4 Noisy syndrome measurements and MAP decoding

4.4.1 Why syndrome noise is more subtle

Quantum error correction measures syndromes rather than the physical qubits. For the repetition code, the available checks are the neighboring products

$$z_i z_{i+1}, \qquad i = 1, \ldots, n-1. \tag{4.24}$$

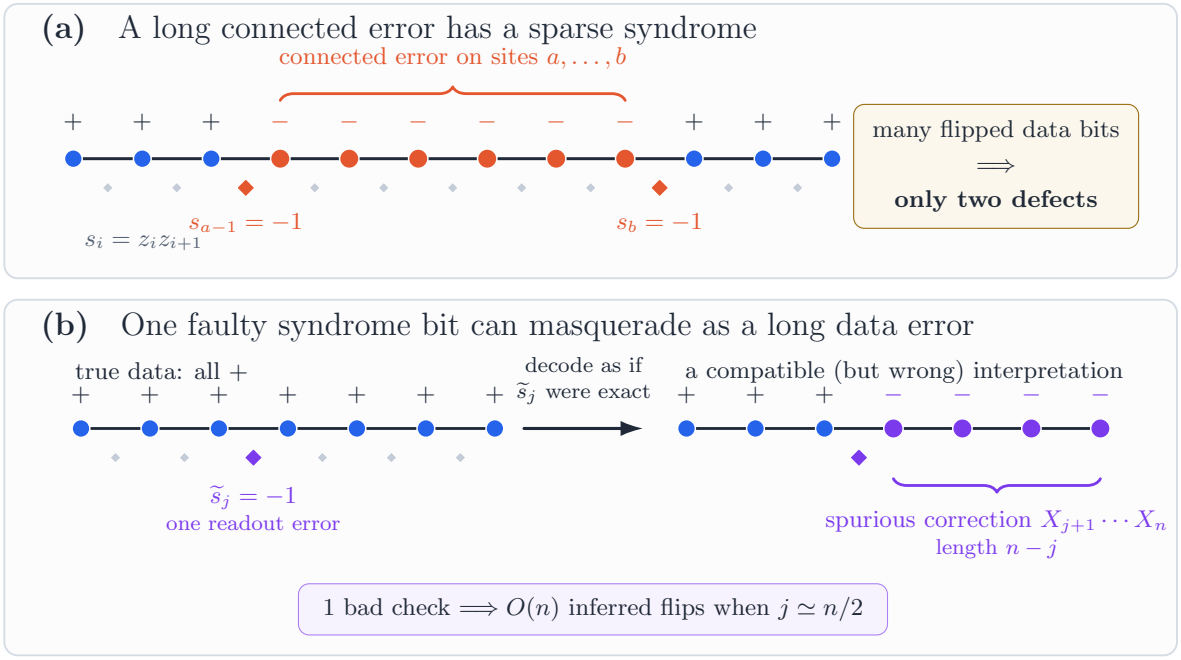

Figure 4.3: Overview of the challenges with decoding the 1d Ising model with noisy syndromes. (a) Large errors can correspond to small syndromes. (b) Not being skeptical of the syndrome can lead to catastrophic decoding errors that introduce very large errors.

A contiguous flipped interval changes only the checks at its endpoints. For example,

$$(z_1, \dots, z_n) = (+, +, \underbrace{-, -, \dots, -}_{\text{flipped interval}}, +) \tag{4.25}$$

has the syndrome pattern

$$(z_1 z_2, z_2 z_3, \dots, z_{n-1} z_n) = (+, -, +, \dots, +, -). \tag{4.26}$$

Only the two domain walls are visible; the syndrome does not directly reveal the length of the flipped interval.

Combined with syndrome-measurement noise, this creates a genuine ambiguity. For example, a measured pattern with one isolated negative check,

$$(s_1, \dots, s_{n-1}) = (+, -, +, \dots, +), \tag{4.27}$$

may be a single faulty syndrome measurement. If it is instead interpreted as a reliable domain wall, the decoder may infer a long boundary-attached error string such as

$$(z_1, \dots, z_n) = (+, +, -, -, \dots, -), \tag{4.28}$$

which is the wrong interpretation of the data: see Figure 4.3.

4.4.2 Bayesian formulation

Given noisy syndrome outcomes

$$s = (s_1, \dots, s_{n-1}), \qquad s_i \in \{+1, -1\}, \tag{4.29}$$

we seek the most likely error configuration

$$z = (z_1, \dots, z_n), \qquad z_i = \begin{cases} +1, & \text{no physical error,} \\ -1, & \text{physical error.} \end{cases} \tag{4.30}$$

Let $t_i \in \{+1, -1\}$ be the syndrome-measurement error, so that

$$s_i = z_i z_{i+1} t_i, \qquad \mathbb{P}(t_i = -1) = p_m. \tag{4.31}$$

The probability of the observed syndrome is

$$\mathbb{P}(s) = \sum_z \mathbb{P}(s \mid z)\mathbb{P}(z). \tag{4.32}$$

Bayes' theorem gives

$$\mathbb{P}(s, z) = \mathbb{P}(s \mid z)\mathbb{P}(z) = \mathbb{P}(z \mid s)\mathbb{P}(s), \tag{4.33}$$

and hence

$$\mathbb{P}(z \mid s) = \frac{\mathbb{P}(s \mid z)\mathbb{P}(z)}{\mathbb{P}(s)}. \tag{4.34}$$

Since $\mathbb{P}(s)$ is fixed by the measured data, the maximum a posteriori (MAP) decoder maximizes $\mathbb{P}(s \mid z)\mathbb{P}(z)$ over z.

4.4.3 Mapping to a random-bond Ising model

Under independent physical errors,

$$\mathbb{P}(z) = \prod_{i=1}^{n} \begin{cases} p_e, & z_i = -1, \\ 1 - p_e, & z_i = +1. \end{cases} \tag{4.35}$$

Introduce h through

$$\mathrm{e}^{-2h} = \frac{p_e}{1 - p_e}, \qquad h = \frac{1}{2} \log \frac{1 - p_e}{p_e}. \tag{4.36}$$

Then, up to a normalization independent of z,

$$\mathbb{P}(z) \propto \exp\left(h \sum_{i=1}^{n} z_i\right). \tag{4.37}$$

Similarly,

$$\mathbb{P}(s \mid z) = \prod_{i=1}^{n-1} \begin{cases} p_m, & z_i z_{i+1} s_i = -1, \\ 1 - p_m, & z_i z_{i+1} s_i = +1. \end{cases} \tag{4.38}$$

Introduce J by

$$\mathrm{e}^{-2J} = \frac{p_m}{1 - p_m}, \qquad J = \frac{1}{2} \log \frac{1 - p_m}{p_m}. \tag{4.39}$$

The likelihood is then

$$\mathbb{P}(s \mid z) \propto \exp\left(J \sum_{i=1}^{n-1} s_i z_i z_{i+1}\right). \tag{4.40}$$

Combining the prior and likelihood gives

$$\mathbb{P}(s \mid z)\mathbb{P}(z) \propto \exp\left[J \sum_{i=1}^{n-1} s_i z_i z_{i+1} + h \sum_{i=1}^{n} z_i\right]. \tag{4.41}$$

Equivalently, define the syndrome-dependent random-bond Ising Hamiltonian

$$H_s(z) := -J \sum_{i=1}^{n-1} s_i z_i z_{i+1} - h \sum_{i=1}^{n} z_i. \tag{4.42}$$

Proposition 4.2: The MAP estimate is a ground state of H_s. This decoder succeeds in the presence of a finite probability for both physical errors and syndrome-measurement errors. Problem 4.1 shows that even for adversarial errors, this decoder works whenever $p_{\mathrm{e}} = p_{\mathrm{m}} \leq 1/6$.

Although the decoder is global, this particular one-dimensional ground-state problem is computationally simple: one may minimize successively over the two possible values of each spin, keeping the best partial cost for each boundary spin. This dynamic-programming procedure takes time proportional to n. But the general mapping of the decoding problem to a statistical mechanics problem (random-bond Ising model, or generalization thereof) is more general, and we will return to it in the context of quantum codes in Section 12.

The decoding decision uses global information from the syndrome record. It is possible in principle to construct local decoders, but doing so is more subtle.

4.5 Finite temperature and the failure of passive order in one dimension

The coding problem suggests an analogy between finite error probability and finite temperature. In fact, we will see in Section 5.6 that some codes admit very simple decoders based on mimicking the process of thermalizing the system with a bath.

Unfortunately, this analogy to physics does not work for the 1d Ising model. Consider the thermal partition function of Eq. (4.4):

$$Z_n(\beta) := \sum_{z_1,\ldots,z_n} \mathrm{e}^{-\beta H(z)} \tag{4.43}$$

$$= \sum_{z_1,\ldots,z_n} \exp\left(\beta \sum_{i=1}^{n-1} z_i z_{i+1}\right). \tag{4.44}$$

Introduce the transfer matrix

$$T_\beta := \begin{pmatrix} \mathrm{e}^{\beta} & \mathrm{e}^{-\beta} \\ \mathrm{e}^{-\beta} & \mathrm{e}^{\beta} \end{pmatrix}, \tag{4.45}$$

whose row and column indices are the two possible spin values. For open boundary conditions,

$$Z_n(\beta) = \begin{pmatrix} 1 & 1 \end{pmatrix} T_\beta^{n-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{4.46}$$

The vector $(1,1)^{\mathsf{T}}$ is an eigenvector:

$$T_\beta \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 2\cosh(\beta) \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{4.47}$$

Therefore

$$Z_n(\beta) = 2\big(2\cosh\beta\big)^{n-1}. \tag{4.48}$$

The finite-size free energy is

$$F_n(\beta) = -\frac{1}{\beta}\log Z_n(\beta) = -\frac{1}{\beta}\left[\log 2 + (n-1)\log(2\cosh\beta)\right], \tag{4.49}$$

and the thermodynamic free-energy density is

$$f(\beta) := \lim_{n\to\infty} \frac{F_n(\beta)}{n} = -\frac{1}{\beta}\log(2\cosh\beta). \tag{4.50}$$

This expression is nonsingular at every finite β, so the one-dimensional Ising model has no finite-temperature thermodynamic phase transition.

Strictly speaking, this does not necessarily show that there is no dynamical stability of the

ferromagnet! However in the 1d Ising model there is genuinely no passive memory. A more physical picture as to why is that domain walls between + and − easily proliferate.

Indeed, thinking about domain walls gives us another nice way to see why the 1d Ising model at finite temperature wouldn't protect information well. The free-energy cost of a domain wall scales as

$$\Delta F_{\mathrm{dw}} \sim \Delta E_{\mathrm{dw}} - T\Delta S_{\mathrm{dw}}, \tag{4.51}$$

where

$$\Delta E_{\mathrm{dw}} = O(1), \qquad \Delta S_{\mathrm{dw}} \sim \log n. \tag{4.52}$$

Thus, at every fixed $T > 0$, the entropic gain eventually overwhelms the finite energy cost as n grows. Domain walls proliferate and destroy long-range ferromagnetic order.

This distinction is important: active majority-vote decoding succeeds whenever $p < 1/2$, while the bare one-dimensional Ising Hamiltonian does not store the bit passively at any nonzero temperature. A decoder that can succeed and a finite-temperature ordered phase are related ideas, but they are not the same notion of protection.

Problems

Problem 4.1 (Decoding adversarial errors)**:** Consider the setup in Section 4.4.3: a fraction p_{e} of all bits have errors and p_{m} of all syndromes are incorrectly measured. Assume that these errors may be *adversarially placed*, but continue to use the MAP decoder as if the errors are random. Apply the most likely error $\widehat{z}_i$ predicted by MAP (given the observed syndrome), and ask whether

$$\frac{1}{n}\sum_{i=1}^{n} \widehat{z}_i z_i > 0? \tag{4.53}$$

If so, we're closer to the initial codeword than its logical counterpart, so the decoder has succeeded.

A: Show that when p_{e} and p_{m} are both sufficiently small, the decoder is guaranteed to work – i.e. (4.53) is obeyed. To do this, compare the highest possible energy of the state $\widehat{z}_i = z_i$ with error fractions p_{e} and p_{m}, with the lowest possible energy of any state that violates (4.53). Show that, in the $n \to \infty$ limit, the state with $\widehat{z}_i = z_i$ necessarily has lower energy whenever

$$\left(\frac{1}{2} - 2p_{\mathrm{e}}\right) \log \frac{1-p_{\mathrm{e}}}{p_{\mathrm{e}}} > p_{\mathrm{m}} \log \frac{1-p_{\mathrm{m}}}{p_{\mathrm{m}}}. \tag{4.54}$$

Deduce that the decoder necessarily works for *random* errors with $p_{\mathrm{e}} = p_{\mathrm{m}} = p$ so long as

$$p \leq \frac{1}{6}. \tag{4.55}$$

B: Do you think we have a high probability of exactly correcting all the errors in the limit $n \to \infty$, i.e. finding $\widehat{z}_i = z_i$? Why or why not?

Problem 4.2 (Correlation decay)**:** A better way to see that there is no finite-temperature protection in the 1d Ising model is to show that

$$\langle z_i z_j \rangle := \frac{1}{Z(\beta)} \sum_{z_1,\ldots,z_n} z_i z_j \mathrm{e}^{-\beta H(z)} \sim \mathrm{e}^{-|i-j|/\xi}. \tag{4.56}$$

Do this calculation using the same transfer matrix method used to derive $Z(\beta)$, and show that the constant $\xi < \infty$ for any $\beta < \infty$. This means that in a sufficiently long chain, there is no correlation between the local value of the codeword.

5 2d Ising model

In Section 4, the classical repetition code was identified with the one-dimensional Ising model. That analogy supplied a physical picture for error correction, but its real utility becomes clearer in two spatial dimensions.

5.1 From the repetition code to the 2d Ising model

Consider an $L \times L$ square lattice, with a classical spin $z_i \in \{+1, -1\}$ at each vertex. The ferromagnetic Ising Hamiltonian is

$$H_L(z) = -\sum_{\langle i,j \rangle} z_i z_j, \tag{5.1}$$

where the sum runs over neighboring vertices. As in one dimension, the two ground states are the two repetition-code words,

$$0_L \longleftrightarrow z_i = +1 \text{ for every } i, \qquad 1_L \longleftrightarrow z_i = -1 \text{ for every } i. \tag{5.2}$$

At inverse temperature β, the thermal Gibbs distribution is

$$\mathbb{P}_\beta(z) := \frac{1}{Z_L(\beta)} \mathrm{e}^{-\beta H_L(z)}, \tag{5.3a}$$

$$Z_L(\beta) := \sum_z \mathrm{e}^{-\beta H_L(z)}. \tag{5.3b}$$

The two-dimensional model has a genuine phase transition: see Figure 5.1. The low-temperature ferromagnetic phase will be useful for error correction.

For every edge $\langle i, j \rangle$, define the bond check

$$s_{ij} := z_i z_j \in \{+1, -1\}. \tag{5.4}$$

A check is violated when $s_{ij} = -1$. Drawing a dual edge across every violated bond produces the domain walls separating regions of positive and negative spins. If these domain walls do not intersect the boundary, they form closed loops, as illustrated in Figure 5.2. We will see that this property leads to a drastic change in the physical behavior of the 2d Ising model vs. 1d Ising model!

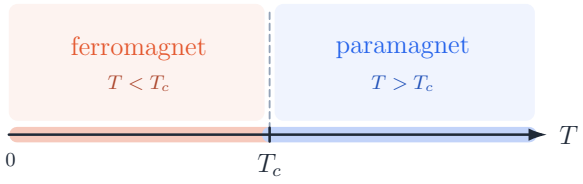

Figure 5.1: The phase diagram of the 2d Ising model.

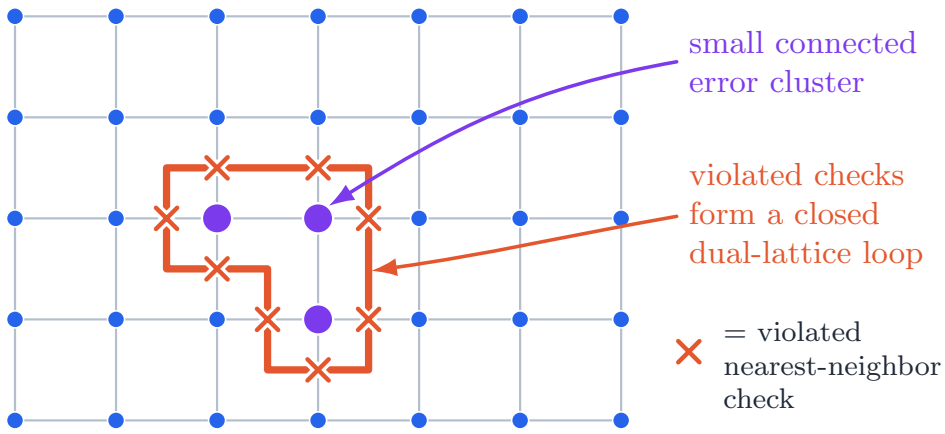

Figure 5.2: Violated parity-checks in the 2d Ising model form closed loops (except at the boundaries). Detecting error clusters is much easier!

Each violated bond changes its contribution to Eq. (5.1) from -1 to $+1$, so a domain wall of length ℓ has energy cost

$$\Delta E_{\mathrm{dw}} = 2\ell. \tag{5.5}$$

Proposition 5.1 (Linear energy barrier)**:** Any sequence of single-spin flips that carries the all-positive codeword to the all-negative codeword must pass through a configuration containing a domain wall whose length is at least of order $L - 1$. Consequently, the maximum energy encountered along the path obeys

$$\Delta E_{\mathrm{barrier}} \gtrsim 2(L - 1). \tag{5.6}$$

Proof sketch. The reason is topological. Initially the positive spins percolate across the sample, whereas finally the negative spins do. At an intermediate stage the percolating region must be cut by a domain wall joining opposite boundaries: see Figure 5.3. Such a wall has length at least of order $L - 1$, and every edge of the wall costs two units of energy. □

5.2 Peierls' argument

The energy of a long domain wall grows with its length, but so does its entropy. A crude Peierls estimate compares these two effects. Once one edge of a domain wall is fixed, there are at most three choices for the next step: the wall cannot immediately retrace the edge from which it arrived.

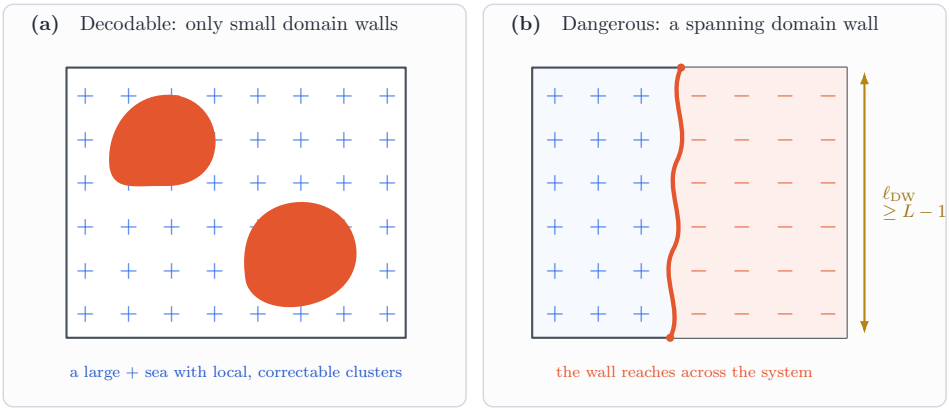

Figure 5.3: Left: small error clusters in the 2d Ising model are easily decodable and do not destroy long-range order. Right: dangerous "non-decodable" configurations can be taken as those where a long domain wall stretches across the entire system.

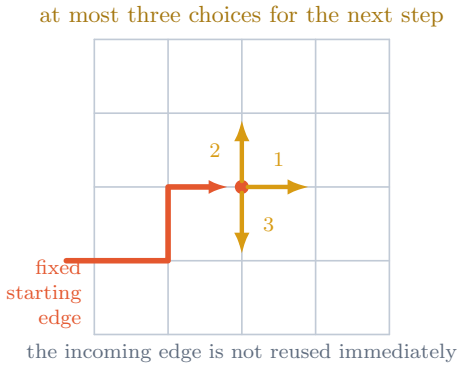

Figure 5.4: Justification for overcounting the number of domain walls in (5.7).

Hence the number of length-ℓ walls passing through a fixed dual edge is bounded by (see Figure 5.4)

$$N_\ell \leq 3^\ell. \tag{5.7}$$

The corresponding entropy is bounded by

$$\Delta S_{\mathrm{dw}} \lesssim \log N_\ell \leq \ell \log 3. \tag{5.8}$$

Combining this estimate with Eq. (5.5) gives the free-energy cost

$$\Delta F_{\mathrm{dw}} = \Delta E_{\mathrm{dw}} - T\Delta S_{\mathrm{dw}} \gtrsim 2\ell - T\log\left(3^\ell\right) = \ell(2 - T\log 3). \tag{5.9}$$

Therefore domain walls are disfavored whenever

$$T < \frac{2}{\log 3}. \tag{5.10}$$

This is a lower bound on the actual critical temperature, not an exact calculation of T_c. It is nevertheless enough to establish a genuine low-temperature ordered phase.

The crude count (5.7) overcounts contours by not restricting to closed, self-avoiding contours, but it correctly captures the exponential competition between contour entropy and Boltzmann suppression.

5.3 Consequences of the low-temperature phase

We will not calculate the exact location of the phase transition. Instead, we summarize the properties of the ferromagnetic phase $T < T_c$ that matter for error correction.

5.3.1 Long-range order

Small error clusters are present at nonzero temperature, but large domain walls remain rare. In the thermodynamic limit, the two-point function obeys

$$\lim_{\mathrm{dist}(i,j)\to\infty} \langle z_i z_j \rangle_\beta > 0, \tag{5.11}$$

where

$$\langle z_i z_j \rangle_\beta := \frac{1}{Z_L(\beta)} \sum_z \mathrm{e}^{-\beta H_L(z)} z_i z_j. \tag{5.12}$$

This is the signature of spontaneous symmetry breaking and long-range ferromagnetic order.

5.3.2 Thermodynamic nonanalyticity

Define the finite-volume free energy and its density by

$$F_L(\beta) := -\frac{1}{\beta} \log Z_L(\beta), \tag{5.13a}$$

$$f(\beta) := \lim_{L\to\infty} \frac{F_L(\beta)}{L^2} = -\lim_{L\to\infty} \frac{1}{\beta L^2} \log Z_L(\beta). \tag{5.13b}$$

The function $f(\beta)$ is nonanalytic at $\beta = \beta_c$, so the phase transition is thermodynamically detectable.

5.3.3 Concentration of the Gibbs measure

For $\beta > \beta_c$, configuration space separates into two large regions A_+ and A_-, concentrated near the all-positive and all-negative codewords. By the global $\mathbb{Z}_2$ symmetry, their probabilities are

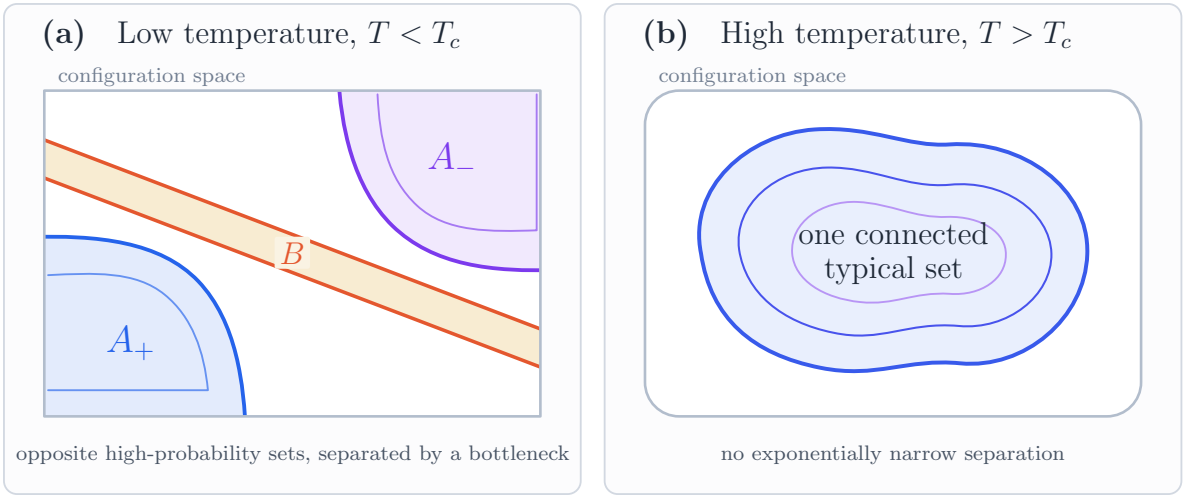

Figure 5.5: A cartoon of where the probability distribution $\mathbb{P}_\beta$ is domianted at: (a) $T < T_c$; (b) $T > T_c$.

approximately equal:

$$\mathbb{P}_\beta(A_+) = \mathbb{P}_\beta(A_-) \simeq \frac{1}{2}. \tag{5.14}$$

The two regions are separated by a bottleneck B consisting of configurations with a system-spanning domain wall. At low temperature,

$$\mathbb{P}_\beta(B) \sim \mathrm{e}^{-\alpha L} \qquad \text{for some } \alpha > 0. \tag{5.15}$$

Thus a thermal configuration is overwhelmingly likely to be found near one of the two codewords, even though it generally contains small error clusters. At high temperature, $\beta < \beta_c$, these two regions merge into one broad cluster in configuration space, and typical states have no long-range order. See Figure 5.5 for illustrations.

5.4 Gibbs samplers and detailed balance

Ferromagnetism suggests a passive memory: a local thermal process should remain within one of the two ordered basins for a very long time. A *Gibbs sampler* is a Markov chain whose stationary distribution is the thermal Gibbs distribution.

Definition 5.2: Consider single-spin Metropolis updates. Let $z^{(i)}$ denote the configuration obtained from z by flipping spin i. Choose a site uniformly and accept the proposed flip with probability

$$a(z \to z^{(i)}) := \min\left\{1, \mathrm{e}^{-\beta[H_L(z^{(i)}) - H_L(z)]}\right\}. \tag{5.16}$$

The transition probabilities may therefore be written as

$$P(z \to z^{(i)}) = \frac{1}{L^2} a(z \to z^{(i)}), \tag{5.17}$$

with

$$P(z \to z') = 0 \qquad \text{if } z \text{ and } z' \text{ differ on two or more spins.} \tag{5.18}$$

The rejection probability is included in the self-transition,

$$P(z \to z) = 1 - \sum_i P(z \to z^{(i)}). \tag{5.19}$$

The Metropolis rule obeys detailed balance:

$$P(z \to z') \mathrm{e}^{-\beta H_L(z)} = P(z' \to z) \mathrm{e}^{-\beta H_L(z')}. \tag{5.20}$$

Proposition 5.3: If the initial configuration is sampled from the thermal distribution, then one Markov step leaves that distribution invariant:

$$\begin{aligned} \mathbb{P}_{\text{after}}(z') &= \sum_z P(z \to z') \frac{\mathrm{e}^{-\beta H_L(z)}}{Z_L(\beta)} = \sum_z P(z' \to z) \frac{\mathrm{e}^{-\beta H_L(z')}}{Z_L(\beta)} \\ &= \frac{\mathrm{e}^{-\beta H_L(z')}}{Z_L(\beta)} \sum_z P(z' \to z) = \mathbb{P}_\beta(z'). \end{aligned} \tag{5.21}$$

Thus the Gibbs state is guaranteed to be stationary.

5.5 Bottlenecks and exponentially long memory times

The separation of the two ordered regions by an exponentially unlikely set implies a long memory time for any local Gibbs sampler.

Theorem 5.4 (Bottleneck estimate)**:** Let A, B, C be regions of configuration space such that every local path from A to C must pass through B, as illustrated in Figure 5.6. For a reversible Markov chain with equilibrium distribution $\mathbb{P}_{\text{eq}}$, the characteristic time τ to cross from one large basin to the other satisfies, up to $\mathrm{O}(1)$ constants,

$$\mathbb{E}[\tau] \gtrsim \frac{\min\{\mathbb{P}_{\text{eq}}(A), \mathbb{P}_{\text{eq}}(C)\}}{\mathbb{P}_{\text{eq}}(B)}. \tag{5.22}$$

Proof sketch. Restrict the dynamics to $A \cup B$, reflecting any attempted transition out of this set. The restricted equilibrium measure assigns probability

$$\mathbb{P}_{AB}(B) = \frac{\mathbb{P}_{\text{eq}}(B)}{\mathbb{P}_{\text{eq}}(A) + \mathbb{P}_{\text{eq}}(B)} \tag{5.23}$$

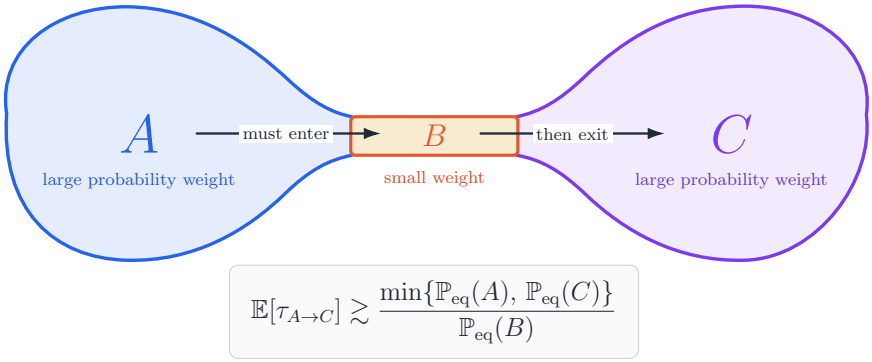

Figure 5.6: A sketch of the sets A, B and C in Theorem 5.4.

to the bottleneck. Until the first visit to B, the restricted dynamics and the full dynamics agree. If the restricted chain starts in its stationary distribution, then

$$\mathbb{P}(\tau_B \le t) \le \sum_{s=0}^{t} \mathbb{P}(X_s \in B) = (t+1)\mathbb{P}_{AB}(B). \tag{5.24}$$

A time of order $1/\pi_{AB}(B)$ is therefore needed before a visit to B is likely. Repeating the argument from the C side gives Eq. (5.22). □

A fully rigorous Markov-chain statement is usually formulated using conductance and specifies the initial distribution and a precise hitting or mixing time. The estimate above is the scaling statement used in this lecture: an exponentially small bottleneck measure produces an exponentially large crossing time.

For the low-temperature two-dimensional Ising model, take B_L to be the set of configurations containing a domain wall of length at least $L-1$, as described in Section 5.3.3:

$$B_L := \{z : \ z \text{ contains a domain wall of length at least } L-1\}. \tag{5.25}$$

There are at most order $L^2 3^\ell$ possible placements and shapes of a wall of length ℓ. As shown in Figure 5.7, removing one selected long wall produces a configuration $z_\star$ of lower energy while leaving the other, smaller walls untouched. For the normalization of Eq. (5.1),

$$\frac{\mathbb{P}_\beta(z)}{\mathbb{P}_\beta(z_\star)} = \mathrm{e}^{-\beta[H_L(z)-H_L(z_\star)]} \le \mathrm{e}^{-2\beta\ell}. \tag{5.26}$$

The Peierls count then gives

$$\mathbb{P}_\beta(B_L) \lesssim \sum_{\ell \ge L-1} L^2 3^\ell \mathrm{e}^{-2\beta\ell} \lesssim \mathrm{e}^{-\alpha L} \qquad \text{for sufficiently large } \beta, \tag{5.27}$$

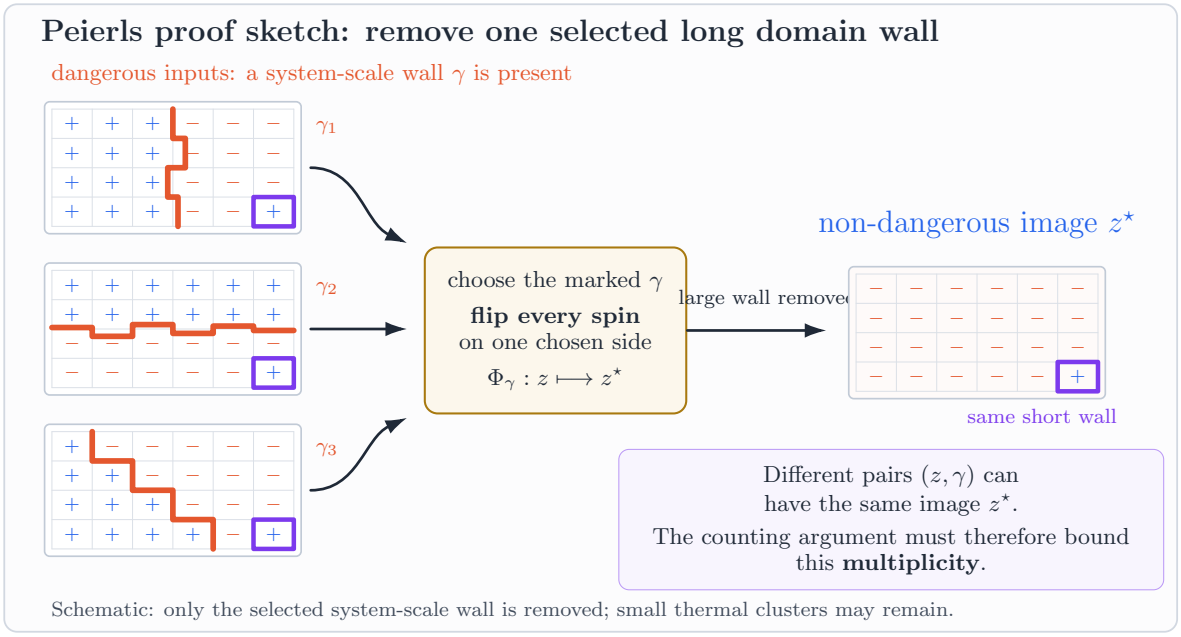

Figure 5.7: Identifying the bottleneck set B in the 2d Ising model and establishing (5.26) proceeds by counting the number of large domain walls that can exist, and showing that the entire probability of all such large-domain configurations is smaller than the probability of the (many fewer) states without these large domain walls.

where $\alpha > 0$; this is (5.15). Combining this with Theorem 5.4 yields

$$\mathbb{E}[\tau] \gtrsim \mathrm{e}^{\alpha L}. \tag{5.28}$$

It is essential that the bottleneck is defined by a *large* domain wall. A typical thermal configuration has a finite density of small domain walls. The dynamics need not remove every error simultaneously; it only needs to prevent those local errors from assembling into a system-spanning wall that changes the stored logical bit.

5.6 The Gibbs sampler as a local, fault-tolerant decoder

Consider a bulk site labeled 1, with its four neighboring sites labeled $2, 3, 4, 5$, as in Figure 5.8. The part of the energy involving z_1 is

$$H_1(z) = -z_1(z_2 + z_3 + z_4 + z_5). \tag{5.29}$$

Flipping z_1 changes the energy by

$$\Delta E_1 := H_L(z^{(1)}) - H_L(z) = 2z_1(z_2 + z_3 + z_4 + z_5) = 2(s_{12} + s_{13} + s_{14} + s_{15}). \tag{5.30}$$

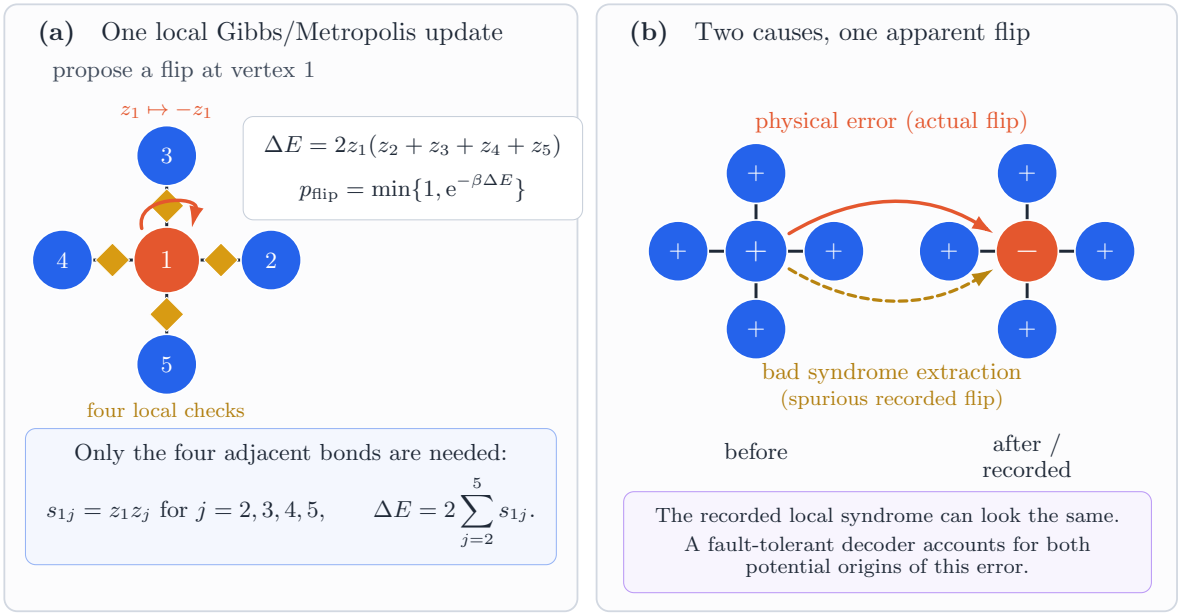

Figure 5.8: A sketch of how the local Gibbs sampler acts on a single vertex in the 2d Ising model by only measuring the local syndromes. As long as we ensure that (5.31) is obeyed, we can in principle convert the local Gibbs sampler into an local and fault-tolerant "passive" decoder.

Thus the Metropolis decision requires only local information about the four neighboring syndrome checks. It does not measure the logical bit. The thermal decoder flips site 1 with probability $\min\left\{1, e^{-\beta \Delta E_1}\right\}$. This is precisely the kind of local update that is safe for error correction.

We say that a decoder has a **threshold** if, when subject to repeated rounds of physical errors and noisy syndrome extraction followed by error correction, the probability for a logical error vanishes as the code increases in size ($n \to \infty$). To see how a nonzero fault-tolerance threshold can arise, consider the worst local energy-increasing move. If all four neighboring checks are initially satisfied, then flipping the central spin creates four violated checks and costs $\Delta E_1 = 8$. Let γ_{phys} be the rate of a physical spin error, $\gamma_{\text{dec}}^{\text{fail}}$ the rate at which a faulty decoder produces the same upward transition, and $\gamma_{\text{dec}}^{\text{succ}}$ the rate of the reverse successful correction. To reproduce detailed balance at inverse temperature β, the local rates should satisfy

$$\gamma_{\text{phys}} + \gamma_{\text{dec}}^{\text{fail}} = e^{-8\beta} \gamma_{\text{dec}}^{\text{succ}}. \tag{5.31}$$

At fixed β, the factor $e^{-8\beta}$ is a nonzero constant independent of the system size. The physical-error and decoder-failure rates may therefore be a finite fraction of the successful-decoding rate while the combined dynamics still behaves like a low-temperature Gibbs sampler. This is the origin of a finite threshold.

For other local patterns, an exact finite-temperature sampler may intentionally accept some

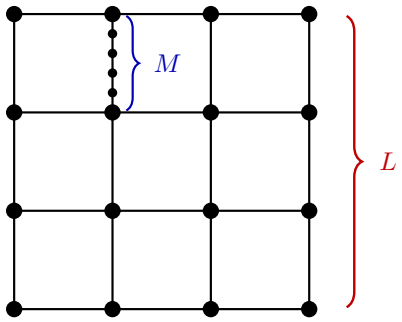

Figure 5.9: Sketch of the modified 2d Ising model where every edge of the square lattice is replaced by M sequential edges.

energy-increasing moves. Schematically, the rates obey

$$\gamma_{\text{phys}} + \gamma_{\text{dec}}^{\text{fail}} + \gamma_{\text{add}}^{\text{err}} = \mathrm{e}^{-\mathrm{O}(\beta)} \gamma_{\text{dec}}^{\text{succ}}, \tag{5.32}$$

where $\gamma_{\text{add}}^{\text{err}}$ denotes the deliberate energy-raising moves needed to match the target Gibbs sampler. A thermal decoder does not monotonically decrease the energy; it preserves the desired low-temperature distribution.

Thus a physical phase of matter can inspire a robust, local, fault-tolerant decoder; classical magnetic memory similarly stores a bit in a ferromagnet.

Problems

Problem 5.1 (A bad 2d Ising model): Consider the Ising model (repetition code) on the following modified $L \times L$ 2d square lattice, where each edge is replaced by M sequential edges (i.e. $M-1$ additional vertices are added along each edge): see Figure 5.9. When $M, L \gg 1$, the total number of bits is then $n \approx 2ML^2$. Freely use $M, L \gg 1$ to simplify answers as you wish.

A: What is the energy barrier that separates the two logical codewords of this peculiar repetition code? The energy of a configuration is given by the natural generalization of (5.1): we check whether each pair of two adjacent bits are the same or not.

B: Fix $M \gg 1$ and then take $L \to \infty$. Estimate the critical temperature T_{c}, such that there is ferromagnetism and the code is a passive memory for $T < T_{\text{c}}$.

C: Take $M \sim L^{\alpha}$ for some exponent $\alpha > 0$. Deduce that, in the thermodynamic limit, diverging energy barriers are not sufficient for passive memory.

6 Classical parity-check codes

6.1 From the Ising repetition code to linear algebra

The one-dimensional Ising model describes the classical repetition code. Its codewords are $0\,0\,0\,0\cdots$ and $1\,1\,1\,1\cdots$ which look a lot like the two ground states of the ferromagnetic energy

$$E(z) = -\sum_{i\sim j} z_i z_j, \qquad z_i \in \{+1, -1\} \tag{6.1}$$

if we replaced 0 and 1 with spin configurations $+1$ and -1. But let's work with the bits $\{0, 1\}$ instead, and write

$$x_i := \frac{1 - z_i}{2} \in \{0, 1\}. \tag{6.2}$$

Then a bond is satisfied or violated according to

$$z_i z_j = \begin{cases} +1, & x_i + x_j = 0 \pmod 2, \\ -1, & x_i + x_j = 1 \pmod 2. \end{cases} \tag{6.3}$$

Thus the ferromagnetic ground-state condition is a parity-check condition. It is helpful to express it in a more mathematical language: addition and multiplication are henceforth understood modulo two, so that we work over the field $\mathbb{F}_2$. An n-bit string is a vector

$$\mathbf{x} = (x_1, \ldots, x_n)^{\mathsf{T}} \in \mathbb{F}_2^n. \tag{6.4}$$

For the one-dimensional repetition code, the neighboring parity checks are

$$x_1 + x_2 = 0, \qquad x_2 + x_3 = 0, \qquad \ldots, \qquad x_{n-1} + x_n = 0. \tag{6.5}$$

They can be collected into the matrix equation

$$H_{\text{rep}}\mathbf{x} = \mathbf{0}, \qquad H_{\text{rep}} := \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 1 \end{pmatrix} \in \mathbb{F}_2^{(n-1)\times n}. \tag{6.6}$$

The matrix H is called a *parity-check matrix*. A bit string is a codeword precisely when it is a right null vector of H:

$$\mathbf{x} \text{ is a codeword} \iff H\mathbf{x} = \mathbf{0} \iff \mathbf{x} \in \ker(H). \tag{6.7}$$

For the repetition code,

$$\ker(H_{\text{rep}}) = \text{span}_{\mathbb{F}_2} \left\{ \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \right\} = \left\{ \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \right\}. \tag{6.8}$$

A *generator matrix* has rows that generate the codewords. For the repetition code one may take

$$G_{\text{rep}} = \begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix}. \tag{6.9}$$

It obeys

$$H_{\text{rep}} G_{\text{rep}}^{\mathsf{T}} = 0, \qquad \ker(H_{\text{rep}}) = \text{im}(G_{\text{rep}}^{\mathsf{T}}). \tag{6.10}$$

This may seem like an overly complicated description of the repetition code, but it has an immediate generalization: other classical parity-check codes are obtained by choosing different matrices H and G.

6.2 Dimension, distance, and syndrome decoding

The first goal is to store more than one logical bit, which requires a larger kernel.

Theorem 6.1 (Rank-nullity over $\mathbb{F}_2$)**:** For a parity-check matrix $H \in \mathbb{F}_2^{m \times n}$, the code $\ker(H)$ is a vector space of dimension

$$k := \dim \ker(H) = n - \text{rank}(H). \tag{6.11}$$

The integer k is the number of logical bits, and the code contains 2^k codewords.

Choosing a basis of $\ker(H)$ as the rows of a matrix $G \in \mathbb{F}_2^{k \times n}$ gives

$$HG^{\mathsf{T}} = 0, \qquad \ker(H) = \text{im}(G^{\mathsf{T}}). \tag{6.12}$$

The second goal is robustness against errors.

Definition 6.2 (Hamming weight and code distance)**:** The Hamming weight of a vector is the number of its nonzero entries:

$$|\mathbf{x}| := \#\{i : x_i = 1\}. \tag{6.13}$$

The distance of a linear code is the smallest weight of a nonzero codeword:

$$d := \min_{\substack{\mathbf{x}\in\ker(H)\\ \mathbf{x}\neq\mathbf{0}}} |\mathbf{x}|. \tag{6.14}$$

It is the minimum number of bit flips needed to change one codeword into a different codeword. Suppose that a codeword $\mathbf{x} \in \ker(H)$ suffers a bit-flip error $\mathbf{e}$, so that the received string is

$$\mathbf{r} = \mathbf{x} + \mathbf{e}. \tag{6.15}$$

A minimum-distance decoder looks for a valid codeword $\mathbf{y} \in \ker(H)$ that minimizes

$$|\mathbf{r} + \mathbf{y}| = |(\mathbf{x} + \mathbf{e}) + \mathbf{y}|. \tag{6.16}$$

Here subtraction and addition are identical because the arithmetic is over $\mathbb{F}_2$.

Proposition 6.3 (Correction and detection from the distance)**:** A minimum-distance decoder corrects every error satisfying

$$|\mathbf{e}| \leq t, \qquad t := \left\lfloor \frac{d-1}{2} \right\rfloor. \tag{6.17}$$

Moreover, every nonzero error of weight at most $d - 1$ is detectable.

Proof. For any other codeword $\mathbf{y} \neq \mathbf{x}$, the Hamming triangle inequality gives

$$|\mathbf{r} + \mathbf{y}| = |\mathbf{e} + (\mathbf{x} + \mathbf{y})| \geq |\mathbf{x} + \mathbf{y}| - |\mathbf{e}| \geq d - |\mathbf{e}| > |\mathbf{e}| = |\mathbf{r} + \mathbf{x}|. \tag{6.18}$$

Thus $\mathbf{x}$ is the unique nearest codeword whenever $2|\mathbf{e}| < d$. An error is undetectable only when it is a nonzero codeword, whose weight is at least d. □

In practice, decoding begins by measuring the *syndrome*

$$\mathbf{s} := H\mathbf{r} = H(\mathbf{x} + \mathbf{e}) = H\mathbf{e}. \tag{6.19}$$

The syndrome is invariant if the error is shifted by a codeword $\mathbf{y} \in \ker(H)$:

$$H(\mathbf{e} + \mathbf{y}) = H\mathbf{e}. \tag{6.20}$$

Errors that differ by a codeword therefore have the same syndrome. A nonzero codeword added to the error is an undetectable logical error.

A classical code with n physical bits, k logical bits, and distance d is called an $[n, k, d]$ code. The repetition code has parameters $[n, 1, n]$. We'd ideally like to have both k and d are large; that this is possible will be confirmed soon in Proposition 6.5.

6.3 The Hamming code

Example 6.4 (The seven-bit Hamming code)**:** Consider the parity-check and generator matrices

$$H_{\text{Ham}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{6.21}$$

$$G_{\text{Ham}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{6.22}$$

One checks that

$$H_{\text{Ham}} G_{\text{Ham}}^{\mathsf{T}} = 0. \tag{6.23}$$

The four rows of G_{Ham} are linearly independent, so $k = 4$. The seven columns of H_{Ham} are the seven distinct nonzero vectors in $\mathbb{F}_2^3$. Consequently no weight-one vector has zero syndrome, and no weight-two vector has zero syndrome because two distinct columns cannot sum to zero. The first row of G_{Ham} has weight three, so the distance is exactly $d = 3$. Thus the Hamming code has parameters [7,4,3].

Like in the Ising model, the Hamming codewords can be viewed as the ground states of a classical check Hamiltonian. Translating bits to spins with Eq. (6.2) gives

$$H_0(z) = -z_1 z_3 z_5 z_7 - z_2 z_3 z_6 z_7 - z_4 z_5 z_6 z_7. \tag{6.24}$$

Logical bit flips are symmetries of this Hamiltonian. For example, the first row of G_{Ham} flips z_1, z_2, z_3, under which

$$H_0 \longmapsto -(-z_1)(-z_3) z_5 z_7 - (-z_2)(-z_3) z_6 z_7 - z_4 z_5 z_6 z_7 = H_0. \tag{6.25}$$

Thus the logical operations are again symmetries of the classical check Hamiltonian. Since $d = 3$, the Hamming code corrects any single-bit error, from Proposition 6.3.

The weight-four checks of the Hamming code are useful. To store four logical bits using four independent length-three repetition codes, whose checks all have weight two, one would use

$$H_{\text{rep}}^{(3)} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \qquad H_{4\text{rep}} = \begin{pmatrix} H_{\text{rep}}^{(3)} & 0 & 0 & 0 \\ 0 & H_{\text{rep}}^{(3)} & 0 & 0 \\ 0 & 0 & H_{\text{rep}}^{(3)} & 0 \\ 0 & 0 & 0 & H_{\text{rep}}^{(3)} \end{pmatrix}. \tag{6.26}$$

The resulting parameters would be

$$[4 \cdot 3,\ 4 \cdot 1,\ 3] = [12, 4, 3]. \tag{6.27}$$

Higher-weight checks have therefore improved both k/n and d/n relative to this collection of repetition codes.

6.4 Asymptotic properties of codes as $n \to \infty$

6.4.1 Classical Hamming bound

If the check weight is allowed to increase, how large can k and d be simultaneously?

Proposition 6.5 (Hamming bound)**:** Every binary linear $[n, k, d]$ code obeys

$$\sum_{j=0}^{t} \binom{n}{j} \leq 2^{n-k}, \qquad t = \left\lfloor \frac{d-1}{2} \right\rfloor, \tag{6.28}$$

or equivalently

$$k \leq n - \log_2 \left(\sum_{j=0}^{t} \binom{n}{j} \right). \tag{6.29}$$

Proof. For a fixed syndrome $\mathbf{s}$, the equation

$$H\mathbf{r} = \mathbf{s} \tag{6.30}$$

has 2^k solutions: if $\mathbf{r}_0$ is one solution, then all solutions are $\mathbf{r}_0 + \mathbf{y}$ with $\mathbf{y} \in \ker(H)$. Since $\operatorname{rank}(H) = n - k$, there are at most 2^{n-k} possible syndromes.

The distance condition implies that every error $\mathbf{e}$ with $|\mathbf{e}| \leq t$ has a unique syndrome. Otherwise two distinct such errors would satisfy

$$H\mathbf{e}_1 = H\mathbf{e}_2 \quad \Longrightarrow \quad H(\mathbf{e}_1 + \mathbf{e}_2) = 0, \tag{6.31}$$

so $\mathbf{e}_1 + \mathbf{e}_2 \in \ker(H)$ is a codeword. But that's an issue:

$$|\mathbf{e}_1 + \mathbf{e}_2| \leq |\mathbf{e}_1| + |\mathbf{e}_2| \leq 2t < d, \tag{6.32}$$

since we'd have a codeword of Hamming weight $< d$. That's a contradiction since

$$\#\{\text{errors of weight} \leq t\} \leq \sum_{j=0}^{t} \binom{n}{j}. \tag{6.33}$$

Injecting these errors into the set of syndromes proves Eq. (6.28). □

The $[7, 4, 3]$ Hamming code saturates this bound. Here $t = 1$, and

$$2^{7-4} = 8 = \binom{7}{0} + \binom{7}{1} = 1 + 7. \tag{6.34}$$

For large n and j, the binomial coefficient has the asymptotic form

$$\binom{n}{j} = 2^{nh_2(j/n)+o(n)}, \tag{6.35}$$

where the binary entropy is

$$h_2(x) := -x \log_2 x - (1-x) \log_2(1-x). \tag{6.36}$$

Thus, we find:

Proposition 6.6: As $n \to \infty$, the Hamming bound gives schematically

$$\frac{k}{n} \leq 1 - h_2\left(\frac{d}{2n}\right) + o(1). \tag{6.37}$$

In particular, the bound permits both k/n and d/n to remain finite as $n \to \infty$.

6.4.2 LDPC codes

Codes with k/n and d/n positive as $n \to \infty$ might use checks whose weights are of order n. We will see in Section 12 that this is very undesirable – especially for a quantum code! What we'd prefer instead is a low-density parity-check code:

Definition 6.7 (Low-density parity-check code)**:** A family of parity-check matrices is *low-density* when there are constants $\Delta_C, \Delta_B = O(1)$, independent of n, such that each check contains at most Δ_C bits, and each bit participates in at most Δ_B checks. Equivalently, every row and every column of H has bounded Hamming weight.

Theorem 6.8 (Asymptotically good LDPC codes)**:** There exist families of LDPC codes with block length $n \to \infty$ for which

$$\liminf_{n\to\infty} \frac{k}{n} > 0, \qquad \liminf_{n\to\infty} \frac{d}{n} > 0. \tag{6.38}$$

Such code families are called *asymptotically good*.

Proof. This is an immediate consequence of Theorem **??**. □

6.5 Tanner graphs

A useful way to depict a classical linear code is with a *Tanner graph*. It is a bipartite graph with a circular node for every bit and a square node for every check. Bit i is connected to check a precisely

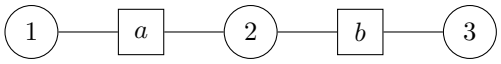

Figure 6.1: The Tanner graph of the three-bit repetition code (Example 1.1). Circles denote bits, squares denote checks, and an edge records a nonzero entry of the parity-check matrix.

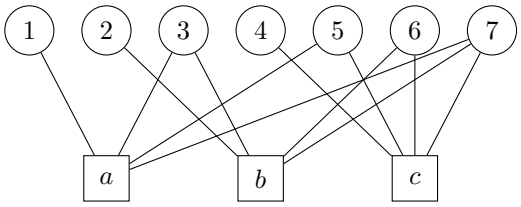

Figure 6.2: A Tanner graph for the seven-bit Hamming code (Example 6.4).

when $H_{ai} = 1$, equivalently annotated $i \in \partial a$.

Example 6.9: The three-bit repetition code (Example 1.1) has parity-check matrix

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \tag{6.39}$$

and the Tanner graph shown in Figure 6.1. The Tanner graph of the $[7, 4, 3]$ Hamming code is shown in Figure 6.2. Check nodes a, b, c correspond to the three rows of Eq. (6.21).

6.6 Redundant checks

The choice of parity-check matrix is not unique. One may add a redundant check, meaning a row that is linearly dependent on the existing rows. For example, adding the first two rows of H_{Ham} gives

$$\mathbf{h}_{\text{red}} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}, \qquad \widetilde{H}_{\text{Ham}} = \begin{pmatrix} H_{\text{Ham}} \\ \mathbf{h}_{\text{red}} \end{pmatrix}. \tag{6.40}$$

Because the new row is already in the row span,

$$\ker(\widetilde{H}_{\text{Ham}}) = \ker(H_{\text{Ham}}), \qquad \operatorname{rank}(\widetilde{H}_{\text{Ham}}) = \operatorname{rank}(H_{\text{Ham}}). \tag{6.41}$$

The codespace and the number of logical bits are unchanged, but a classical Hamiltonian containing one energy term per listed check has different energetics. Redundant checks can therefore be useful: e.g. compare the physics of the 1d Ising model (Section 4) vs. the 2d Ising model (Section 5).

6.7 Canonical forms for check and generator matrices

A parity-check matrix can be simplified by row reduction, removal of redundant rows, and permutation of columns, where a column permutation merely relabels the physical bits. After these

operations it can be written in the canonical form

$$H_{\text{can}} = \left(\mathbb{I}_{(n-k)\times(n-k)} \;\middle|\; A_{(n-k)\times k} \right), \tag{6.42}$$

where A is arbitrary. This reduction generally destroys the LDPC property, even when the original matrix was sparse.

Proposition 6.10 (Canonical generator matrix)**:** For the check matrix in Eq. (6.42), a canonical generator matrix is

$$G_{\text{can}} = \left(A^{\mathsf{T}}_{k\times(n-k)} \;\middle|\; \mathbb{I}_{k\times k} \right). \tag{6.43}$$

Its rows form a basis of $\ker(H_{\text{can}})$, so

$$\ker(H_{\text{can}}) = \operatorname{im}(G^{\mathsf{T}}_{\text{can}}). \tag{6.44}$$

Proof. First,

$$H_{\text{can}} G^{\mathsf{T}}_{\text{can}} = (\mathbb{I}_{n-k} \mid A) \begin{pmatrix} A \\ \mathbb{I}_k \end{pmatrix} = A + A = 0, \tag{6.45}$$

because the arithmetic is over $\mathbb{F}_2$. The identity blocks also imply

$$\mathbf{s}^{\mathsf{T}} H_{\text{can}} = 0 \quad \Longrightarrow \quad \mathbf{s} = 0, \tag{6.46a}$$

$$G^{\mathsf{T}}_{\text{can}} \mathbf{v} = 0 \quad \Longrightarrow \quad \mathbf{v} = 0. \tag{6.46b}$$

Hence H_{can} has rank $n-k$, while G_{can} has rank k. Since $H_{\text{can}} G^{\mathsf{T}}_{\text{can}} = 0$, one has

$$\operatorname{im}(G^{\mathsf{T}}_{\text{can}}) \subseteq \ker(H_{\text{can}}). \tag{6.47}$$

Both spaces have dimension k, so the inclusion is an equality. □

Problems

Problem 6.1 (More Hamming codes)**:** There is a natural generalization of the 7-bit Hamming code (Example 6.4): given $n = 2^r - 1$ physical bits for any $r \geq 3$, take the columns of the parity-check matrix H to be all possible non-zero vectors in $\mathbb{F}_2^r$, each included exactly once. Find k and d for this code.

7 Pauli stabilizer codes

7.1 From classical parity checks to Pauli Hamiltonians

To transition from classical to quantum code design, let us first rewrite a classical linear code in quantum language. Consider the three-bit repetition code with parity-check matrix

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}. \tag{7.1}$$

The corresponding quantum Hamiltonian is

$$H_0 = -Z_1 Z_2 - Z_2 Z_3. \tag{7.2}$$

The classical codewords are generated by

$$G = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \tag{7.3}$$

while the classical codespace is

$$C_{\mathrm{cl}} := \ker_{\mathbb{F}_2}(H) = \mathrm{im}_{\mathbb{F}_2}(G^{\mathsf{T}}). \tag{7.4}$$

They define the Hilbert-space codespace

$$\mathcal{C} := \mathrm{span}\left\{|\mathbf{x}\rangle : \mathbf{x} \in C_{\mathrm{cl}}\right\} = \mathrm{span}\{|000\rangle, |111\rangle\}. \tag{7.5}$$

This is precisely the ground-state space of (7.2). The generator of the classical code becomes the quantum symmetry

$$X_L = X_1 X_2 X_3, \qquad [H_0, X_L] = 0. \tag{7.6}$$

This is the bit-flip repetition code that we have already seen (Example 1.1).

More generally, a parity-check matrix $H \in \mathbb{F}_2^{m \times n}$ determines the commuting Hamiltonian

$$H_0 = -\sum_{c=1}^{m} \prod_{\substack{b=1 \\ H_{cb}=1}}^{n} Z_b. \tag{7.7}$$

If the rows of $G \in \mathbb{F}_2^{k \times n}$ generate $\ker(H)$, then each row gives a symmetry

$$X_{L,\alpha} := \prod_{\substack{b=1 \\ G_{\alpha b}=1}}^{n} X_b. \tag{7.8}$$

It is a symmetry because $[H_0, X_{L,\alpha}] = 0$. Indeed, the parity of the overlap between check c and

generator α is

$$\sum_{b=1}^{n} H_{cb} G_{\alpha b} = (H G^{\mathsf{T}})_{c\alpha} = 0 \qquad \text{in } \mathbb{F}_2, \tag{7.9}$$

so the corresponding Z- and X-type Pauli strings commute.

A natural generalization is to look for quantum codes defined by Hamiltonians

$$H_0 = -\sum_i S_i, \tag{7.10}$$

where each S_i is a Pauli string and they mutually commute: $[S_i, S_j] = 0$. This leads to the theory of Pauli stabilizer codes that we develop in this lecture. The Shor code (Section 2) already fits into this framework; the goal of this lecture is to build a formalism that makes the construction easy to generalize.

7.2 Groups and the Pauli group

Definition 7.1 (Group)**:** A *group* is a set G equipped with a multiplication satisfying

$$\text{closed:} \qquad g_1, g_2 \in G \Longrightarrow g_1 g_2 \in G, \tag{7.11a}$$

$$\text{identity:} \qquad \exists\, 1 \in G \text{ such that } 1g = g1 = g, \tag{7.11b}$$

$$\text{inverses:} \qquad \forall g \in G\ \exists\, g^{-1} \in G \text{ such that } g g^{-1} = g^{-1} g = 1, \tag{7.11c}$$

$$\text{associativity:} \qquad (g_1 g_2) g_3 = g_1 (g_2 g_3). \tag{7.11d}$$

Definition 7.2 (Abelian group)**:** A group is *Abelian* if

$$g_1 g_2 = g_2 g_1 \qquad \text{for every } g_1, g_2 \in G. \tag{7.12}$$

Definition 7.3 (Pauli group)**:** The n-qubit Pauli group is

$$\mathcal{P}_n := \{\omega P_1 \otimes \cdots \otimes P_n : \omega \in \{+1, -1, +\mathrm{i}, -\mathrm{i}\},\ P_b \in \{\mathbb{I}, X, Y, Z\}\}\,. \tag{7.13}$$

The Pauli group is non-Abelian. On one qubit,

$$X^2 = Y^2 = Z^2 = \mathbb{I}, \qquad \mathbb{I}X = X\mathbb{I}, \qquad XZ = -\mathrm{i}Y, \qquad ZX = \mathrm{i}Y, \tag{7.14}$$

and analogous relations hold for the other pairs. On different qubits the operators commute. Consequently, any two Pauli strings either commute or anticommute:

$$PQ = \pm QP \qquad \text{for every } P, Q \in \mathcal{P}_n. \tag{7.15}$$

Definition 7.4 (Pauli stabilizer code)**:** A *Pauli stabilizer code* is a quantum code whose codespace is

$$\mathcal{C} = \left\{ |\psi\rangle \in (\mathbb{C}^2)^{\otimes n} : S\,|\psi\rangle = |\psi\rangle \ \text{ for every } S \in \mathcal{S} \right\}, \tag{7.16}$$

where $\mathcal{S} \leq \mathcal{P}_n$ is a subgroup called the *stabilizer group*.

A key question will be how to choose $\mathcal{S}$ so that the resulting code has useful error-correcting properties. We begin with two elementary constraints that follow directly from the definition.

Proposition 7.5 (The stabilizer group is Abelian)**:** If $\mathcal{C} \neq \{0\}$, then every pair of elements of $\mathcal{S}$ commutes.

Proof. Take $S_1, S_2 \in \mathcal{S}$ and a nonzero $|\psi\rangle \in \mathcal{C}$. Both operators stabilize the codeword, so

$$S_1 S_2\,|\psi\rangle = |\psi\rangle = S_2 S_1\,|\psi\rangle\,. \tag{7.17}$$

Any two Pauli strings commute or anticommute. If they anticommuted, then Eq. (7.17) would imply

$$|\psi\rangle = S_1 S_2\,|\psi\rangle = -S_2 S_1\,|\psi\rangle = -\,|\psi\rangle\,, \tag{7.18}$$

which is impossible for a nonzero state. Therefore $S_1 S_2 = S_2 S_1$. □

Proposition 7.6 (Nontrivial phases are not stabilizers)**:** If $S \in \mathcal{S}$, then

$$-S \notin \mathcal{S}, \qquad +\mathrm{i}S \notin \mathcal{S}, \qquad -\mathrm{i}S \notin \mathcal{S}. \tag{7.19}$$

Proof. For every $|\psi\rangle \in \mathcal{C}$, one has

$$S\,|\psi\rangle = |\psi\rangle\,, \qquad (-S)\,|\psi\rangle = -\,|\psi\rangle\,, \qquad (\pm\mathrm{i}S)\,|\psi\rangle = \pm\mathrm{i}\,|\psi\rangle\,. \tag{7.20}$$

Only the first operator fixes the codeword with eigenvalue $+1$. □

In particular, every stabilizer element is Hermitian and obeys

$$S^\dagger = S, \qquad S^2 = \mathbb{I}. \tag{7.21}$$

The classical codes at the beginning of the lecture fit neatly into this framework: their Z-type parity checks generate an Abelian Pauli stabilizer group.

7.3 Independent generators and the dimension of the codespace

Let $S_1, \ldots, S_m$ be independent generators of $\mathcal{S}$. Thus every stabilizer has a unique expression as a product of the generators, and no nonempty product of distinct generators equals the identity:

$$\prod_{j \in A} S_j = \mathbb{I} \qquad \Longrightarrow \qquad A = \varnothing. \tag{7.22}$$

Consequently,

$$\mathcal{S} = \left\{ \prod_{j=1}^{m} S_j^{a_j} : a_j \in \{0, 1\} \right\}, \qquad |\mathcal{S}| = 2^m. \tag{7.23}$$

Proposition 7.7 (Dimension of a stabilizer code)**:** The joint $+1$ eigenspace of m independent commuting stabilizer generators has dimension

$$\dim(\mathcal{C}) = 2^{n-m} =: 2^k, \qquad k = n - m. \tag{7.24}$$

The integer k is the number of logical qubits.

Proof. The projector onto the joint $+1$ eigenspace is

$$P_{\mathcal{C}} = \prod_{j=1}^{m} \frac{\mathbb{I} + S_j}{2} = \frac{1}{2^m} \sum_{S \in \mathcal{S}} S. \tag{7.25}$$

Every nonidentity Pauli string has vanishing trace, while

$$\operatorname{tr}(\mathbb{I}) = 2^n. \tag{7.26}$$

Independence guarantees that the identity appears exactly once in the sum in (7.25). Therefore

$$\dim(\mathcal{C}) = \operatorname{tr}(P_{\mathcal{C}}) = \frac{1}{2^m} \sum_{S \in \mathcal{S}} \operatorname{tr}(S) = \frac{2^n}{2^m} = 2^{n-m}, \tag{7.27}$$

which implies (7.24). □

We take $m < n$, so (as we will see) at least one logical qubit is encoded; however, all the formulas we develop here allow $m = n$, which gives a one-dimensional stabilizer-state codespace with $k = 0$. This stores no logical information.

For a syndrome

$$\boldsymbol{\eta} = (\eta_1, \ldots, \eta_m) \in \{+1, -1\}^m, \tag{7.28}$$

define the simultaneous eigenspace and its projector by

$$\mathcal{C}_{\boldsymbol{\eta}} := \{|\psi\rangle : S_j |\psi\rangle = \eta_j |\psi\rangle \text{ for every } j\}, \tag{7.29a}$$

$$P_{\boldsymbol{\eta}} := \prod_{j=1}^{m} \frac{\mathbb{I} + \eta_j S_j}{2}. \tag{7.29b}$$

The projectors are mutually orthogonal and resolve the identity:

$$P_{\boldsymbol{\eta}} P_{\boldsymbol{\eta}'} = \delta_{\boldsymbol{\eta},\boldsymbol{\eta}'} P_{\boldsymbol{\eta}}, \tag{7.30a}$$

$$\sum_{\boldsymbol{\eta}\in\{\pm1\}^m} P_{\boldsymbol{\eta}} = \mathbb{I}. \tag{7.30b}$$

Corollary 7.8 (Dimension of every syndrome sector)**:** Every syndrome sector has dimension

$$\dim(\mathcal{C}_{\boldsymbol{\eta}}) = 2^{n-m} = 2^k. \tag{7.31}$$

Proof. Expanding Eq. (7.29b) gives a signed sum of all elements of $\mathcal{S}$. As in the proof of Proposition 7.7, only the identity has nonzero trace, and its coefficient remains 2^{-m}. Hence

$$\operatorname{tr}(P_{\boldsymbol{\eta}}) = 2^{-m} \operatorname{tr}(\mathbb{I}) = 2^{n-m}. \tag{7.32}$$

Since $\operatorname{tr}(P_{\boldsymbol{\eta}}) = \dim(\mathcal{C}_{\boldsymbol{\eta}})$ this leads to (7.31). □

Proposition 7.9 (Syndrome of a Pauli error)**:** Let $E \in \mathcal{P}_n$, and define signs $\eta_j \in \{+1, -1\}$ by

$$S_j E = \eta_j E S_j. \tag{7.33}$$

If $|\psi\rangle \in \mathcal{C}$, then

$$S_j(E|\psi\rangle) = \eta_j(E|\psi\rangle) \qquad \text{for every } j. \tag{7.34}$$

Thus $E|\psi\rangle \in \mathcal{C}_{\boldsymbol{\eta}}$.

Proof. This is analogous to Proposition 2.2. Since $S_j|\psi\rangle = |\psi\rangle$, $S_j E|\psi\rangle = \eta_j E S_j|\psi\rangle = \eta_j E|\psi\rangle$. □

The preceding corollary shows that every formal syndrome sector is nonzero. For completeness, we now prove an additional fact used below: every such sector is reached from the codespace by an actual Pauli operator.

Lemma 7.10 (Every syndrome has a Pauli representative)**:** For every $\boldsymbol{\eta} \in \{+1, -1\}^m$, there exists a phase-free Pauli string $E_{\boldsymbol{\eta}} \in \{\mathbb{I}, X, Y, Z\}^{\otimes n}$ such that

$$S_j E_{\boldsymbol{\eta}} S_j = \eta_j E_{\boldsymbol{\eta}} \qquad \text{for every } j. \tag{7.35}$$

Moreover,

$$E_{\boldsymbol{\eta}}\mathcal{C} = \mathcal{C}_{\boldsymbol{\eta}}. \tag{7.36}$$

Proof. Choose nonzero states

$$|c\rangle \in \mathcal{C}, \qquad |\boldsymbol{\eta}\rangle \in \mathcal{C}_{\boldsymbol{\eta}}, \tag{7.37}$$

and define the nonzero operator

$$A := |\boldsymbol{\eta}\rangle\langle c| \,. \tag{7.38}$$

Since every stabilizer generator fixes $|c\rangle$ and has eigenvalue η_j on $|\boldsymbol{\eta}\rangle$,

$$S_j A S_j = \eta_j A \qquad \text{for every } j. \tag{7.39}$$

Expand A in the phase-free Pauli operator basis:

$$A = \sum_{P \in \{I,X,Y,Z\}^{\otimes n}} a_P P. \tag{7.40}$$

Conjugation of a Pauli by a Pauli only changes its sign, so for every j and P there is a sign $\chi_j(P) \in \{+1, -1\}$ such that

$$S_j P S_j = \chi_j(P) P. \tag{7.41}$$

Combining Eqs. (7.39) and (7.40), and using linear independence of the Pauli basis, gives

$$a_P \neq 0 \qquad \Longrightarrow \qquad \chi_j(P) = \eta_j \quad \text{for every } j. \tag{7.42}$$

Because $A \neq 0$, at least one coefficient is nonzero. Choose one such Pauli and call it $E_{\boldsymbol{\eta}}$. It obeys Eq. (7.35). Proposition 7.9 then gives

$$E_{\boldsymbol{\eta}}\mathcal{C} \subseteq \mathcal{C}_{\boldsymbol{\eta}}. \tag{7.43}$$

The Pauli operator is unitary, so the space on the left has dimension 2^k. Corollary 7.8 gives the same dimension for the space on the right. The inclusion is therefore an equality. □

7.4 Syndrome measurement and a candidate recovery channel

The commuting observables $S_1, \ldots, S_m$ may be measured in any order. Their outcomes give the syndrome $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_m)$, while preserving all logical information inside the corresponding 2^k-dimensional sector.

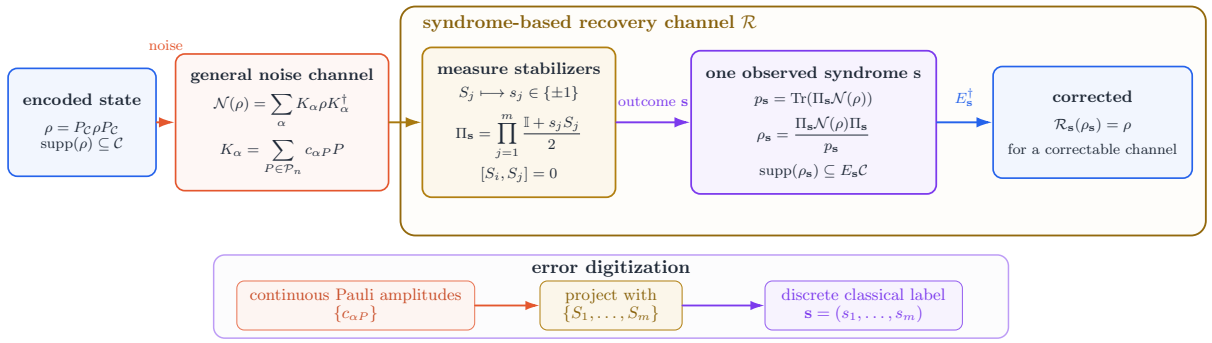

Figure 7.1: As in Section 3, measuring the Pauli stabilizers digitizes the error into a Pauli error, which can be corrected by applying an appropriate choice of $E_{\boldsymbol{\eta}}^\dagger$.

Choose one Pauli representative $E_{\boldsymbol{\eta}}$ for every syndrome. After observing $\boldsymbol{\eta}$, one applies $E_{\boldsymbol{\eta}}^\dagger$. The resulting candidate recovery channel is

$$\mathcal{R}(\rho) := \sum_{\boldsymbol{\eta} \in \{\pm 1\}^m} E_{\boldsymbol{\eta}}^\dagger P_{\boldsymbol{\eta}} \rho P_{\boldsymbol{\eta}} E_{\boldsymbol{\eta}}. \tag{7.44}$$

This placement of the syndrome projectors makes the measurement-and-correction order explicit. Equivalently, since $P_{\boldsymbol{\eta}} = E_{\boldsymbol{\eta}} P_{\mathcal{C}} E_{\boldsymbol{\eta}}^\dagger$, one may write

$$\mathcal{R}(\rho) = \sum_{\boldsymbol{\eta}} P_{\mathcal{C}} E_{\boldsymbol{\eta}}^\dagger \rho E_{\boldsymbol{\eta}} P_{\mathcal{C}}. \tag{7.45}$$

The Kraus operators

$$K_{\boldsymbol{\eta}} := E_{\boldsymbol{\eta}}^\dagger P_{\boldsymbol{\eta}} \tag{7.46}$$

satisfy

$$\sum_{\boldsymbol{\eta}} K_{\boldsymbol{\eta}}^\dagger K_{\boldsymbol{\eta}} = \sum_{\boldsymbol{\eta}} P_{\boldsymbol{\eta}} = \mathbb{I}, \tag{7.47}$$

so $\mathcal{R}$ is a quantum channel. This is illustrated in Figure 7.1.

Suppose an actual Pauli error E has syndrome $\boldsymbol{\eta}$ and $\rho = P_{\mathcal{C}} \rho P_{\mathcal{C}}$. The candidate recovery gives

$$\mathcal{R}(E \rho E^\dagger) = E_{\boldsymbol{\eta}}^\dagger E \rho E^\dagger E_{\boldsymbol{\eta}}. \tag{7.48}$$

Thus syndrome recovery always returns the state to the codespace, but it corrects the logical state only when $E_{\boldsymbol{\eta}}^\dagger E$ acts trivially on the code. We now classify exactly when this happens.

A Pauli error is *detectable by the stabilizer measurement* if at least one syndrome bit is negative:

$$\eta_j = -1 \text{ for at least one } j \qquad \Longleftrightarrow \qquad \{E, S_j\} = 0 \text{ for at least one } j. \tag{7.49}$$

It has zero syndrome if

$$\eta_j = +1 \text{ for every } j \qquad \Longleftrightarrow \qquad [E, S_j] = 0 \text{ for every } j. \tag{7.50}$$

A zero-syndrome operator is undetectable by the stabilizer measurements, but it need not be a harmful logical error. It may instead be a global phase times a stabilizer, in which case its action on the encoded state is trivial. The normalizer formalism below separates these two possibilities.

7.5 The normalizer and logical Pauli operators

Define the Pauli normalizer of the stabilizer group by

$$N(\mathcal{S}) := \{E \in \mathcal{P}_n : [E, S] = 0 \text{ for every } S \in \mathcal{S}\}. \tag{7.51}$$

It is exactly the set of zero-syndrome Pauli operators.

Strictly speaking, Eq. (7.51) defines the centralizer of $\mathcal{S}$ inside $\mathcal{P}_n$. It equals the group-theoretic Pauli normalizer: conjugating one Pauli by another gives $ESE^\dagger = \pm S$, and $-S \notin \mathcal{S}$ by Proposition 7.6. Hence conjugation preserves $\mathcal{S}$ precisely when the operators commute.

Proposition 7.11 (Number of Pauli operators with a fixed syndrome)**:** For every syndrome $\boldsymbol{\eta}$,

$$|\{E \in \mathcal{P}_n : S_j E S_j = \eta_j E \text{ for every } j\}| = 4 \cdot 2^{n+k}. \tag{7.52}$$

In particular,

$$|N(\mathcal{S})| = 4 \cdot 2^{n+k}. \tag{7.53}$$

Proof. There are four choices of global phase and four phase-free Pauli choices on each qubit, so

$$|\mathcal{P}_n| = 4 \cdot 4^n. \tag{7.54}$$

Let $E_{\boldsymbol{\eta}}$ be one representative of syndrome $\boldsymbol{\eta}$. Multiplication by a zero-syndrome Pauli preserves the syndrome, and if F has syndrome $\boldsymbol{\eta}$, then $E_{\boldsymbol{\eta}}^\dagger F$ has zero syndrome. Therefore the full fiber is the coset

$$\{E \in \mathcal{P}_n : E \text{ has syndrome } \boldsymbol{\eta}\} = E_{\boldsymbol{\eta}} N(\mathcal{S}). \tag{7.55}$$

All fibers consequently have cardinality $|N(\mathcal{S})|$. By Lemma 7.10, all $2^m = 2^{n-k}$ syndromes occur. Hence

$$|N(\mathcal{S})| = \frac{|\mathcal{P}_n|}{2^{n-k}} = \frac{4 \cdot 4^n}{2^{n-k}} \tag{7.56}$$

which implies (7.53), and therefore (7.52). □

The count has the useful factorization

$$4 \cdot 2^{n+k} = \underbrace{4}_{\substack{\text{global}\\ \text{phases}}} \times \underbrace{2^{n-k}}_{\text{stabilizers}} \times \underbrace{2^{2k}}_{\substack{\text{logical Pauli}\\ \text{classes}}} . \tag{7.57}$$

Indeed, if $Q \in N(\mathcal{S})$, $S \in \mathcal{S}$, and $|\psi\rangle \in \mathcal{C}$, then

$$QS\,|\psi\rangle = Q\,|\psi\rangle\,. \tag{7.58}$$

Thus multiplying by a stabilizer does not change the encoded action. Global phases are also physically irrelevant because

$$|\psi\rangle \sim e^{i\phi}\,|\psi\rangle\,, \qquad e^{i\phi}\rho e^{-i\phi} = \rho. \tag{7.59}$$

Let

$$\Phi_{\text{ph}} := \{\mathbb{I}, -\mathbb{I}, +i\mathbb{I}, -i\mathbb{I}\} \tag{7.60}$$

be the phase subgroup. The logical Pauli group is the quotient

$$\mathcal{P}_{\text{logical}} := N(\mathcal{S})/(\Phi_{\text{ph}}\mathcal{S}), \tag{7.61}$$

and its cardinality is

$$|\mathcal{P}_{\text{logical}}| = \frac{4 \cdot 2^{n+k}}{4 \cdot 2^{n-k}} = 2^{2k} = 4^k. \tag{7.62}$$

The $4^k = (2^k)^2$ quotient classes are the logical Pauli operators modulo stabilizers and global phases. Choosing one representative of each class gives an operator basis on the 2^k-dimensional codespace. This is a basis of the operator space; of course there are infinitely many linear combinations of such operators that act as valid logical operators.

7.6 Stabilizer equivalence and the Knill–Laflamme condition

Definition 7.12 (Equivalent Pauli errors)**:** Two Pauli errors $E_1, E_2 \in \mathcal{P}_n$ are *equivalent on the code* if they have the same action on every code state up to an overall phase:

$$E_1 P_{\mathcal{C}} = \omega E_2 P_{\mathcal{C}} \tag{7.63}$$

for some

$$\omega \in \{+1, -1, +\mathrm{i}, -\mathrm{i}\} =: \Phi_{\text{ph}}. \tag{7.64}$$

Proposition 7.13 (Stabilizer characterization of equivalent errors)**:** Two Pauli errors are equivalent on

the code if and only if for some $S \in \mathcal{S}$ and $\omega \in \Phi_{\mathrm{ph}}$,

$$E_1 = \omega E_2 S. \tag{7.65}$$

Equivalently, $E_2^\dagger E_1 \in \Phi_{\mathrm{ph}} \mathcal{S}$.

Proof. If Eq. (7.65) holds, then

$$E_1 P_{\mathcal{C}} = \omega E_2 S P_{\mathcal{C}} = \omega E_2 P_{\mathcal{C}}, \tag{7.66}$$

so the two errors are equivalent.

Conversely, suppose Eq. (7.63) holds and set

$$Q := \omega^{-1} E_2^\dagger E_1 \in \mathcal{P}_n. \tag{7.67}$$

Then

$$Q P_{\mathcal{C}} = P_{\mathcal{C}}. \tag{7.68}$$

Using the stabilizer expansion of the projector gives

$$Q P_{\mathcal{C}} = \frac{1}{2^{n-k}} \sum_{S \in \mathcal{S}} QS = \frac{1}{2^{n-k}} \sum_{S \in \mathcal{S}} S = P_{\mathcal{C}}. \tag{7.69}$$

The phase-free Pauli matrices are linearly independent. The Pauli terms on the two sides of Eq. (7.69) must therefore match. In particular, $QS_0 = S_1$ for some $S_0, S_1 \in \mathcal{S}$, and hence

$$Q = S_1 S_0^{-1} \in \mathcal{S}. \tag{7.70}$$

Substituting this into Eq. (7.67) proves Eq. (7.65). □

The relation to the Knill–Laflamme condition is especially simple for Pauli errors. Let

$$Q := E_a^\dagger E_b. \tag{7.71}$$

If $Q \notin N(\mathcal{S})$, then it anticommutes with some stabilizer S, and

$$P_{\mathcal{C}} Q P_{\mathcal{C}} = P_{\mathcal{C}} S Q P_{\mathcal{C}} = -P_{\mathcal{C}} Q S P_{\mathcal{C}} = -P_{\mathcal{C}} Q P_{\mathcal{C}}, \tag{7.72}$$

so

$$P_{\mathcal{C}} Q P_{\mathcal{C}} = 0. \tag{7.73}$$

If $Q = \omega S \in \Phi_{\mathrm{ph}}\mathcal{S}$, then

$$P_{\mathcal{C}} Q P_{\mathcal{C}} = \omega P_{\mathcal{C}}. \tag{7.74}$$

The only forbidden case is

$$Q \in N(\mathcal{S}) \setminus (\Phi_{\mathrm{ph}}\mathcal{S}), \tag{7.75}$$

where the two errors have the same syndrome but differ by a nontrivial logical Pauli. Thus a set of Pauli errors is correctable precisely when this third case never occurs for any pair.

Operationally, after observing a syndrome, the decoder must choose a Pauli to apply. Multiplication of that choice by a stabilizer is harmless,

$$E \longmapsto ES, \qquad S \in \mathcal{S}, \tag{7.76}$$

whereas multiplication by a nontrivial logical Pauli is harmful:

$$E \longmapsto EL, \qquad L \in N(\mathcal{S}) \setminus (\Phi_{\mathrm{ph}}\mathcal{S}). \tag{7.77}$$

7.7 Code distance

The weight of a Pauli string is the number of qubits on which it acts nontrivially:

$$\mathrm{wt}(E) := \#\left\{b : E_b \neq I\right\}. \tag{7.78}$$

Definition 7.14 (Distance of a stabilizer code)**:** The code distance is the smallest weight of a nontrivial logical Pauli:

$$d := \min\left\{\mathrm{wt}(L) : L \in N(\mathcal{S}) \setminus (\Phi_{\mathrm{ph}}\mathcal{S})\right\}. \tag{7.79}$$

Proposition 7.15 (Correction radius from the distance)**:** A stabilizer code of distance d corrects every Pauli error of weight at most

$$t := \left\lfloor \frac{d-1}{2} \right\rfloor. \tag{7.80}$$

Proof. Let E_1, E_2 be two Pauli errors with $\mathrm{wt}(E_1), \mathrm{wt}(E_2) \leq t$, and set

$$Q := E_1^{\dagger} E_2. \tag{7.81}$$

Its weight obeys

$$\mathrm{wt}(Q) \leq \mathrm{wt}(E_1) + \mathrm{wt}(E_2) \leq 2t < d. \tag{7.82}$$

By the definition of d, Q cannot belong to $N(\mathcal{S}) \setminus (\Phi_{\mathrm{ph}}\mathcal{S})$. There are only two possibilities.

If $Q \notin N(\mathcal{S})$, then the errors have different syndromes and Eq. (7.73) gives

$$P_{\mathcal{C}} E_1^{\dagger} E_2 P_{\mathcal{C}} = 0. \tag{7.83}$$

If $Q \in N(\mathcal{S})$, then the weight bound forces $Q \in \Phi_{\mathrm{ph}}\mathcal{S}$, so

$$P_{\mathcal{C}} E_1^{\dagger} E_2 P_{\mathcal{C}} = \omega_{12} P_{\mathcal{C}} \tag{7.84}$$

for some phase ω_{12}. In either case the Knill–Laflamme condition holds for all errors of weight at most t, proving the claim. □

The first part of this proof is identical to the classical distance argument: two sufficiently small errors cannot differ by a logical operator. The new quantum feature is that errors may differ by a stabilizer and still be perfectly correctable. The decoder only needs to identify an error up to stabilizer equivalence.

7.8 The Shor code revisited

Let us briefly revisit the Shor code to see how the abstract statements fit together. Its stabilizer group is generated by the eight operators given in (2.38). They are independent, so the codespace encodes $k = 9 - 8 = 1$ logical qubits. Up to global phase, the normalizer is generated by $\mathcal{S}$ together with the logical operators

$$X_L = X_1 X_2 X_3, \qquad Z_L = Z_1 Z_4 Z_7. \tag{7.85}$$

Logical representatives are not unique. For example, multiplying X_L by the first X-type stabilizer gives

$$(X_1 X_2 X_3)(X_1 X_2 X_3 X_4 X_5 X_6) = X_4 X_5 X_6, \tag{7.86}$$

so $X_1X_2X_3$ and $X_4X_5X_6$ represent the same logical Pauli. By checking the size of all the stabilizer-equivalent logical Paulis in $N(\mathcal{S}) \setminus \mathcal{S}$, one can show that the Shor code has $d = 3$, and therefore corrects every single-qubit Pauli error, as we claimed in Section 2.

Not all single-qubit errors need to be distinguishable. Since $Z_1Z_2 \in \mathcal{S}$,

$$Z_1(Z_1 Z_2) = Z_2, \qquad Z_1 P_{\mathcal{C}} = Z_2 P_{\mathcal{C}}. \tag{7.87}$$

Thus Z_1 and Z_2 have the same syndrome and the same action on every encoded state. The same correction representative works for both errors, which is precisely stabilizer degeneracy in the language developed above.

Problems

Problem 7.1 (Quantum Hamming bound)**:** Consider a general quantum Pauli stabilizer code.

A: Suppose that we want a code with a distance $d = 2t + 1$; namely, the code must correct every error on $\leq t$ bits. Find a lower bound on the number of unique errors we must be able to distinguish, assuming that every Pauli error of weight $\leq t$ maps to a unique syndrome, and use this to deduce a lower bound on n as a function of k and t. Your result will be the quantum Hamming bound – the generalization of the classical Hamming bound (6.29) to quantum codes.

B: Suppose we want codes that, as $n \to \infty$, have a fixed ratio $r = k/n$. We'd like the ratio $d/n = a$ to be as large as possible. Find an upper bound on a using the result of **A**.

8 CSS codes

Today we will describe Pauli stabilizer codes using a linear-algebra formalism much like the one used for classical parity-check codes. Recall that a Pauli stabilizer code has codespace

$$\mathcal{C} = \{|\psi\rangle : S|\psi\rangle = |\psi\rangle \text{ for every } S \in \mathcal{S}\}, \qquad \mathcal{S} \leq \mathcal{P}_n, \qquad -\mathbb{I} \notin \mathcal{S}, \tag{8.1}$$

where the stabilizer group is generated by commuting Pauli operators. For error correction, the central question is which Pauli error gives which syndrome, or equivalently which commutation relations it has with the stabilizers. These data admit a simple binary description.

8.1 Binary representation of Pauli operators

Definition 8.1 (Binary Pauli vector)**:** For one qubit, define a map

$$\varphi : \mathcal{P}_1 \longrightarrow \mathbb{F}_2^2 \tag{8.2}$$

by

$$\varphi(\mathbb{I}) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \qquad \varphi(X) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \varphi(Y) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \varphi(Z) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{8.3}$$

The map ignores global phases:

$$\varphi(\omega Q) = \varphi(Q) \qquad \text{for } \omega \in \{+1, -1, +\mathrm{i}, -\mathrm{i}\}. \tag{8.4}$$

For n qubits, write

$$\varphi(Q) = \begin{pmatrix} \mathbf{q}^{(x)} \\ \mathbf{q}^{(z)} \end{pmatrix} \in \mathbb{F}_2^{2n}, \qquad \mathbf{q}^{(x)}, \mathbf{q}^{(z)} \in \mathbb{F}_2^n, \tag{8.5}$$

where the first n bits record the locations of X factors and the last n bits record the locations of Z factors. A Y contributes one to both blocks.

For example, on two qubits,

$$\varphi(X_1) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad \varphi(Y_1 Z_2) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}. \tag{8.6}$$

Proposition 8.2 (Pauli multiplication becomes binary addition)**:** For all $Q_1, Q_2 \in \mathcal{P}_n$,

$$\varphi(Q_1 Q_2) = \varphi(Q_1) + \varphi(Q_2), \tag{8.7}$$

where the addition on the right is in $\mathbb{F}_2^{2n}$.

The proposition follows by checking the one-qubit Pauli multiplication table. Thus, after quotienting by global phases, φ is a group isomorphism from Pauli multiplication to binary-vector addition.

Define the $2n \times 2n$ matrix

$$J := \begin{pmatrix} 0_{n\times n} & \mathbb{I}_{n\times n} \\ \mathbb{I}_{n\times n} & 0_{n\times n} \end{pmatrix}. \tag{8.8}$$

Proposition 8.3 (Binary commutation test)**:** For $Q_1, Q_2 \in \mathcal{P}_n$,

$$[Q_1, Q_2] = 0 \quad \Longleftrightarrow \quad \varphi(Q_1)^{\mathsf{T}} J \varphi(Q_2) = 0, \tag{8.9a}$$

$$\{Q_1, Q_2\} = 0 \quad \Longleftrightarrow \quad \varphi(Q_1)^{\mathsf{T}} J \varphi(Q_2) = 1. \tag{8.9b}$$

Proof. If

$$\varphi(Q_a) = \begin{pmatrix} \mathbf{q}_a^{(x)} \\ \mathbf{q}_a^{(z)} \end{pmatrix}, \qquad a \in \{1, 2\}, \tag{8.10}$$

then

$$\varphi(Q_1)^{\mathsf{T}} J \varphi(Q_2) = \mathbf{q}_1^{(x)} \cdot \mathbf{q}_2^{(z)} + \mathbf{q}_1^{(z)} \cdot \mathbf{q}_2^{(x)}. \tag{8.11}$$

This expression counts modulo two the qubits on which the two Pauli strings locally anticommute. An odd number of local anticommutations gives an overall minus sign, while an even number gives an overall plus sign. This proves Eqs. (8.9a) and (8.9b). □

8.2 Stabilizer codes as binary linear codes

Choose stabilizer generators $S_1, \ldots, S_m$ and collect their binary vectors as the rows of the parity-check matrix

$$H := \begin{pmatrix} \varphi(S_1)^{\mathsf{T}} \\ \vdots \\ \varphi(S_m)^{\mathsf{T}} \end{pmatrix} \in \mathbb{F}_2^{m\times 2n}. \tag{8.12}$$

Every entry of HJH^{T} is the binary commutation pairing of two stabilizer generators. Therefore the stabilizer commutation condition is

$$HJH^{\mathsf{T}} = 0. \tag{8.13}$$

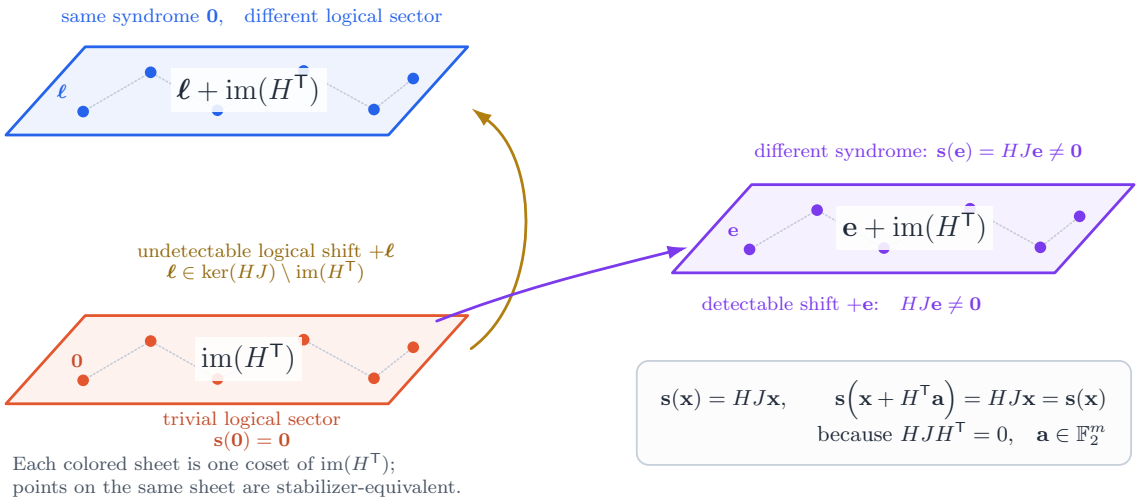

Figure 8.1: Illustration of different syndrome cosets (and logical cosets) in a general Pauli stabilizer code.

The Pauli operators commuting with every stabilizer are represented by

$$\varphi(N(\mathcal{S})) = \ker(HJ) = \left\{\mathbf{v} \in \mathbb{F}_2^{2n} : HJ\mathbf{v} = 0\right\}. \tag{8.14}$$

Meanwhile, products of stabilizer generators are represented by

$$\mathrm{im}(H^{\mathsf{T}}) = \left\{H^{\mathsf{T}}\mathbf{a} : \mathbf{a} \in \mathbb{F}_2^m\right\}. \tag{8.15}$$

Consequently, binary logical Pauli operators are the quotient $\ker(HJ)/\mathrm{im}(H^{\mathsf{T}})$. This is the binary version of identifying Pauli operators that differ by a stabilizer. More generally, all Pauli errors modulo stabilizer equivalence form the error space $\frac{\mathbb{F}_2^{2n}}{\mathrm{im}(H^{\mathsf{T}})}$. If an error has binary vector $\mathbf{e}$, then its binary syndrome is

$$\mathbf{s} = HJ\mathbf{e}. \tag{8.16}$$

Equation (8.13) implies that the syndrome is independent of the chosen stabilizer-equivalent representative:

$$HJ(\mathbf{e} + H^{\mathsf{T}}\mathbf{a}) = HJ\mathbf{e} + HJH^{\mathsf{T}}\mathbf{a} = HJ\mathbf{e}. \tag{8.17}$$

The measured eigenvalue of generator S_a is $(-1)^{s_a}$. Thus the binary syndrome contains exactly the same information as the usual list of stabilizer eigenvalues: see Figure 8.1.

Definition 8.4 (Pauli Hamming weight and reduced weight)**:** For $\mathbf{w} = (\mathbf{w}^{(x)}, \mathbf{w}^{(z)}) \in \mathbb{F}_2^{2n}$, its Pauli

Hamming weight is

$$|\mathbf{w}|_{\mathrm{H}} := \sum_{j=1}^{n} \begin{cases} 1, & (w_j^{(x)}, w_j^{(z)}) \neq (0,0), \\ 0, & (w_j^{(x)}, w_j^{(z)}) = (0,0). \end{cases} \tag{8.18}$$

Its reduced error weight is the smallest weight among all stabilizer-equivalent representatives:

$$|\mathbf{w}|_{\mathrm{red}} := \min_{\mathbf{v} \in \mathrm{im}(H^{\mathsf{T}})} |\mathbf{w} + \mathbf{v}|_{\mathrm{H}}. \tag{8.19}$$

Definition 8.5 (Distance of a stabilizer code)**:** The code distance is

$$d := \min_{\mathbf{w} \in \ker(HJ) \setminus \mathrm{im}(H^{\mathsf{T}})} |\mathbf{w}|_{\mathrm{red}}. \tag{8.20}$$

Thus d is the minimum physical weight of a nontrivial logical Pauli, after optimizing over stabilizer-equivalent representatives.

8.3 The special case of CSS codes

We will mainly study stabilizer codes of a special form.

Definition 8.6 (CSS code)**:** A CSS code has a binary parity-check matrix

$$H = \begin{pmatrix} H_X & 0 \\ 0 & H_Z \end{pmatrix}, \qquad H_X \in \mathbb{F}_2^{m_X \times n}, \qquad H_Z \in \mathbb{F}_2^{m_Z \times n}. \tag{8.21}$$

Thus every stabilizer generator is either purely X-type or purely Z-type.

Example 8.7 (The Shor code)**:** For the nine-qubit Shor code, one may take

$$H_X = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{8.22a}$$

$$H_Z = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}. \tag{8.22b}$$

The first row of H_X, for example, represents $X_1X_2X_3X_4X_5X_6$, while the first row of H_Z represents Z_1Z_2.

8.3.1 Useful properties of CSS codes

Proposition 8.8 (CSS commutation condition)**:** The X- and Z-type stabilizers commute if and only if

$$H_X H_Z^{\mathsf{T}} = 0. \tag{8.23}$$

Proof. For the block form in Eq. (8.21),

$$HJH^{\mathsf{T}} = \begin{pmatrix} H_X & 0 \\ 0 & H_Z \end{pmatrix} \begin{pmatrix} 0 & \mathbb{I} \\ \mathbb{I} & 0 \end{pmatrix} \begin{pmatrix} H_X^{\mathsf{T}} & 0 \\ 0 & H_Z^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} 0 & H_X H_Z^{\mathsf{T}} \\ H_Z H_X^{\mathsf{T}} & 0 \end{pmatrix}. \tag{8.24}$$

The lower-left block is the transpose of the upper-right block, so Eq. (8.13) is equivalent to Eq. (8.23). □

For a CSS code,

$$HJ = \begin{pmatrix} 0 & H_X \\ H_Z & 0 \end{pmatrix}. \tag{8.25}$$

Therefore the X and Z parts of a logical operator obey separate kernel conditions. After quotienting by stabilizers, the logical-vector spaces are

$$\mathcal{L}_X := \frac{\ker(H_Z)}{\mathrm{im}(H_X^{\mathsf{T}})}, \tag{8.26a}$$

$$\mathcal{L}_Z := \frac{\ker(H_X)}{\mathrm{im}(H_Z^{\mathsf{T}})}. \tag{8.26b}$$

It is convenient to introduce the classical codes

$$C_X := \ker(H_X), \qquad C_Z := \ker(H_Z). \tag{8.27}$$

Since

$$C_X^{\perp} = \mathrm{im}(H_X^{\mathsf{T}}), \qquad C_Z^{\perp} = \mathrm{im}(H_Z^{\mathsf{T}}), \tag{8.28}$$

we may equivalently write

$$\mathcal{L}_X = C_Z / C_X^{\perp}, \qquad \mathcal{L}_Z = C_X / C_Z^{\perp}. \tag{8.29}$$

The commutation condition gives the inclusions $C_X^{\perp} \subseteq C_Z$ and $C_Z^{\perp} \subseteq C_X$ needed for these quotients.

A generator matrix $\widetilde{G}_X$ for C_Z and a generator matrix $\widetilde{G}_Z$ for C_X may be chosen so that

$$\mathrm{im}(\widetilde{G}_X^{\mathsf{T}}) = C_Z, \qquad \mathrm{im}(\widetilde{G}_Z^{\mathsf{T}}) = C_X. \tag{8.30}$$

For the Shor code, one convenient choice is

$$\widetilde{G}_X = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}. \tag{8.31}$$

Only one of these three directions is an independent logical direction. We may take

$$G_X = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{8.32}$$

so that

$$\ker(H_Z) = \mathrm{im}(G_X^{\mathsf{T}}) \oplus \mathrm{im}(H_X^{\mathsf{T}}). \tag{8.33}$$

Here the symbol $\oplus$ means that the vector space $\ker(H_Z)$ is generated by vectors either from $\mathrm{im}(G_X^{\mathsf{T}})$ or $\mathrm{im}(H_X^{\mathsf{T}})$. For example,

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}, \tag{8.34}$$

where the first vector on the right is the chosen logical generator and the second is an X-type stabilizer. Similarly, a Z-logical generator is

$$G_Z = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}. \tag{8.35}$$

We can compare these answers with what we found previously in Section 2; they agree.

8.3.2 Logical operators

Proposition 8.9 (Number of encoded qubits)**:** The CSS codespace has dimension 2^k, where

$$k := \dim \ker(H_X) - \dim \mathrm{im}(H_Z^{\mathsf{T}}) = \dim \ker(H_Z) - \dim \mathrm{im}(H_X^{\mathsf{T}}). \tag{8.36}$$

Equivalently, if

$$m := \mathrm{rank}(H_X) + \mathrm{rank}(H_Z) \tag{8.37}$$

is the number of independent stabilizers, then $k = n - m$.

Proof. Rank–nullity gives

$$2^n = |\ker(H_X)|\,|\mathrm{im}(H_X^{\mathsf{T}})| = |\ker(H_Z)|\,|\mathrm{im}(H_Z^{\mathsf{T}})|. \tag{8.38}$$

Therefore

$$\frac{|\ker(H_X)|}{|\mathrm{im}(H_Z^{\mathsf{T}})|} = \frac{|\ker(H_Z)|}{|\mathrm{im}(H_X^{\mathsf{T}})|} = \frac{2^n}{|\mathrm{im}(H_X^{\mathsf{T}})||\mathrm{im}(H_Z^{\mathsf{T}})|} = 2^{n-m} = 2^k. \tag{8.39}$$

The two quotients in Eqs. (8.26a) and (8.26b) consequently both have dimension k. As in the preceding lecture, m independent stabilizers leave a joint $+1$ eigenspace of dimension $2^{n-m} = 2^k$. □

Proposition 8.10 (Canonical commutation pairing for logical generators)**:** One can choose logical generator matrices

$$G_X, G_Z \in \mathbb{F}_2^{k\times n} \tag{8.40}$$

whose rows represent bases of $\mathcal{L}_X$ and $\mathcal{L}_Z$, respectively, such that

$$G_X G_Z^{\mathsf{T}} = \mathbb{I}_{k\times k}. \tag{8.41}$$

Thus the ath logical X anticommutes with the ath logical Z and commutes with the other logical generators.

Proof. First note that

$$\ker(H_X) = \mathrm{im}(H_X^{\mathsf{T}})^{\perp}, \qquad \ker(H_Z) = \mathrm{im}(H_Z^{\mathsf{T}})^{\perp}. \tag{8.42}$$

Let $\mathbf{v}_Z \in \ker(H_X)$ represent a Z-logical class and let $\mathbf{v}_X \in \ker(H_Z)$ represent an X-logical class. If we replace them by stabilizer-equivalent representatives, then

$$\begin{aligned}&(\mathbf{v}_Z + H_Z^{\mathsf{T}}\mathbf{s}_Z)^{\mathsf{T}}(\mathbf{v}_X + H_X^{\mathsf{T}}\mathbf{s}_X)\\ &\quad = \mathbf{v}_Z^{\mathsf{T}}\mathbf{v}_X + \mathbf{s}_Z^{\mathsf{T}}H_Z\mathbf{v}_X + \mathbf{v}_Z^{\mathsf{T}}H_X^{\mathsf{T}}\mathbf{s}_X + \mathbf{s}_Z^{\mathsf{T}}H_Z H_X^{\mathsf{T}}\mathbf{s}_X = \mathbf{v}_Z^{\mathsf{T}}\mathbf{v}_X.\end{aligned} \tag{8.43}$$

Here the middle terms vanish because $H_Z\mathbf{v}_X = 0$ and $H_X\mathbf{v}_Z = 0$, while the final term vanishes by Eq. (8.23). The dot product therefore defines a pairing

$$\mathcal{L}_X \times \mathcal{L}_Z \longrightarrow \mathbb{F}_2, \qquad ([\mathbf{v}_X], [\mathbf{v}_Z]) \longmapsto \mathbf{v}_X^{\mathsf{T}}\mathbf{v}_Z. \tag{8.44}$$

This pairing is nondegenerate. Indeed, if $\mathbf{v}_Z^{\mathsf{T}}\mathbf{v}_X = 0$ for every $\mathbf{v}_X \in \ker(H_Z)$, then

$$\mathbf{v}_Z \in \ker(H_Z)^{\perp} = \mathrm{im}(H_Z^{\mathsf{T}}), \tag{8.45}$$

so $\mathbf{v}_Z$ represents the trivial Z-logical class. The same argument with X and Z exchanged proves nondegeneracy in the other slot.

Here is the explicit Gaussian-elimination step. Choose arbitrary matrices $\widetilde{G}_X, \widetilde{G}_Z \in \mathbb{F}_2^{k\times n}$ whose

rows represent bases of $\mathcal{L}_X$ and $\mathcal{L}_Z$, and form their pairing matrix

$$M := \widetilde{G}_X \widetilde{G}_Z^{\mathsf{T}} \in \mathbb{F}_2^{k\times k}. \tag{8.46}$$

The matrix M must be invertible. Otherwise there would be a nonzero coefficient vector $\mathbf{a} \in \mathbb{F}_2^k$ with $\mathbf{a}^{\mathsf{T}} M = 0$. The nontrivial logical class represented by $\mathbf{a}^{\mathsf{T}} \widetilde{G}_X$ would then pair to zero with every row of $\widetilde{G}_Z$, and hence with every Z-logical class, contradicting nondegeneracy. Row-reducing the invertible matrix M to the identity is equivalent to replacing the X-logical basis by

$$G_X := M^{-1} \widetilde{G}_X, \qquad G_Z := \widetilde{G}_Z. \tag{8.47}$$

Then

$$G_X G_Z^{\mathsf{T}} = M^{-1} \widetilde{G}_X \widetilde{G}_Z^{\mathsf{T}} = M^{-1} M = \mathbb{I}_{k\times k}, \tag{8.48}$$

which is Eq. (8.41). □

The noncommuting logical X and Z operators are essential: they allow us to manipulate the encoded qubits rather than merely preserve them. For the Shor code, (8.32) and (8.35) give

$$X_L = X_1 X_2 X_3, \qquad Z_L = Z_1 Z_4 Z_7. \tag{8.49}$$

Their binary representatives have dot product one, so they anticommute. Because the pairing is independent of stabilizer representatives, any stabilizer-equivalent representatives also anticommute.

Definition 8.11 (X and Z distances)**:** For a CSS code, define

$$d_X := \min_{\mathbf{v}\in \ker(H_Z)\setminus \mathrm{im}(H_X^{\mathsf{T}})} |\mathbf{v}|, \tag{8.50a}$$

$$d_Z := \min_{\mathbf{v}\in \ker(H_X)\setminus \mathrm{im}(H_Z^{\mathsf{T}})} |\mathbf{v}|. \tag{8.50b}$$

The full code distance is

$$d = \min(d_X, d_Z). \tag{8.51}$$

Finding some logical generator matrices G_X and G_Z is a straightforward linear-algebra problem. Finding the smallest-weight nontrivial logical operator, and hence determining d_X or d_Z, is generally much harder numerically.

For quantum codes, we write a code with n physical qubits, k logical qubits, and code distance d as an $[[n, k, d]]$ code.

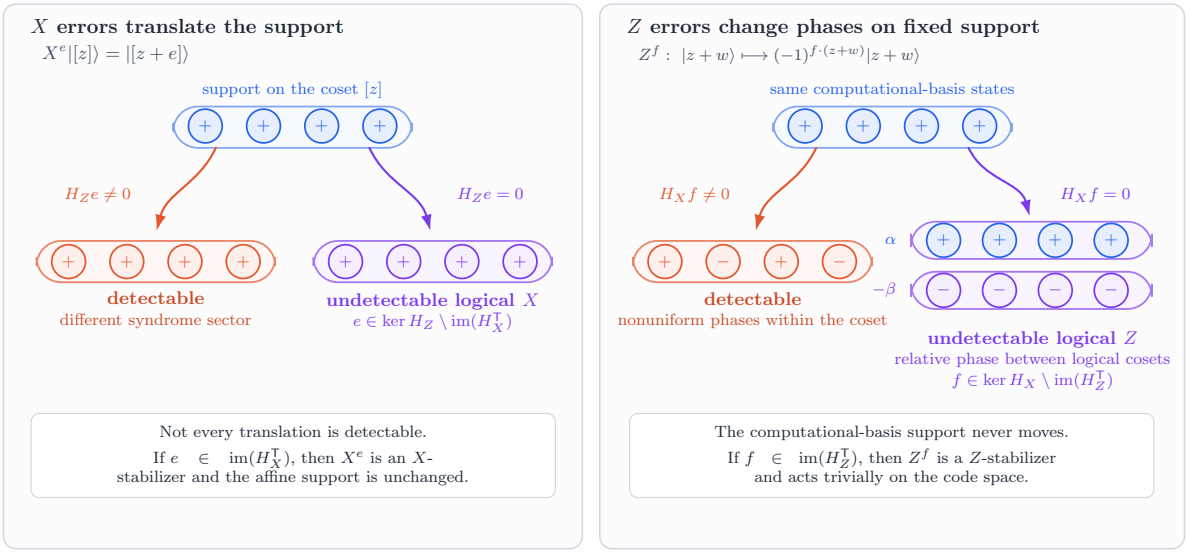

Figure 8.2: Action of X vs. Z errors on (8.53).

8.3.3 Logical codewords as coset states

We have now developed enough formalism to begin studying practical quantum codes. The key will be to use the binary formalism rather than manipulate Pauli strings directly. Nevertheless, it is sometimes useful to think about the wave functions of their codewords.

For $\mathbf{z} \in \mathbb{F}_2^n$, choose the computational basis so that

$$Z_j |\mathbf{z}\rangle = (1 - 2z_j) |\mathbf{z}\rangle = (-1)^{z_j} |\mathbf{z}\rangle . \tag{8.52}$$

For a coset $[\mathbf{z}] \in C_Z/C_X^\perp = \ker(H_Z)/\mathrm{im}(H_X^\mathsf{T})$, define the logical codeword

$$|[\mathbf{z}]\rangle := \frac{1}{\sqrt{|\mathrm{im}(H_X^\mathsf{T})|}} \sum_{\mathbf{w}\in\mathrm{im}(H_X^\mathsf{T})} |\mathbf{z}+\mathbf{w}\rangle . \tag{8.53}$$

There are 2^k such cosets and hence 2^k logical basis states.

The stabilizer conditions are transparent in this representation. An X-type stabilizer translates every summation variable by an element of $\mathrm{im}(H_X^\mathsf{T})$ and therefore only permutes the terms in Eq. (8.53). A Z-type stabilizer labeled by $\mathbf{b} \in \mathrm{im}(H_Z^\mathsf{T})$ contributes the phase

$$(-1)^{\mathbf{b}^\mathsf{T}(\mathbf{z}+\mathbf{w})} = 1, \tag{8.54}$$

because $\mathbf{z} \in \ker(H_Z)$ and $\mathbf{w} \in \mathrm{im}(H_X^\mathsf{T}) \subseteq \ker(H_Z)$. Distinct logical cosets are disjoint, so the states in Eq. (8.53) are orthonormal.

As illustrated in Figure 8.2, an X error translates the entire coset. If its binary vector $\mathbf{e}$ obeys

$H_Z\mathbf{e} \neq 0$, it moves the state into a detectable syndrome sector. If instead

$$\mathbf{e} \in \ker(H_Z) \setminus \mathrm{im}(H_X^\mathsf{T}), \tag{8.55}$$

it moves one logical coset to a different logical coset and is an undetectable logical X error.

A Z error leaves the computational-basis labels in the same coset but changes their phases. For a binary vector $\mathbf{f}$,

$$Z^{\mathbf{f}} \left|[\mathbf{z}]\right\rangle = \frac{1}{\sqrt{|\mathrm{im}(H_X^\mathsf{T})|}} \sum_{\mathbf{w}\in\mathrm{im}(H_X^\mathsf{T})} (-1)^{\mathbf{f}^\mathsf{T}(\mathbf{z}+\mathbf{w})} \left|\mathbf{z}+\mathbf{w}\right\rangle. \tag{8.56}$$

If $H_X\mathbf{f} \neq 0$, these phases vary along at least one X-stabilizer direction and the error is detectable. If

$$\mathbf{f} \in \ker(H_X) \setminus \mathrm{im}(H_Z^\mathsf{T}), \tag{8.57}$$

the phase is cosetwise constant but logical-label dependent: an undetectable logical Z error.

Problems

Problem 8.1 (Quantum Hamming codes)**:** Take the parity-check matrix H that you derived in Problem 6.1 and show that you can make a quantum CSS code with $H_X = H_Z = H$. These are called **quantum Hamming codes**. Find k and d for this code.

Problem 8.2 (Smallest quantum code)**:** Show that if you want a quantum code with $k = 1$ that can correct arbitrary single-qubit errors, the quantum Hamming bound (Problem 7.1) shows that $n \geq 5$ physical qubits are required. An explicit example of such a code with $n = 5$ has the four stabilizers

$$S_1 = X_1Z_2Z_3X_4, \tag{8.58a}$$
$$S_2 = X_2Z_3Z_4X_5, \tag{8.58b}$$
$$S_3 = X_1X_3Z_4Z_5, \tag{8.58c}$$
$$S_4 = Z_1X_2X_4Z_5. \tag{8.58d}$$

Write the 4×10 parity-check matrix H over $\mathbb{F}_2$ for this code. Then, find (one choice of) smallest-weight logical X and Z operators for this code. Explain how you can use the measured syndrome to correct arbitrary single-qubit errors on any physical qubit: in particular, provide a table showing which syndrome corresponds to which correctable single-qubit error.

9 2d toric code

9.1 The toric code on a square lattice

We are now ready to introduce a quantum analogue of the one-dimensional Ising model. The toric code will be a workhorse for our study of quantum codes. Consider an $L \times L$ square lattice with periodic boundary conditions, so that the lattice forms a two-dimensional torus. Let V, E, and F denote its sets of vertices, edges, and faces, respectively. We place one physical qubit on every edge $e \in E$: see Figure 9.1.

For every vertex $v \in V$, define an X-type check supported on the four edges incident to v. For every face $f \in F$, define a Z-type check supported on the four edges in the boundary of f:

$$X^{\mathrm{d}v} := \prod_{e \ni v} X_e, \tag{9.1a}$$

$$Z^{\partial f} := \prod_{e \in \partial f} Z_e. \tag{9.1b}$$

The support of $X^{\mathrm{d}v}$ and the support of $Z^{\partial f}$ overlap on either zero or two edges. Since $X_e Z_e = -Z_e X_e$ on a shared edge, the two possible minus signs cancel whenever the supports overlap. Therefore

$$[X^{\mathrm{d}v}, Z^{\partial f}] = 0 \qquad \text{for every } v \in V \text{ and } f \in F. \tag{9.2}$$

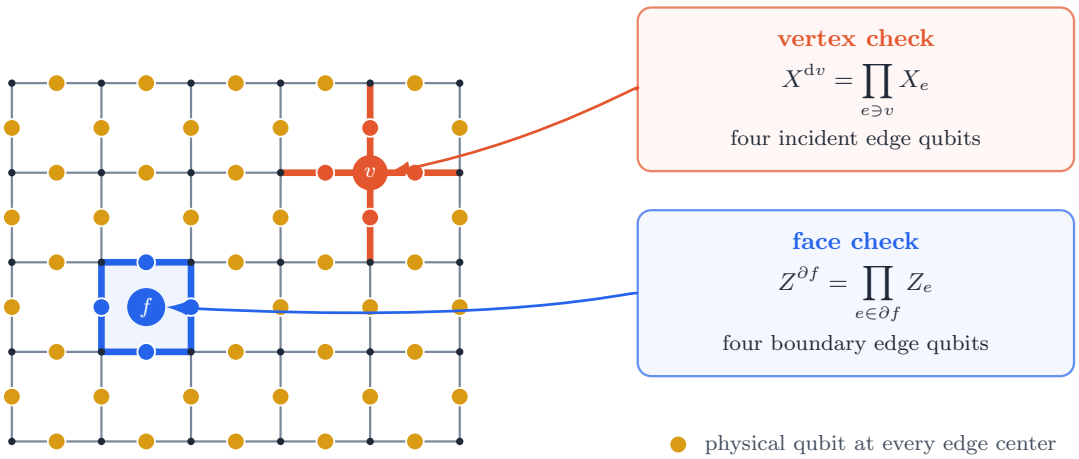

Figure 9.1: Setup of X- and Z-checks in the toric code.

The toric-code codespace is the simultaneous +1 eigenspace of all these commuting checks:

$$\mathcal{C}_{\mathrm{TC}} = \left\{ |\psi\rangle : X^{\mathrm{d}v} |\psi\rangle = |\psi\rangle \ \forall v, \quad Z^{\partial f} |\psi\rangle = |\psi\rangle \ \forall f \right\}. \tag{9.3}$$

Two key advantages of the toric code are its geometric locality in two spatial dimensions and its scalable distance, $d \sim \sqrt{n}$. The checks are local on the torus. A literal periodic identification is not local in a planar device, however, and we will later replace the torus by a planar surface with suitable boundaries.

9.2 Discrete vector calculus and the parity-check matrices

Before studying error correction in two-dimensional codes, we will spend some time understanding where this construction comes from. The basic mathematics is a discrete version of vector calculus.

Choose orientations for the edges and faces. A function G_v on vertices is a discrete 0-form, a vector field u_e on edges is a discrete 1-form, and a dual function ϕ_f on faces is a discrete 2-form. Over the real numbers, the discrete gradient and curl are linear maps

$$\mathrm{d}_0 : \mathbb{R}^V \longrightarrow \mathbb{R}^E, \qquad u = \mathrm{d}_0 G \sim \nabla G, \tag{9.4a}$$

$$\mathrm{d}_1 : \mathbb{R}^E \longrightarrow \mathbb{R}^F, \qquad \phi = \mathrm{d}_1 u \sim \partial_x u_y - \partial_y u_x. \tag{9.4b}$$

The familiar identity that the curl of a gradient vanishes becomes

$$\mathrm{d}_1 \mathrm{d}_0 = 0. \tag{9.5}$$

For one consistent orientation convention, the matrix elements of these maps are

$$\langle e|\mathrm{d}_0|v\rangle = \begin{cases} +1, & e \text{ is oriented away from } v, \\ -1, & e \text{ is oriented toward } v, \\ 0, & e \text{ is not incident to } v, \end{cases} \tag{9.6a}$$

$$\langle f|\mathrm{d}_1|e\rangle = \begin{cases} +1, & e \subset \partial f \text{ and its orientation agrees with } \partial f, \\ -1, & e \subset \partial f \text{ and its orientation opposes } \partial f, \\ 0, & e \not\subset \partial f. \end{cases} \tag{9.6b}$$

See Figure 9.2 for an illustration.

These integer-valued matrices may be reduced modulo two. Over $\mathbb{F}_2$ the signs disappear, and the matrix elements simply record incidence:

$$\langle e|\mathrm{d}_0|v\rangle = \begin{cases} 1, & e \text{ is incident to } v, \\ 0, & \text{otherwise}, \end{cases} \tag{9.7a}$$

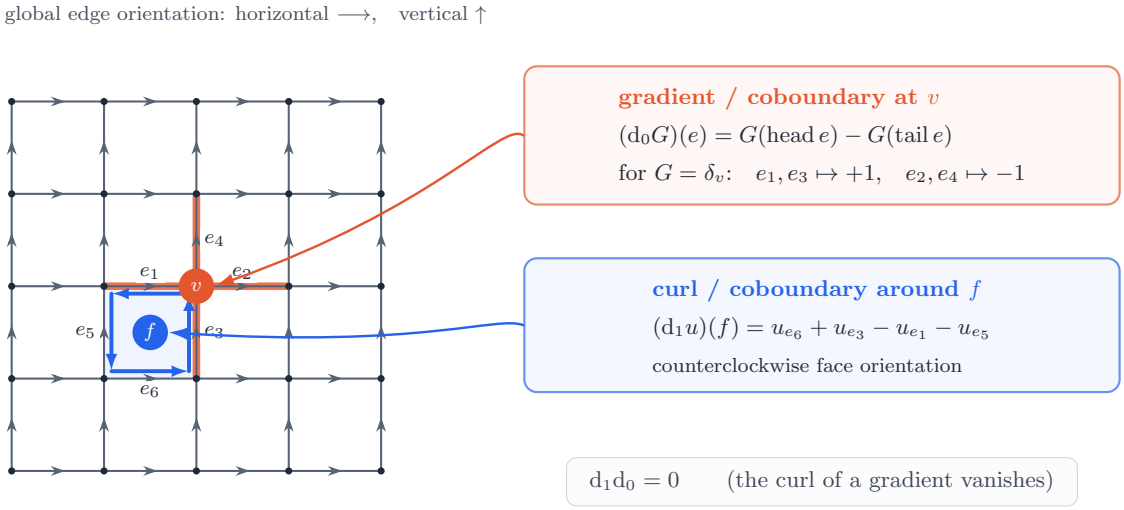

Figure 9.2: Illustration of d_0 and d_1 as discretizations of gradient and curl on a two-dimensional square lattice.

$$\langle f|\mathrm{d}_1|e\rangle = \begin{cases} 1, & e \subset \partial f, \\ 0, & \text{otherwise.} \end{cases} \tag{9.7b}$$

Vertices label X-checks, edges label qubits, and faces label Z-checks. Consequently,

$$\mathrm{d}_0 = H_X^{\mathsf{T}}, \tag{9.8a}$$

$$\mathrm{d}_1 = H_Z, \tag{9.8b}$$

$$H_Z H_X^{\mathsf{T}} = \mathrm{d}_1\mathrm{d}_0 = 0. \tag{9.8c}$$

Thus the elementary vector-calculus identity "curl grad equals zero" is precisely the CSS commutation condition.

9.3 Cohomology and logical X operators

Recall that the X-type logical operators of a CSS code are parameterized by

$$\mathcal{L}_X = \frac{\ker(H_Z)}{\mathrm{im}(H_X^{\mathsf{T}})} = \frac{\ker(\mathrm{d}_1)}{\mathrm{im}(\mathrm{d}_0)}. \tag{9.9}$$

This quotient is a standard object in mathematics.

Definition 9.1 (Cochain complex)**:** A cochain complex over $\mathbb{F}_2$ is a sequence of vector spaces and linear maps

$$V_0 \xrightarrow{\mathrm{d}_0} V_1 \xrightarrow{\mathrm{d}_1} V_2 \tag{9.10}$$

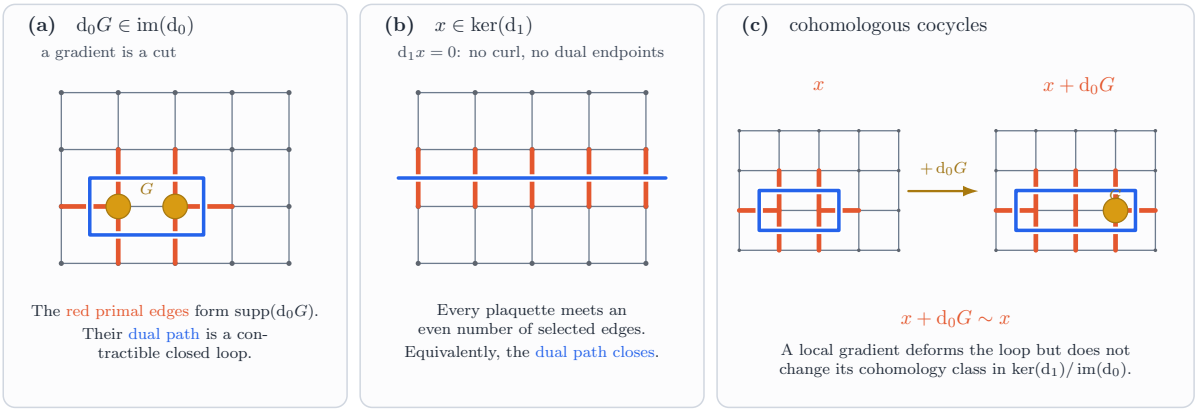

Figure 9.3: Illustration of elements of $\ker(\mathrm{d}_1)$ and $\mathrm{im}(\mathrm{d}_0)$.

that obey $\mathrm{d}_1 \mathrm{d}_0 = 0$.

Definition 9.2 (First cohomology group)**:** The first cohomology group of the complex in Eq. (9.10) is

$$\mathrm{H}^1 := \frac{\ker(\mathrm{d}_1)}{\mathrm{im}(\mathrm{d}_0)}. \tag{9.11}$$

A vector $x \in \mathbb{F}_2^E$ may be identified with its support, namely the subset of edges on which $x_e = 1$. Addition of vectors corresponds to symmetric difference of their supports. The condition $x \in \ker(\mathrm{d}_1)$ says that every face boundary contains an even number of selected edges. Equivalently, when the selected edges are drawn on the dual lattice, they form a union of closed loops. These are the discrete vector fields with no curl.

An element of $\mathrm{im}(\mathrm{d}_0)$ is a discrete gradient. In the code, it is the support of a product of vertex stabilizers. On the dual lattice, such a support is a boundary. Therefore two curl-free configurations x_1 and x_2 represent the same cohomology class exactly when

$$x_1 \sim x_2 \quad \Longleftrightarrow \quad x_1 = x_2 + \mathrm{d}_0 G \quad \text{for some } G \in V_0. \tag{9.12}$$

The quotient H^1 is therefore the space of stabilizer-equivalence classes of undetectable X operators: see Figure 9.3.

If $\dim \mathrm{H}^1 = k$, then there are 2^k distinct cohomology classes. The dimension, rather than the cardinality of the quotient, is the number of independent logical X generators and hence the number of encoded qubits.

Theorem 9.3 (First cohomology of the torus)**:** For the square lattice with periodic boundary conditions,

$$\mathrm{H}^1(T^2; \mathbb{F}_2) \cong \mathbb{F}_2^2. \tag{9.13}$$

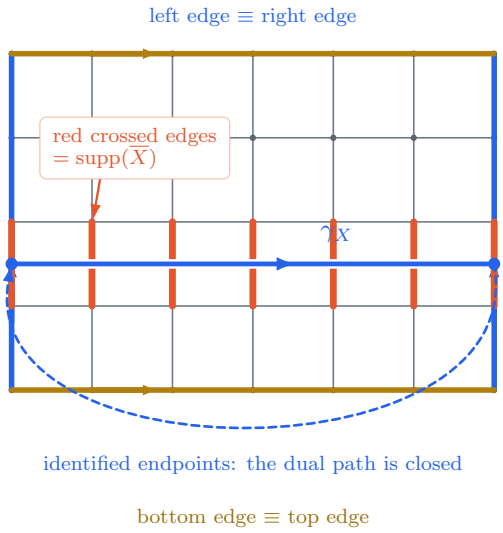

Figure 9.4: An X-logical, i.e. non-trivial element of H^1, is a cocycle that wraps around the periodic boundary conditions on the torus.

In particular, the toric code encodes $k = 2$ logical qubits.

Proof. See Figure 9.4. A curl-free dual-lattice loop can be deformed by adding boundaries, which are precisely the elements of $\mathrm{im}(\mathrm{d}_0)$. Contractible loops can therefore be removed. What remains is characterized by two winding parities: whether the configuration winds around the two independent cycles of the torus. A horizontal winding loop and a vertical winding loop generate these two independent classes; their sum gives the fourth class. □

On an $L \times L$ lattice there are L^2 horizontal edges and L^2 vertical edges, so

$$n = |E| = 2L^2. \tag{9.14}$$

The shortest nontrivial dual-lattice loop has length L. Consequently,

$$d_X = L = \sqrt{\frac{n}{2}}. \tag{9.15}$$

There is also a useful algebraic check on $k = 2$. Every edge appears twice in the product of all vertex checks and twice in the product of all face checks, so

$$\prod_{v \in V} X^{\mathrm{d}v} = \mathbb{I}, \tag{9.16a}$$

$$\prod_{f \in F} Z^{\partial f} = \mathbb{I}. \tag{9.16b}$$

For the connected square torus these are the only relations of their respective types, so $\operatorname{rank} H_X =$

$\operatorname{rank} H_Z = L^2 - 1$ and

$$k = n - \operatorname{rank} H_X - \operatorname{rank} H_Z = 2. \tag{9.17}$$

9.4 Homology and logical Z operators

There is a parallel description of the Z-type logical operators. Reverse the cochain complex and transpose its maps.

Definition 9.4 (Chain complex)**:** The chain complex dual to Eq. (9.10) is

$$V_2 \xrightarrow{\partial_2} V_1 \xrightarrow{\partial_1} V_0, \qquad \partial_{j+1} := \mathrm{d}_j^{\mathsf{T}}. \tag{9.18}$$

It obeys

$$\partial_1 \partial_2 = 0. \tag{9.19}$$

Definition 9.5 (First homology group)**:** The first homology group is

$$\mathrm{H}_1 := \frac{\ker(\partial_1)}{\operatorname{im}(\partial_2)}. \tag{9.20}$$

A vector $z \in V_1 \cong \mathbb{F}_2^E$ is a string on the original lattice. The boundary $\partial_1 z$ is the set of endpoints of that string. In the code, if e_Z is the support vector of a Z error, then its X-syndrome is

$$s_X = H_X e_Z = \partial_1 e_Z. \tag{9.21}$$

Thus an open string has nonzero syndrome at its endpoints, whereas a closed string lies in $\ker(\partial_1)$. The image $\operatorname{im}(\partial_2)$ consists of boundaries of collections of faces; these are products of face stabilizers. Two closed strings are homologous when they differ by such a boundary: see Figure 9.5. It follows that the Z-logical space is

$$\mathcal{L}_Z = \frac{\ker(H_X)}{\operatorname{im}(H_Z^{\mathsf{T}})} = \frac{\ker(\partial_1)}{\operatorname{im}(\partial_2)} = \mathrm{H}_1. \tag{9.22}$$

As proved by linear algebra in the previous lecture, the dual chain and cochain complexes satisfy

$$\dim \mathrm{H}^1 = \dim \mathrm{H}_1. \tag{9.23}$$

Thus there is one independent logical Z generator for every independent logical X generator.

More precisely, logical commutation defines a natural perfect pairing

$$\mathrm{H}^1 \times \mathrm{H}_1 \longrightarrow \mathbb{F}_2, \qquad ([x], [z]) \longmapsto x^{\mathsf{T}} z. \tag{9.24}$$

It is well defined because adding an exact cocycle or a boundary does not change the overlap. Hence

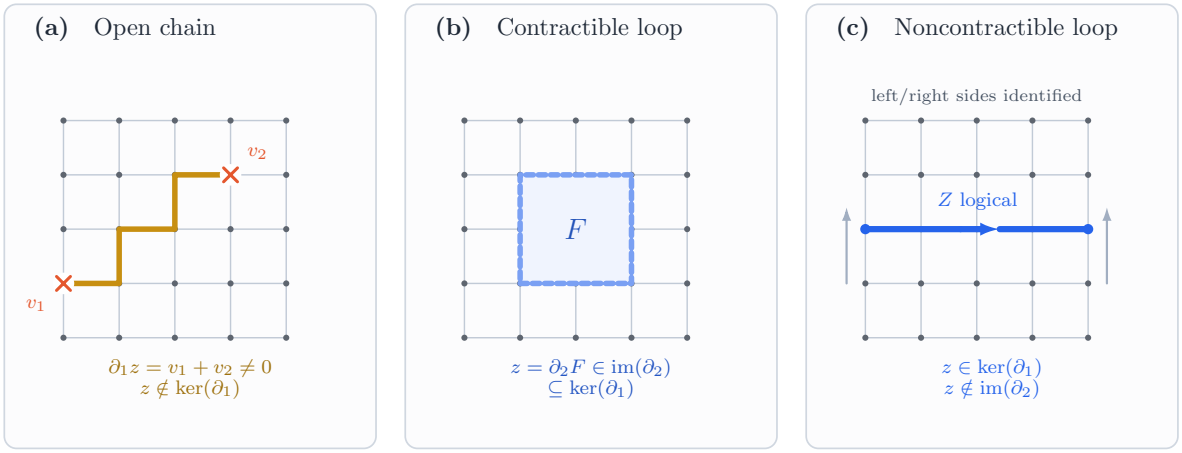

Figure 9.5: Illustration of elements of $\mathrm{im}(\partial_2)$ and $\ker(\partial_1)$.

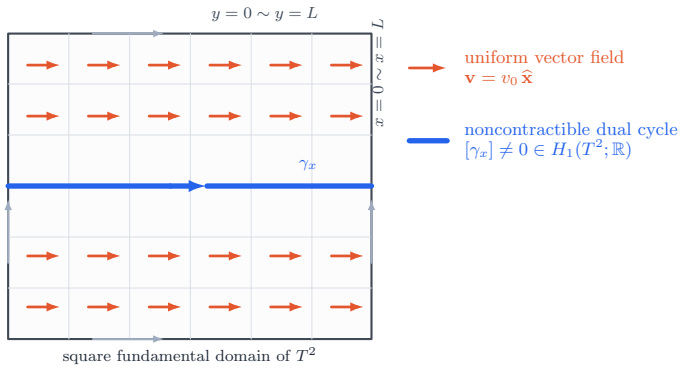

Figure 9.6: An illustration of a natural pairing between homology and cohomology on the torus. Note that this illustration is for real-valued vector fields on a continuum torus, vs. discrete $\mathbb{F}_2$ on the discretized torus.

the canonical statement is $\mathrm{H}^1 \cong \mathrm{H}_1^*$. Choosing dual bases of logical operators gives

$$G_X G_Z^{\mathsf{T}} = \mathbb{I}_{k \times k}, \tag{9.25}$$

which is the binary form of the canonical Pauli commutation relations. This is the topological interpretation of what we did in the proof of Proposition 8.10.

9.5 Topology and code distance

The distinction between closed and exact fields is topological. On a continuum torus, for example, a uniform horizontal real-valued vector field is curl-free but is not the gradient of a globally defined, single-valued function: the would-be coordinate x is not globally defined on the torus: see Figure 9.6. This is a real-valued analogy; the code itself uses coefficients in $\mathbb{F}_2$.

For the CSS code obtained from a cellulation, the cochain and chain complexes are

$$\mathbb{F}_2^{m_X} \xrightarrow{H_X^\mathsf{T}=\mathrm{d}_0} \mathbb{F}_2^n \xrightarrow{H_Z=\mathrm{d}_1} \mathbb{F}_2^{m_Z}, \tag{9.26a}$$

$$\mathbb{F}_2^{m_Z} \xrightarrow{H_Z^\mathsf{T}=\partial_2} \mathbb{F}_2^n \xrightarrow{H_X=\partial_1} \mathbb{F}_2^{m_X}. \tag{9.26b}$$

The logical spaces are therefore

$$\mathcal{L}_X = \mathrm{H}^1, \tag{9.27a}$$

$$\mathcal{L}_Z = \mathrm{H}_1. \tag{9.27b}$$

For the square torus, the shortest homologically nontrivial primal and dual cycles both have length L. Hence

$$d = d_X = d_Z = L = \sqrt{\frac{n}{2}}, \tag{9.28}$$

so the toric code has parameters $[[2L^2, 2, L]]$. For a general cellulation, d_Z is the length of the shortest homologically nontrivial cycle on the primal lattice, while d_X is the corresponding quantity on the dual lattice. They need not be equal when the cellulation is not self-dual.

9.6 Codes on general surfaces

The same construction works for any cellulation of a closed two-dimensional surface: put qubits on edges, X-checks on vertices, and Z-checks on faces. The associated chain complex automatically guarantees $H_Z H_X^\mathsf{T} = 0$. The resulting logical operators are determined by the homology and cohomology of the surface, which is why these are called *topological codes*.

Theorem 9.6 (Homology of a genus-g surface)**:** Let Σ_g be a closed, connected, orientable surface with g handles. Then

$$\mathrm{H}_1(\Sigma_g; \mathbb{F}_2) \cong \mathbb{F}_2^{2g}, \tag{9.29a}$$

$$\mathrm{H}^1(\Sigma_g; \mathbb{F}_2) \cong \mathbb{F}_2^{2g}. \tag{9.29b}$$

Consequently, the corresponding surface code encodes

$$k = 2g \tag{9.30}$$

logical qubits.

Each handle contributes two independent nontrivial cycles, conventionally called an a-cycle and a b-cycle: see an illustration in Figure 9.7. In experiment it is difficult to realize literal handles or periodic identifications in a planar architecture. We will soon see how suitable boundaries allow one to store logical states in a planar surface code in Section 11.

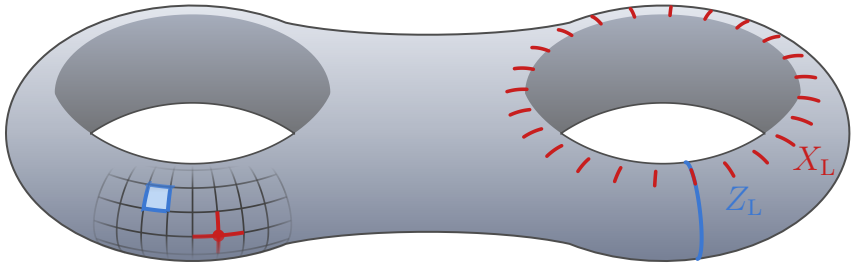

Figure 9.7: Illustration of a toric code tiling of a genus-2 surface. One of the 4 pairs of logicals are depicted.

Finally, note that it is not possible to make a square lattice *globally* fit on a shape with topology other than the torus – this follows from techniques analogous to Lemma 11.2. However, since the boundary maps ∂_1 and ∂_2 are well-defined for any triangulation of a surface, we can still define a toric code on any genus-g surface.

9.7 Quantum LDPC codes and Tanner graphs

One more important property, especially for the later theory of fault tolerance (Section 12), is that the toric code is a quantum low-density parity-check code.

Definition 9.7 (Quantum LDPC code)**:** A family of CSS codes is *quantum LDPC* if both H_X and H_Z are LDPC matrices. Equivalently, there are constants Δ_{C} and Δ_{B}, independent of n, such that every stabilizer check acts on at most Δ_{C} qubits and every qubit participates in at most Δ_{B} checks.

For the square toric-code family, $\Delta_{\mathrm{B}} = \Delta_{\mathrm{C}} = 4$ because every vertex or face check has weight four, while every edge belongs to two vertex checks and two face checks.

It is often useful to represent the code by a Tanner graph, as depicted in Figure 9.8. A combined CSS Tanner graph has one class of nodes for qubits, one class for X-checks, and one class for Z-checks; an edge records that a check acts on a qubit. The orthogonality constraint has the graph-theoretic form

$$(H_X H_Z^{\mathsf{T}})_{ab} = \big|N(a) \cap N(b)\big| \pmod 2 = 0, \tag{9.31}$$

where a is an X-check node, b is a Z-check node, and $N(\cdot)$ denotes the neighboring qubits. Thus every X-check/Z-check pair must share an even number of qubits.

Independently chosen sparse parity-check matrices generally fail this orthogonality condition. The difficult problem is to enforce it together with the LDPC condition and useful code parameters n, k, and d. The topological construction is important because the chain-complex identity $\partial_1 \partial_2 = 0$ enforces commutation automatically, while the topology controls the logical operators and distance.

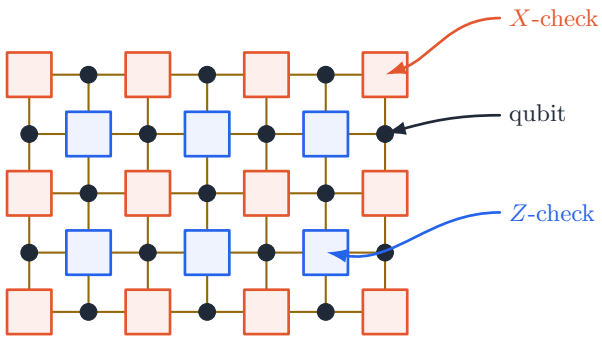

Figure 9.8: Tanner graph for a patch of the toric code.

We will see some more sophisticated strategies for how to build chain complexes not based on finite-dimensional manifolds in Section **??**, which can lead to better values of k in particular.

10 Topologically ordered phases of matter

In Section 9 we introduced the toric code on an $L \times L$ square lattice with periodic boundary conditions. There is one qubit on each edge, an X-check on each vertex, and a Z-check on each face. Recall the notation in (9.1), and consider the Hamiltonian defined by the sum of all of the commuting stabilizers of the toric code:

$$H_{\mathrm{TC}} = -\sum_{v \in V} X^{\mathrm{d}v} - \sum_{f \in F} Z^{\partial f}. \tag{10.1}$$

Today we will explore this code from the condensed-matter perspective: the toric code realizes a topologically ordered phase of matter.

10.1 The toric code as a $\mathbb{Z}_2$ lattice gauge theory

The Hamiltonian in Eq. (10.1) can be interpreted as a $\mathbb{Z}_2$ lattice gauge theory. To motivate this interpretation, first recall the more familiar example of electromagnetism, which is a U(1) gauge theory.

10.1.1 Putting electromagnetism on a lattice

In the absence of matter, Maxwell's equations are

$$\nabla \cdot \mathbf{E} = 0, \tag{10.2a}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{10.2b}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \tag{10.2c}$$

$$\nabla \times \mathbf{B} = \frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t}. \tag{10.2d}$$

The homogeneous equations $\nabla \cdot \mathbf{B} = 0$ and $\nabla \times \mathbf{E} = -\partial_t \mathbf{B}$ can be solved locally by introducing a scalar potential ϕ and a vector potential $\mathbf{A}$:

$$\mathbf{B} = \nabla \times \mathbf{A}, \tag{10.3a}$$

$$\mathbf{E} = -\nabla\phi - \frac{\partial \mathbf{A}}{\partial t}. \tag{10.3b}$$

Choose the gauge $\phi = 0$ and set $c = 1$. The electromagnetic Lagrangian then becomes

$$\mathcal{L} = \frac{1}{2}\mathbf{E}^2 - \frac{1}{2}\mathbf{B}^2 = \frac{1}{2}\left(\frac{\partial \mathbf{A}}{\partial t}\right)^2 - \frac{1}{2}(\nabla \times \mathbf{A})^2. \tag{10.4}$$

Thus $\mathbf{E}$ and $\mathbf{A}$ are canonically conjugate, with the sign convention

$$[E_i(\mathbf{x}), A_j(\mathbf{y})] = \mathrm{i}\,\delta_{ij}\delta(\mathbf{x} - \mathbf{y}), \tag{10.5}$$

and the Hamiltonian is

$$H_{\mathrm{EM}} = \int \mathrm{d}^3x \left[\frac{1}{2}\mathbf{E}^2 + \frac{1}{2}(\nabla \times \mathbf{A})^2\right]. \tag{10.6}$$

We have written the familiar $3+1$-dimensional vector notation in Eqs. (10.2)–(10.6). Only the canonical gauge-field structure will be used below. In $2+1$ dimensions, $\mathbf{B}$ is replaced by a pseudoscalar magnetic field and the spatial integral is $\int \mathrm{d}^2x$; the lattice construction proceeds in the same way.

To obtain a lattice gauge theory, place A_e and E_e on each oriented edge e. Using the discrete curl d_1 introduced in Section 9.2, the lattice Hamiltonian is

$$H_{\mathrm{lat}} = \frac{1}{2}\sum_{e\in E} E_e^2 + \frac{1}{2}\sum_{f\in F}\left(\sum_{e\in E}\langle f|\mathrm{d}_1|e\rangle A_e\right)^2. \tag{10.7}$$

The remaining Maxwell equation $\nabla\cdot\mathbf{E}=0$ becomes the Gauss-law constraint

$$\sum_{e\in E}\langle e|\mathrm{d}_0|v\rangle E_e = 0 \qquad \text{for every } v\in V. \tag{10.8}$$

One may enforce this constraint softly by adding a penalty:

$$H'_{\mathrm{lat}} = \frac{1}{2}\sum_{e\in E} E_e^2 + \frac{1}{2}\sum_{f\in F}\left(\sum_{e\in E}\langle f|\mathrm{d}_1|e\rangle A_e\right)^2 + g\sum_{v\in V}\left(\sum_{e\in E}\langle e|\mathrm{d}_0|v\rangle E_e\right)^2. \tag{10.9}$$

10.1.2 Making a discrete lattice gauge theory

The first modification is to make the gauge field compact:

$$A_e \sim A_e + 2\pi. \tag{10.10}$$

Just as angular momentum is quantized when its conjugate angle is periodic, Eq. (10.10) makes E_e integer-valued. It is then cleaner to replace the formal commutator $[E_e, A_e] = \mathrm{i}$ by the exponentiated Weyl relation

$$\mathrm{e}^{\mathrm{i}\phi E_e}\mathrm{e}^{\mathrm{i}\ell A_e} = \mathrm{e}^{\mathrm{i}\ell A_e}\mathrm{e}^{\mathrm{i}\phi E_e}\mathrm{e}^{-\mathrm{i}\ell\phi}, \qquad \ell\in\mathbb{Z}. \tag{10.11}$$

For a genuinely compact U(1) lattice gauge theory, the magnetic energy should also be periodic in the lattice flux, for example

$$-K\sum_{f\in F}\cos\left(\sum_{e\in E}\langle f|\mathrm{d}_1|e\rangle A_e\right). \tag{10.12}$$

The quadratic magnetic term in Eq. (10.7) is its small-flux approximation. This distinction will not affect the algebraic reduction to $\mathbb{Z}_2$.

The second modification is to restrict the edge Hilbert space to a binary $\mathbb{Z}_2$ degree of freedom. Heuristically, take

$$A_e \in \{0, \pi\}, \qquad E_e \in \{0, 1\}, \tag{10.13}$$

and define

$$Z_e := \mathrm{e}^{\mathrm{i}A_e}, \tag{10.14a}$$

$$X_e := \mathrm{e}^{\mathrm{i}\pi E_e}. \tag{10.14b}$$

Taking $\phi = \pi$ and $\ell = 1$ in Eq. (10.11) gives

$$X_e Z_e = Z_e X_e \mathrm{e}^{-\mathrm{i}\pi} = -Z_e X_e, \tag{10.15}$$

which is precisely the Pauli algebra.

The finite-dimensional theory should be understood directly through the exponentiated algebra in Eqs. (10.14) and (10.15). Two finite-dimensional Hermitian operators cannot obey an exact canonical commutator, so the notation in Eq. (10.13) is a useful mnemonic rather than a simultaneous diagonalization of A_e and E_e.

The resulting $\mathbb{Z}_2$ lattice gauge Hamiltonian takes the form

$$H_{\mathbb{Z}_2} = -h \sum_{e \in E} X_e - \sum_{f \in F} \prod_{e \in \partial f} Z_e - g \sum_{v \in V} \prod_{e \ni v} X_e. \tag{10.16}$$

The three terms are the $\mathbb{Z}_2$ analogues of the electric-field energy, the magnetic-field energy, and the Gauss-law penalty, respectively. At $h = 0$ and $g = 1$, Eq. (10.16) is the toric code Hamiltonian (10.1). For nonzero h, it is the toric code with an extra few-body perturbation.

10.1.3 Topological order and perturbative stability

The additional electric-field term in Eq. (10.16) does not immediately destroy the toric code. Instead, sufficiently weak few-body perturbations remain in the same phase of matter.

Theorem 10.1 (Stability of the toric-code phase)**:** Let

$$H_0' = H_{\mathrm{TC}} + hV, \tag{10.17}$$

where V is an extensive few-body, or q-local, perturbation, with q independent of L. For sufficiently small $|h|$, independent of L:

1. there exists a quasilocal unitary U that maps the ground-state space of H_{TC} to the four-dimensional low-energy space of H_0';

2. H_{TC} and H_0' are both gapped above their respective low-energy spaces;

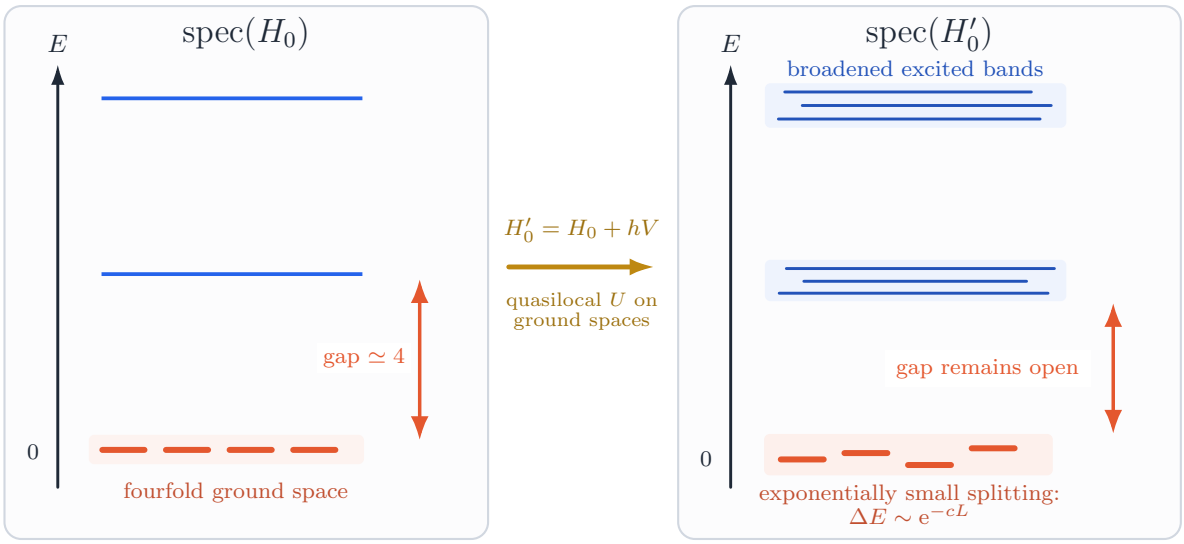

Figure 10.1: The toric code is a robust gapped phase of matter to any perturbation.

3. the splitting of the four low-energy levels obeys

$$\Delta E \lesssim \mathrm{e}^{-fL} \tag{10.18}$$

for some constant $f > 0$.

Here q-local means that V is a sum of terms acting on at most q qubits, with no requirement that those qubits be geometrically nearby. The smallness condition bounds the total interaction strength felt by each qubit; it does not bound the global operator norm $\|V\|$, which is extensive. The word "gap" means the gap above the entire four-dimensional low-energy band. At finite L, a generic perturbation may split the states inside that band by the exponentially small amount in Eq. (10.18). The consequences of Theorem 10.1 are illustrated in Figure 10.1.

The basic idea behind the stability theorem is a local form of the Knill–Laflamme conditions. Let $|\psi_\alpha\rangle$ denote the four toric-code ground states. Any operator O_R supported on a correctable region R obeys

$$\langle\psi_\alpha|O_R|\psi_\beta\rangle = c_{O_R}\delta_{\alpha\beta}. \tag{10.19}$$

In particular, an operator of weight less than the code distance cannot tell the ground states apart: recall Figure 9.3 and Figure 9.5 for an illustration of the operators O_R (correctable vs. uncorrectable errors) which can or cannot tell apart the codewords. Since the toric-code distance is $d = L$, perturbation theory must reach order proportional to L before it can generate a logical operator and lift the degeneracy. For a q-local perturbation, a product of r perturbing terms has weight at most qr. Thus the elementary distance-counting intuition is that one needs $r \geq d/q = \Theta(L)$ before a contribution can act logically. The full stability theorem controls the many perturbative processes

and the gap, using the textbook degenerate perturbation theory of quantum mechanics applied to operators (called Schrieffer-Wolff transformations).

The other required property, satisfied by the toric code, is check soundness.

Definition 10.2 (Check soundness)**:** A stabilizer Hamiltonian has (d_c, F)-check soundness if every stabilizer acting nontrivially on $M < d_c$ qubits can be written as a product of at most $F(M)$ Hamiltonian checks. For the toric code one may take

$$d_c = \Theta(L), \qquad F(M) \lesssim M^2. \tag{10.20}$$

Geometrically, a contractible loop of length M can be filled by a region containing at most order M^2 plaquettes or vertices (recall Figure 9.3 and Figure 9.5). Multiplying the checks in that filling produces the loop operator, which gives the scaling in Eq. (10.20).

For the reasons above, we say that the toric code is therefore a **topologically ordered phase of matter**. Two characteristic properties are:

1. the low-energy degeneracy depends on the topology of the spatial surface;

2. no local order parameter distinguishes the different ground states.

Indeed, products of local stabilizers produce contractible closed loops, whereas an operator that distinguishes or maps between ground states must wind nontrivially around the torus.

10.2 Comparison with a symmetry-breaking phase

It is useful to contrast the toric code with an ordinary phase of matter described by a classical repetition code. Consider

$$H_{\mathrm{rep}} = -\sum_{i\sim j} Z_i Z_j. \tag{10.21}$$

Theorem 10.3 (Stability of the ferromagnetic phase)**:** Let

$$H = H_{\mathrm{rep}} + hV. \tag{10.22}$$

For sufficiently small $|h|$, the system remains in a ferromagnetic phase if the perturbation is locally symmetric: writing $V = \sum_R V_R$, every local term obeys

$$[V_R, X_1 X_2 \cdots X_n] = 0. \tag{10.23}$$

A simple example is the one-dimensional quantum Ising model

$$H_{\mathrm{Ising}} = -\sum_{i=1}^{n-1} Z_i Z_{i+1} - \sum_{i=1}^{n} (hX_i + \epsilon Z_i). \tag{10.24}$$

When $\epsilon = 0$ and h is small, the system is ferromagnetic. The $\epsilon = 0$ transverse-field Ising chain is exactly solvable. By contrast, when $\epsilon \neq 0$, the longitudinal field explicitly breaks the global $\mathbb{Z}_2$ symmetry and selects a unique ground state as $n \to \infty$. The reason is visible directly from the Knill–Laflamme condition:

$$\langle 0 \cdots 0 | \epsilon \sum_{i=1}^{n} Z_i | 0 \cdots 0 \rangle = \epsilon n, \tag{10.25a}$$

$$\langle 1 \cdots 1 | \epsilon \sum_{i=1}^{n} Z_i | 1 \cdots 1 \rangle = -\epsilon n. \tag{10.25b}$$

Thus an arbitrarily weak symmetry-breaking field distinguishes the two codewords at first order, with an extensive energy difference.

As usual for spontaneous symmetry breaking, the degenerate ferromagnetic ground states are most sharply defined in the thermodynamic limit. At finite n, a symmetry-preserving transverse field can produce an exponentially small tunneling splitting between the even and odd combinations of the two classical codewords. This is similar to the exponentially small splitting allowed by Theorem 10.1.

The contrast can be summarized by comparing the phase diagram of (10.24) to a generic perturbed toric code:

$$H'_{\mathrm{TC}} = H_{\mathrm{TC}} - h \sum_{e \in E} X_e - \epsilon \sum_{e \in E} Z_e. \tag{10.26}$$

As depicted in Figure 10.2, for the Ising model, the ferromagnetic phase is protected only along the symmetry-preserving line $\epsilon = 0$; a generic nonzero longitudinal field selects a trivial phase. For the toric code, an open two-dimensional region around $(h, \epsilon) = (0, 0)$ remains topologically ordered. At sufficiently large ϵ, the e particles condense, while at sufficiently large h, the m particles condense. The labels follow from the error strings discussed below: a Z_e term creates or moves e excitations, while an X_e term creates or moves m excitations. The key point is that the stability theorem permits finite generic perturbation strength without imposing a microscopic symmetry. This is the robustness of the $\mathbb{Z}_2$ gauge theory as a phase of matter.

10.3 Excitations in the toric code

Unfortunately, this zero-temperature robustness does not extend to finite temperature in two spatial dimensions.

10.3.1 e and m particles as Z and X syndromes

To understand why, reinterpret an error syndrome as a collection of excitations, as depicted in Figure 10.3. An open Z-string anticommutes with the two vertex checks at its endpoints and therefore creates two e excitations. An open dual-lattice X-string anticommutes with the two face checks at its dual endpoints and creates two m excitations. Thus e and m excitations are created

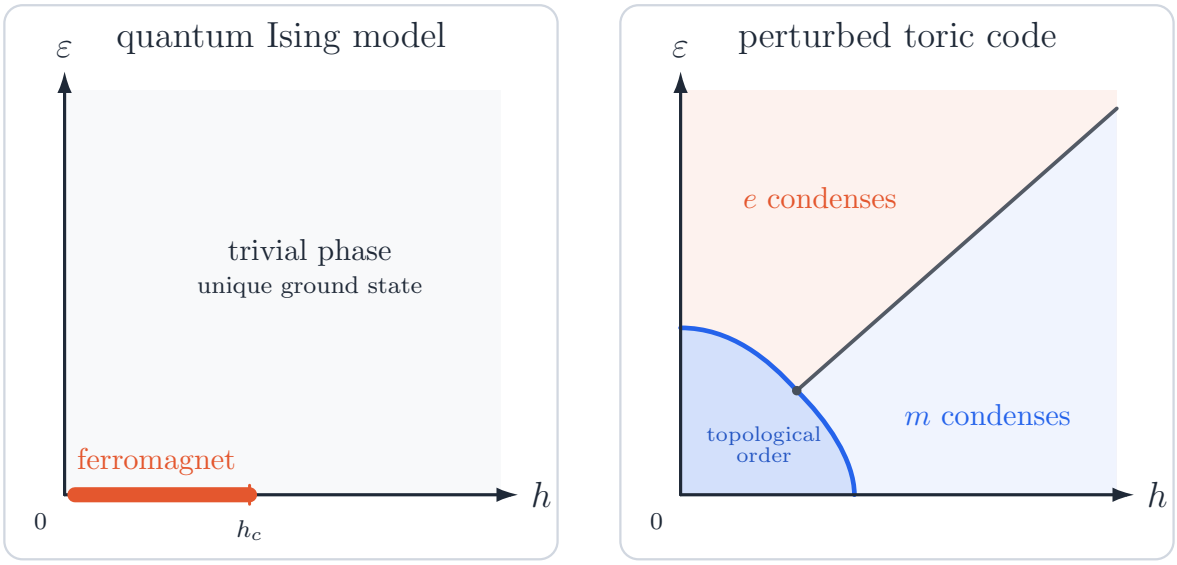

Figure 10.2: Comparing the phase diagrams of the quantum Ising model (10.24) vs. the perturbed toric code (10.26). Symmetry-breaking perturbations destroy ferromagnetism immediately, but no such analogue exists for the toric code.

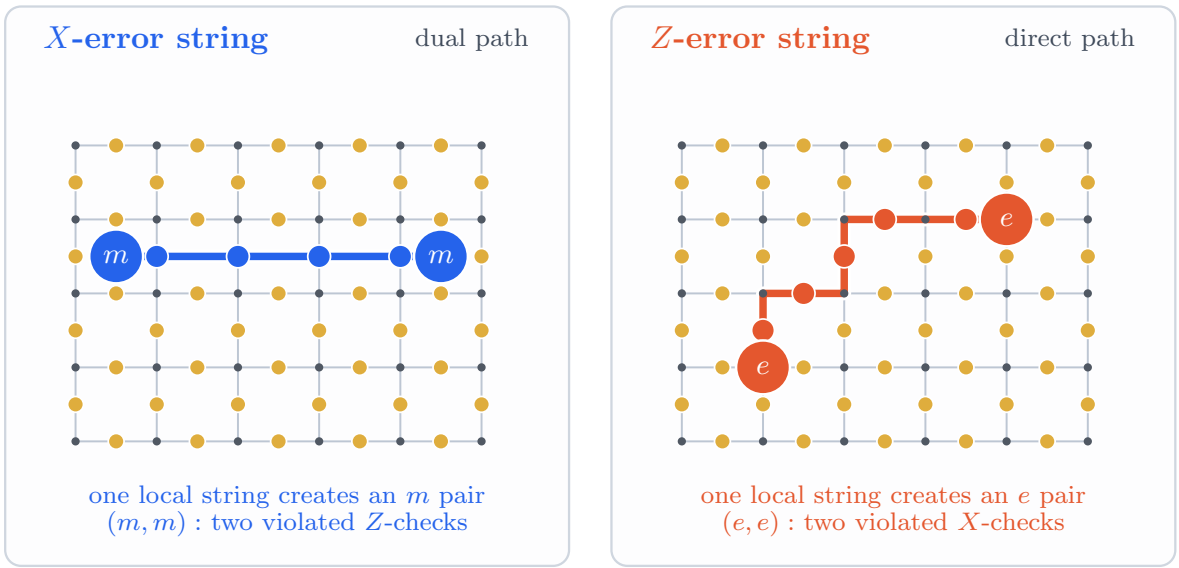

Figure 10.3: e vs. m excitations are created by Z vs. X type errors in the toric code.

in pairs on the torus, much as domain walls occur in pairs in one-dimensional Ising model (except at the boundary, with open boundary conditions).

Definition 10.4 (Fractionalization): The e and m excitations are fractionalized: an isolated e or m cannot be created by a finite-support operator, but after a pair has been created, the two

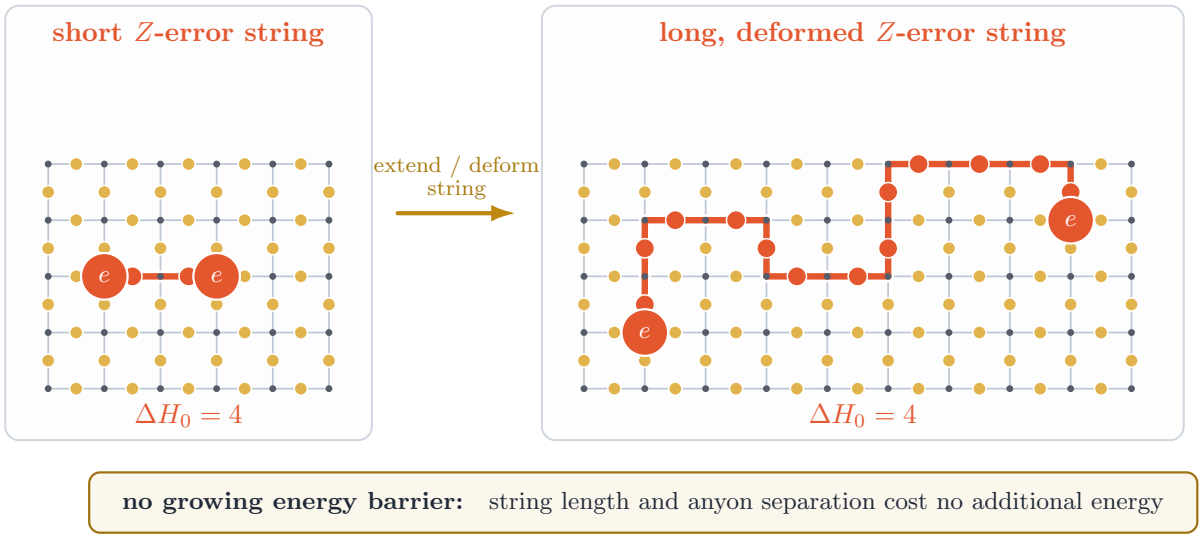

Figure 10.4: There is no energy barrier to causing e or m pairs to widely separate.

excitations can be moved independently.

10.3.2 No energy barrier in the toric code

Once a pair has been created, there is no additional energy penalty for moving its members farther apart: see Figure 10.4. Each violated toric-code check costs energy 2, so a pair has energy

$$\Delta H_0 = 4 \tag{10.27}$$

independently of its separation. The entropy associated with placing the two endpoints on an $L \times L$ torus scales as

$$\Delta S_{e\text{-pair}} \sim \log\left[(L^2)^2\right] = 4 \log L. \tag{10.28}$$

Setting Boltzmann's constant to one, the corresponding free-energy cost is

$$\Delta F_{e\text{-pair}} \sim \Delta H_0 - T\Delta S_{e\text{-pair}} = 4 - T \log\left[(L^2)^2\right] = 4(1 - T \log L). \tag{10.29}$$

For every fixed $T > 0$, this becomes negative at sufficiently large L. Consequently, both e and m excitations proliferate in the thermodynamic limit because their energy barriers are finite. Thus, the two-dimensional toric code has an $O(1)$ energy barrier: a noncontractible logical string can be grown while maintaining only its two endpoint excitations, at energy cost 4. We will see that it is not a self-correcting quantum memory at any fixed $T > 0$ in Section **??**; we require an active decoder to correct errors in the toric code.

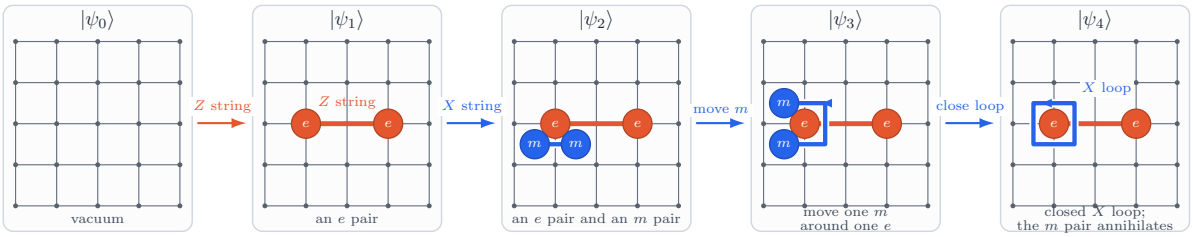

Figure 10.5: Braiding an m around an e leads to an overall phase in the wave function.

10.3.3 Braiding statistics and Abelian anyons

Before discussing the statistics of these excitations, recall that ordinary bosonic and fermionic wavefunctions obey

$$\psi_{\text{boson}}(x_1, x_2) = \psi_{\text{boson}}(x_2, x_1), \tag{10.30a}$$

$$\psi_{\text{fermion}}(x_1, x_2) = -\psi_{\text{fermion}}(x_2, x_1). \tag{10.30b}$$

The e and m excitations have nontrivial braiding statistics and are Abelian anyons. Consider the thought experiment of Figure 10.5. Begin in a toric-code ground state $|\psi_0\rangle$ and create a pair of e excitations with an open Z-string along γ:

$$|\psi_1\rangle = (Z\text{-string})|\psi_0\rangle. \tag{10.31}$$

Next create a pair of m excitations, move one of them around one endpoint of γ, and subsequently annihilate the m pair. The net operation is a contractible X-loop $W_X(\ell^*)$ on the dual lattice which crosses γ once. The final state is therefore

$$|\psi_4\rangle = (X\text{-loop})(Z\text{-string})|\psi_0\rangle = -(Z\text{-string})(X\text{-loop})|\psi_0\rangle = -|\psi_1\rangle \tag{10.32}$$

The minus sign in the second line arises because the two string operators cross once. Recall that the X-loop is a stabilizer of the toric-code ground state. We conclude that braiding an m excitation around an e excitation produces the phase -1.

By contrast, all Z Pauli operators commute with one another, as do all X Pauli operators. Hence braiding two excitations of the same species gives phase $+1$. For these reasons, one says that e and m each have bosonic self-statistics, but they are **mutual semions**: winding one species around the other produces the phase -1. Their bound state $\varepsilon = e \times m$ is a fermion.

11 Surface code

The logical operators of the toric code rely on nontrivial topology. Although it is possible to realize a torus in the laboratory with movable neutral atoms, or cleverly wired superconducting qubits, it is much more natural to ask whether the same basic construction can be implemented on a two-dimensional planar grid. The essential step is to choose suitable boundary conditions. The result is the **surface code**.

11.1 Planar code on an annulus

We first place the toric code checks on a cellulation of an annulus. There is a qubit on every edge. For every face f and vertex v, define

$$Z^{\partial f} := \prod_{e \in \partial f} Z_e, \tag{11.1a}$$

$$X^{\mathrm{d}v} := \prod_{e \ni v} X_e. \tag{11.1b}$$

There is no face, and hence no Z-check, filling the hole in the center of the annulus: see Figure 11.1. Exactly as for the toric code, a vertex and a face share either zero or two edges, and therefore

$$\left[X^{\mathrm{d}v}, Z^{\partial f}\right] = 0. \tag{11.2}$$

Proposition 11.1: The annular code encodes one logical qubit.

Proof. See Figure 11.2. The annulus has one nontrivial first cohomology class,

$$\mathrm{H}^1(\text{annulus}; \mathbb{F}_2) \cong \mathbb{F}_2. \tag{11.3}$$

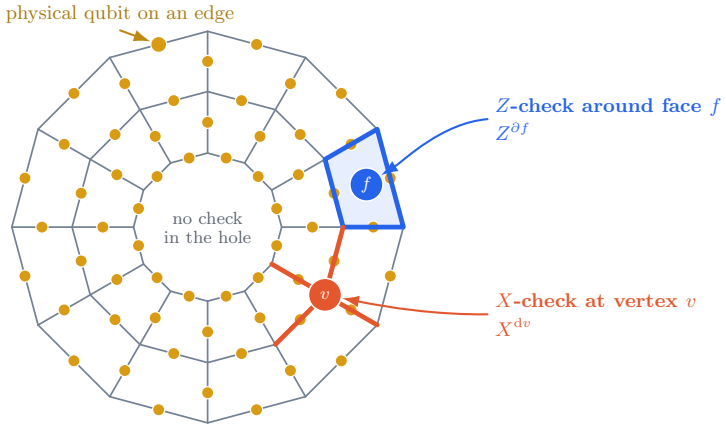

Figure 11.1: Putting toric code checks on the cellulation of an annulus.

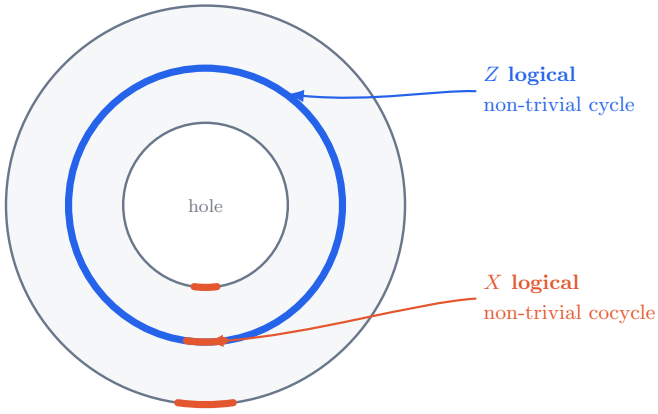

Figure 11.2: Sketch of logical operators (cycles and cocycles) on the annulus code.

A noncontractible primal cycle around the hole supports a Z-logical operator: it commutes with every X-check but cannot be shrunk to a product of face checks. A dual path from the inner boundary to the outer boundary supports the corresponding X-logical operator. The two representatives can be chosen to cross once and therefore anticommute. Hence the code carries one logical qubit. □

Equivalently, the Z-logical is the nonzero class in $\mathrm{H}_1(\text{annulus}; \mathbb{F}_2)$, while the X-logical is its Poincaré-dual cohomology class. This is the same cycle–cocycle pairing that appeared for the toric code.

The annulus is already a substantial improvement over a torus, since it can be embedded in the plane. A still cleaner and more compact construction replaces the hole by mixed boundary conditions.

11.2 A planar surface code from boundary conditions

Begin with a rectangular cellulation of a disk. It is useful to represent two opposite boundaries by two *ghost vertices*. We retain all face Z-checks adjacent to the ghost vertices, but remove the two corresponding vertex X-checks from the stabilizer group, as shown in Figure 11.3. Thus, if the augmented cellulation has V vertices, E edges, and F faces, then the physical qubits live on the E edges, while there are $V - 2$ vertex checks and F face checks.

The following elementary Euler-characteristic identity will be useful.

Lemma 11.2: For a cellulation of a disk,

$$V - E + F = 1, \tag{11.4}$$

where F counts the faces inside the disk and does not include the exterior.

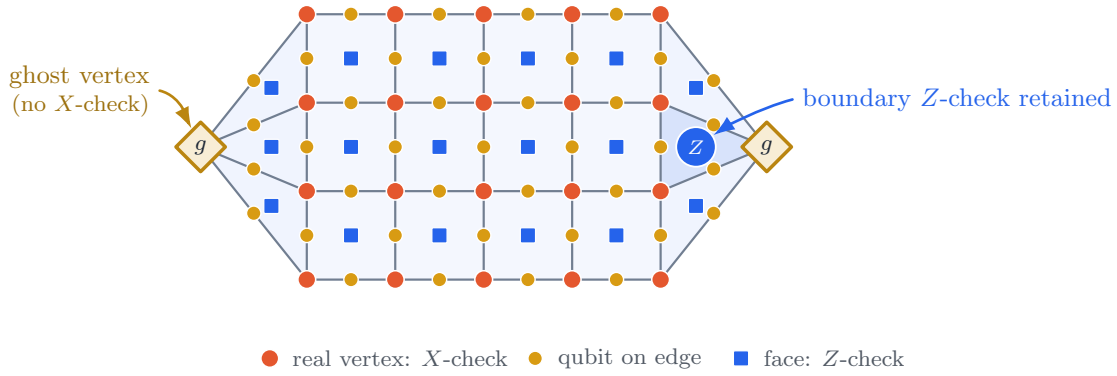

Figure 11.3: A special cellulation of a planar grid with two single ghost vertices at the left and right ends, which will lead us to our first surface code.

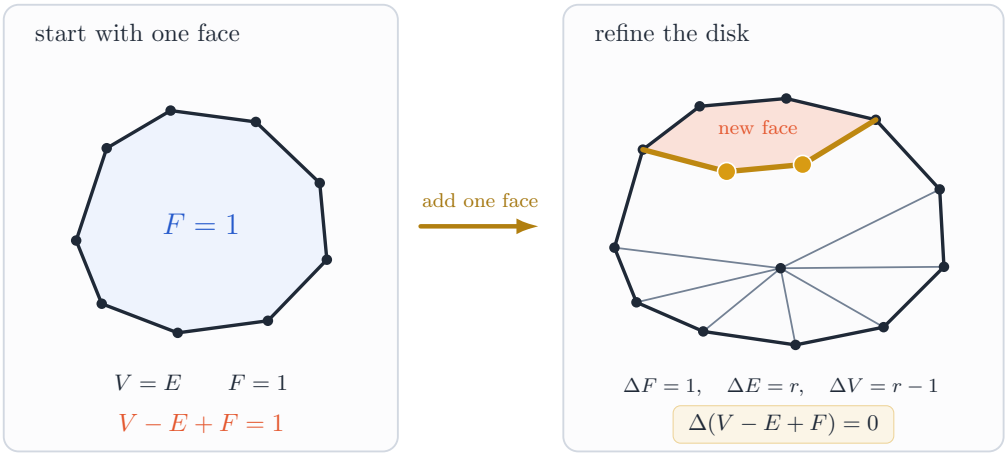

Figure 11.4: Illustrating the proof of Lemma 11.2.

Proof. For a polygon with no internal edges, $V = E$ and $F = 1$, so Eq. (11.4) holds. Add the internal faces one at a time. If a newly added face introduces ΔE edges, then it introduces one face and $\Delta E - 1$ new vertices. Therefore

$$F \longmapsto F + 1, \tag{11.5a}$$

$$E \longmapsto E + \Delta E, \tag{11.5b}$$

$$V \longmapsto V + (\Delta E - 1), \tag{11.5c}$$

which leaves $V - E + F$ unchanged: see Figure 11.4. □

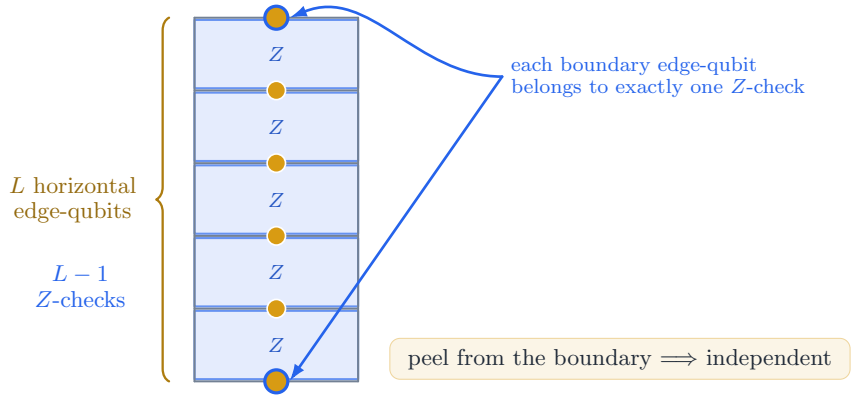

Figure 11.5: The Z-checks in each strip are all linearly independent of each other, and linearly independent from any Z-check in any other strip.

Lemma 11.3: All checks in this planar construction are independent. In particular,

$$\#(X\text{-checks}) = V - 2 = \operatorname{rank}(H_X), \tag{11.6a}$$

$$\#(Z\text{-checks}) = F = \operatorname{rank}(H_Z). \tag{11.6b}$$

Proof. The first equalities follow directly from the construction, so it remains to show independence. Consider the Z-checks in one boundary strip, illustrated in Figure 11.5. The L horizontal qubits in that strip are contained only in the $L-1$ adjacent Z-checks, and the corresponding rows of H_Z are full rank, since:

$$H_Z = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & \cdots \\ 0 & 1 & 1 & \ddots & \vdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \cdots \\ 0 & \cdots & 0 & 1 & 1 & \cdots \end{pmatrix}, \tag{11.7}$$

The same argument is then applied to the next strip. Iterating proves that all Z-checks are independent. The same peeling argument, applied in the dual direction, proves independence of the X-checks. □

Proposition 11.4: The planar surface code encodes one logical qubit.

Proof. Using Lemmas 11.2 and 11.3,

$$k = n - \operatorname{rank}(H_X) - \operatorname{rank}(H_Z) = E - (V-2) - F = 2 - (V - E + F) = 1, \tag{11.8}$$

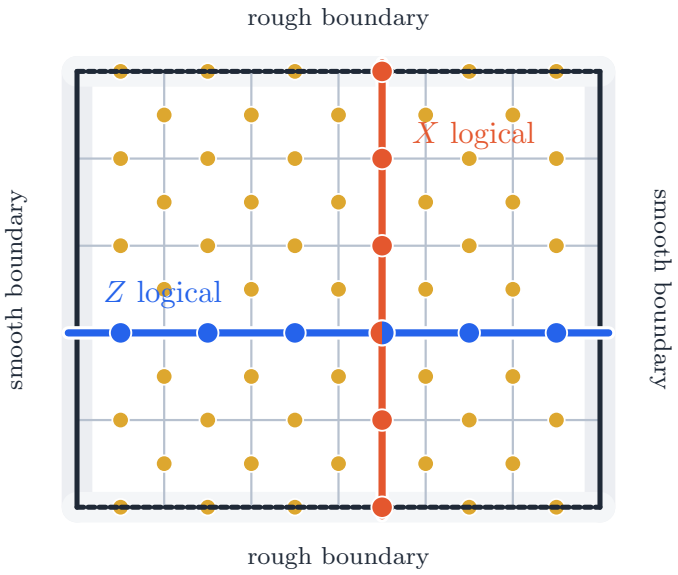

Figure 11.6: Logical operators in the surface code.

which completes the proof. □

A primal Z-string may terminate on the two boundaries where the relevant X-checks have been removed; these are the *rough* boundaries. A dual X-string may terminate on the complementary pair of *smooth* boundaries. Thus a representative Z-logical joins the rough boundaries, while a representative X-logical joins the smooth boundaries: see Figure 11.6.

For the rectangular family drawn in the lecture script, the numbers of qubits, ordinary vertices, and faces are

$$n = L(L-1) + (L-1)(L-2) = 2(L-1)^2, \tag{11.9a}$$

$$\#(X\text{-checks}) = L(L-2), \tag{11.9b}$$

$$\#(Z\text{-checks}) = (L-1)^2. \tag{11.9c}$$

The shortest Z-logical has weight $L-1$, whereas the shortest X-logical has weight L. Hence the parameters are $[\![2(L-1)^2, 1, L-1]\!]$; more precisely,

$$d_Z = L-1, \qquad d_X = L. \tag{11.10}$$

11.3 The rotated surface code

The preceding construction can be made more efficient. Put qubits on the vertices of an $L \times L$ square lattice. As shown in Figure 11.7, color the plaquettes in a checkerboard pattern and place X-checks on one color and Z-checks on the other. Along the boundary, add alternating weight-2

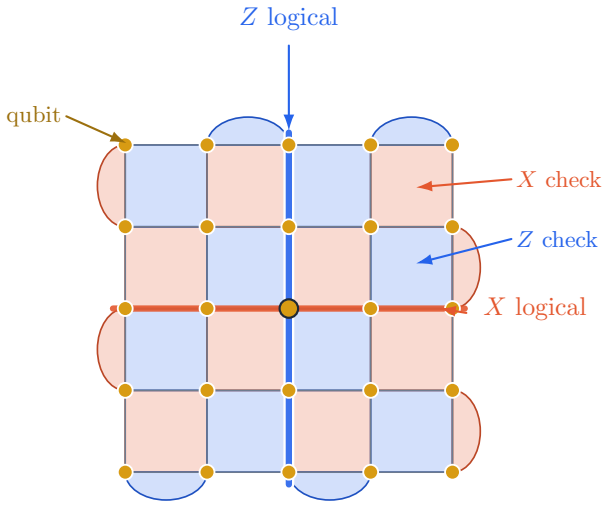

Figure 11.7: The rotated surface code.

checks, so that the boundary "bubbles" skip every other plaquette. The weight-2 Z-checks lie on the rough boundaries, while the weight-2 X-checks lie on the smooth boundaries.

Proposition 11.5: The rotated checks obey

$$H_X H_Z^{\mathsf{T}} = 0. \tag{11.11}$$

Proof. If two plaquettes share an edge, they overlap on two qubits, so an X-check and a Z-check commute. If two plaquettes share only one vertex, then the bipartite coloring of the square lattice assigns them the same check type. The alternating boundary checks are arranged by the same rule. Thus every X-check and Z-check overlap on either zero or two qubits, which proves Eq. (11.11). □

Proposition 11.6: The rotated surface code has parameters $[\![L^2, 1, L]\!]$.

Proof. There are L^2 qubits. The number of bulk plaquette checks plus boundary weight-2 checks is

$$(L-1)^2 + 2(L-1) = L^2 - 1. \tag{11.12}$$

A boundary-peeling argument, analogous to the proof of Lemma 11.3, shows that these checks are independent. Therefore $k = L^2 - (L^2 - 1) = 1$. A Z-string joining the rough boundaries and an X-string joining the smooth boundaries each have weight L: see Figure 11.7. Conversely, a nontrivial logical string must span the patch between the corresponding pair of boundaries, and hence cannot have smaller weight. □

The rotated layout uses roughly half as many qubits as the unrotated layout at fixed distance. It is also the layout most commonly used in experimental work. Henceforth, when we say "surface

code," we will generally mean this rotated version.

11.4 The Bravyi–Poulin–Terhal bound

The surface code constructed above has only $k = 1$ logical qubit. One may ask whether a geometrically local code can have $k \gg 1$ at large n without sacrificing the scaling $d \sim L$.

Theorem 11.7 (Bravyi–Poulin–Terhal bound)**:** For a family of codes in $D \geq 2$ spatial dimensions with geometrically local checks,

$$k\, d^{\frac{2}{D-1}} = \mathrm{O}(n). \tag{11.13}$$

Here geometrically local means that the check diameter and the density of qubits per unit volume are bounded independently of n. The implicit constant in Eq. (11.13) may depend on these microscopic bounds.

We prove the result in two spatial dimensions. The key algebraic input is the cleaning lemma.

Lemma 11.8 (Cleaning lemma)**:** Let R be a set of qubits such that every Pauli error supported in R is correctable. For every logical Pauli class $[L] \in N(\mathcal{S})/\mathcal{S}$, there exists $S \in \mathcal{S}$ such that

$$\mathrm{supp}(LS) \cap R = \varnothing. \tag{11.14}$$

In other words, every logical operator has an equivalent representative with no support in R.

Proof. Work with Pauli operators modulo phases, represented as vectors in $\mathbb{F}_2^{2n}$ with the symplectic inner product. Let H be the Pauli parity-check matrix, so that the stabilizer subspace is

$$\mathcal{S} = \mathrm{im}(H^{\mathsf{T}}). \tag{11.15}$$

Let π_R restrict a Pauli vector to R, and define

$$W := \pi_R(\mathcal{S}). \tag{11.16}$$

For $q \in W^{\perp}$, extend q by zero outside R. The resulting Pauli commutes with every stabilizer. Since it is supported in the correctable region R, it cannot represent a nontrivial logical operator and must itself lie in $\mathcal{S}$.

Now let ℓ represent the logical operator L. For every $q \in W^{\perp}$, the preceding paragraph implies

$$q \cdot \pi_R(\ell) = 0. \tag{11.17}$$

Consequently,

$$\pi_R(\ell) \in (W^{\perp})^{\perp} = W. \tag{11.18}$$

There is therefore a stabilizer vector $s \in \mathcal{S}$ satisfying $\pi_R(s) = \pi_R(\ell)$. In additive Pauli notation,

$$\pi_R(\ell + s) = 0, \tag{11.19}$$

which is precisely Eq. (11.14) in multiplicative operator notation. □

The first geometric consequence is that well-separated correctable regions may be cleaned simultaneously.

Lemma 11.9 (Union lemma)**:** Let $R_1, \ldots, R_m$ be correctable regions. Suppose that no stabilizer generator intersects two distinct regions R_i and R_j. Then the union

$$R := \bigcup_{i=1}^{m} R_i \tag{11.20}$$

is correctable, and every logical operator can be cleaned out of all of R.

Proof. First consider a Pauli Q supported on R that commutes with every stabilizer. Decompose it as $Q = Q_1 \cdots Q_m$, with Q_i supported on R_i. Because no stabilizer generator meets two distinct regions, the commutation of Q with every generator implies that each Q_i separately commutes with every generator. Correctability of R_i then implies $Q_i \in \mathcal{S}$, and hence $Q \in \mathcal{S}$. Thus R supports no nontrivial logical operator and is correctable.

To clean a logical operator from the full union, apply the cleaning lemma successively to the regions R_i. When cleaning R_i, write the required stabilizer as a product of local generators and discard every generator disjoint from R_i; this leaves its restriction to R_i unchanged. Every remaining generator intersects R_i and, by the separation hypothesis, intersects no other R_j. Thus cleaning R_i cannot restore support on a region that has already been cleaned. After all regions have been treated, the logical representative has no support on their union. □

The second consequence is sometimes called a holographic principle for local codes.

Lemma 11.10: For a two-dimensional local stabilizer code of distance d, there is a constant $c > 0$ such that every square of linear size

$$\ell_0 = cd \tag{11.21}$$

is correctable.

Proof. Let $w = \mathrm{O}(1)$ be larger than the diameter of every stabilizer generator. A region containing fewer than d qubits is correctable by the definition of code distance. Begin with a square core of linear size $\mathrm{O}(\sqrt{d})$, chosen so that it contains fewer than d qubits. Surround it by concentric shells of width w, as shown in Figure 11.8. A shell at linear scale ℓ contains $\mathrm{O}(w\ell)$ qubits, and is therefore correctable as long as ℓ is at most a sufficiently small constant times d/w.

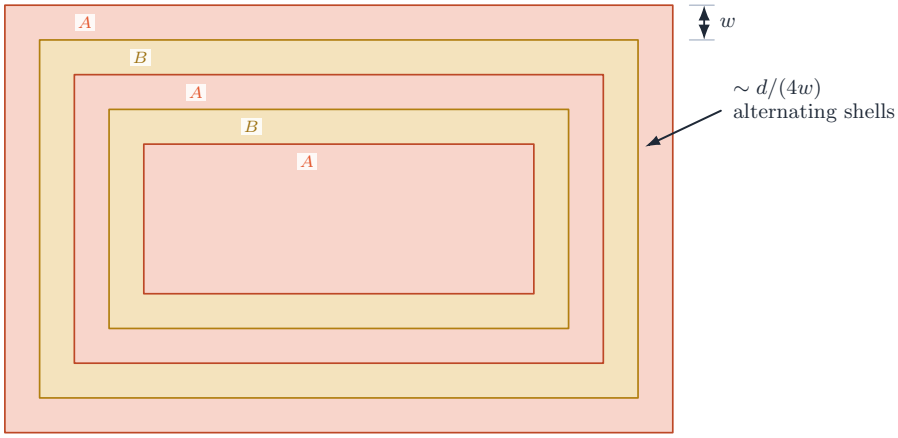

Figure 11.8: Applying the cleaning lemma to a series of concentric shells proves Lemma 11.10.

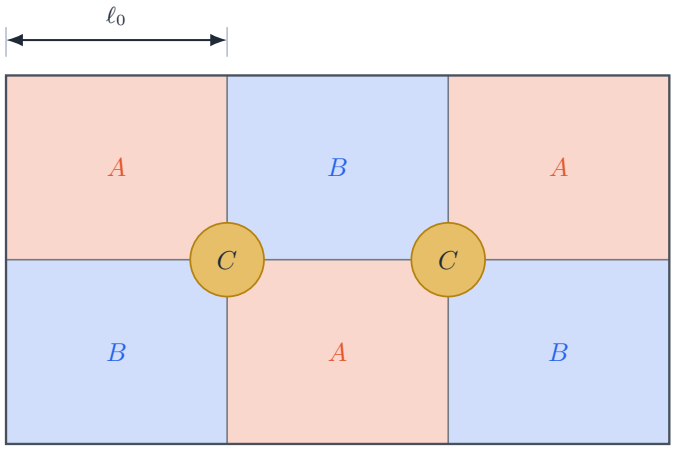

Figure 11.9: The regions A, B and C used for cleaning logicals.

Apply the union lemma separately to all red shells and to all yellow shells. Shells of the same color are separated by more than the check diameter, so each color class is correctable. Choose the outermost shell to be yellow. If a nontrivial logical operator were supported inside the entire square, clean it out of the red shells using only generators that meet those shells. The outermost yellow shell is a buffer of width at least w, so these generators remain inside the square. The resulting logical representative is supported entirely on the yellow region, contradicting its correctability. Thus the full square is correctable up to linear size $\ell_0 \sim d/w \sim d$. □

As shown in Figure 11.9, we now partition the two-dimensional system into square patches of linear size ℓ_0, colored alternately A and B. Around every point where patches of the same color would otherwise meet, remove an $\mathrm{O}(w)$-sized buffer region and place it in a third set C. The buffers are large enough that no stabilizer generator connects two distinct A-patches or two distinct

B-patches.

Each individual patch is correctable by Lemma 11.10, and the union lemma implies that the full regions A and B are correctable. Let $\{\overline{X}_a, \overline{Z}_a\}_{a=1}^{k}$ be canonical logical Pauli generators. Clean every $\overline{X}_a$ out of A and every $\overline{Z}_a$ out of B. The resulting representatives obey

$$\operatorname{supp}(\overline{X}_a) \subseteq B \cup C, \tag{11.22a}$$

$$\operatorname{supp}(\overline{Z}_a) \subseteq A \cup C. \tag{11.22b}$$

Since A and B are disjoint, the canonical anticommutation pairing is realized entirely inside C:

$$\pi_C(\overline{X}_a) \cdot \pi_C(\overline{Z}_b) = \delta_{ab}. \tag{11.23}$$

The $2k$ restricted logical vectors are therefore linearly independent in the $2|C|$-dimensional Pauli space on C, and hence

$$k \leq |C|. \tag{11.24}$$

There is one $\mathrm{O}(w)$-sized C-region per area ℓ_0^2, so

$$|C| = O\left(\frac{n}{\ell_0^2}\right) = O\left(\frac{n}{d^2}\right). \tag{11.25}$$

Combining Eqs. (11.24) and (11.25) proves (11.13) in two dimensions.

In D dimensions, a constant-width shell around a cube of side ℓ contains $\mathrm{O}(w\ell^{D-1})$ qubits. The same shell argument therefore gives $\ell_0 = \Theta(d^{1/(D-1)})$. The set C is an $\mathrm{O}(w)$-thickened codimension-two skeleton, so it occupies an $\mathrm{O}(\ell_0^{-2})$ fraction of the system. This produces the exponent in Eq. (11.13).

For the rotated surface code, $n = L^2$, $k = 1$, and $d = L$, so

$$kd^2 = n. \tag{11.26}$$

Thus the surface code is "best possible" if one demands both geometric locality and the largest possible distance at fixed n.

11.5 Thermodynamics of the surface code

We have constructed a good two-dimensional code, but this does not yet explain how quantum information should be protected in it. The surface code is not a passive quantum memory at finite temperature: it requires an active decoder, which will be the subject of the next lecture.

A closely related, although not equivalent, observation comes from equilibrium thermodynamics.

Definition 11.11 (Gibbs state)**:** Define the Gibbs state

$$\rho_\beta := \frac{\mathrm{e}^{-\beta H_0}}{\mathrm{tr}(\mathrm{e}^{-\beta H_0})}, \tag{11.27}$$

where the (here, surface code) Hamiltonian is

$$H_0 := -\sum_{a \in \text{checks}} S_a. \tag{11.28}$$

Proposition 11.12: For a surface code on n qubits with one logical qubit and independent checks, the partition function is

$$Z(\beta) := \mathrm{tr}(\mathrm{e}^{-\beta H_0}) = 2(2\cosh\beta)^{n-1}. \tag{11.29}$$

Proof. There are $n-1$ independent stabilizer checks. Consequently every assignment of check eigenvalues $s_a \in \{\pm 1\}$ occurs, and the simultaneous eigenspace for each syndrome has dimension two, corresponding to the one logical qubit. Choosing simultaneous eigenstates of all checks,

$$Z(\beta) = 2\sum_{s_1,\ldots,s_{n-1}=\pm 1} \exp\left(\beta \sum_{a=1}^{n-1} s_a\right) = 2\prod_{a=1}^{n-1}\left(\mathrm{e}^{\beta} + \mathrm{e}^{-\beta}\right) = 2(2\cosh\beta)^{n-1}, \tag{11.30}$$

which is (11.29). □

The free-energy density is therefore

$$\begin{aligned} f_n(\beta) &:= -\frac{1}{\beta n}\log Z(\beta) \\ &= -\frac{\log 2}{\beta n} - \frac{n-1}{\beta n}\log(2\cosh\beta) \xrightarrow[n\to\infty]{} -\frac{1}{\beta}\log(2\cosh\beta). \end{aligned} \tag{11.31}$$

This limiting function is analytic for every finite β, so there is no finite-temperature thermodynamic phase transition detectable in the free energy. This static statement is suggestive of, but is not logically equivalent to, a statement about memory lifetime.

The result is consistent with the anyon picture from the toric code. There is only an O(1) energy barrier to nucleate an e- or m-anyon pair, and the two anyons can then be pulled apart without any additional energy cost. At any finite temperature, thermally activated anyons can therefore diffuse over long distances under local dynamics, producing large error strings. An active decoding procedure is needed to infer and remove these strings before they implement a logical operator.

Problems

Problem 11.1 (Wilson loops)**:** In high energy physics, one often uses **Wilson loop** observables to analyze lattice gauge theories, like the toric and surface code. In this problem, consider the *unrotated* surface code (Figure 11.3) on an $L \times L$ square lattice. Take a square loop γ of side

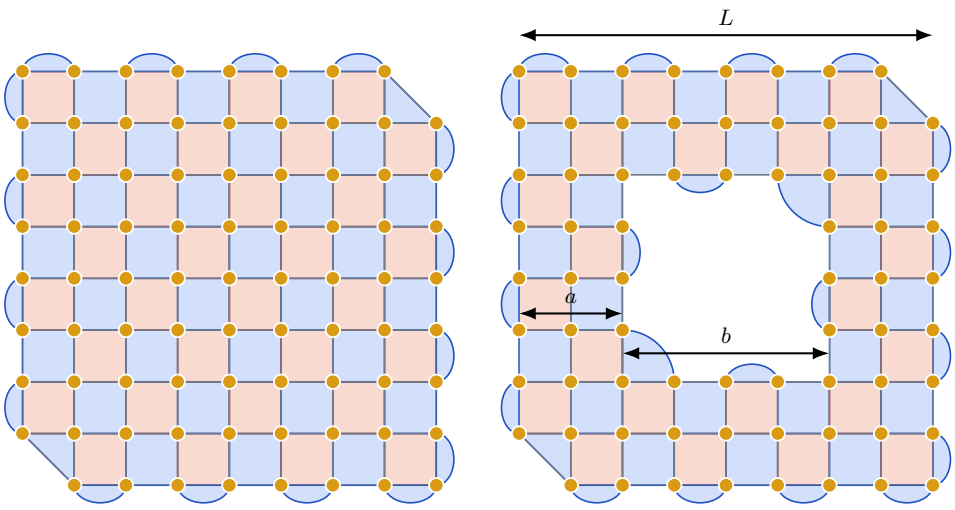

Figure 11.10: *Left:* A surface code with only rough boundaries. *Right:* A surface code with a hole in the middle. We use only rough boundaries for both the interior and exterior boundary.

length $b \ll L$ in the interior and consider the Wilson loop observable

$$Z^\gamma := \prod_{e \in \gamma} Z_e. \tag{11.32}$$

Let ρ_β denote the thermal density matrix for the surface code. Show that

$$\mathrm{tr}\left(\rho_\beta Z^\gamma\right) = \mathrm{e}^{-\alpha \cdot b^2} \tag{11.33}$$

and find the constant α as a function of β. It turns out that the exponential decay of the Wilson loop in terms of volume b^2, rather than perimeter $\sim 4b$, is a gauge-theoretic way of seeing that the topological phase in the 2d surface code is destroyed by thermal fluctuations.

Problem 11.2 (Holes in the surface code)**:** One mechanism for enhancing the surface code to store more logical qubits, while maintaining a simple planar architecture with only spatially local interactions, is to add holes to the surface code.

A: Consider the left surface code in Figure 11.10 which only has rough boundaries. What is n, the number of physical qubits? What are $\mathrm{rank}(H_X)$ and $\mathrm{rank}(H_Z)$, and therefore k? Contrast with the rough surface code (Figure 11.7) and explain the difference succinctly.

B: Now consider the right surface code in Figure 11.10, which has a hole in the middle. Again calculate n and k. What are the code distances d_X and d_Z, and where are the (smallest) logical representatives? Be sure to give a physical explanation for why the logicals are where they are, so that you could easily explain how your answer generalizes to a surface code (with hole) of any size.

C: Suppose that we changed the top and bottom boundaries to be smooth, instead of rough. Again, calculate n, k, d_X and d_Z. You should find that the answer differs from **B** – explain why by giving a clear physical picture for where all of the logical operators are.

D: Return to the scenario of **B**, where all interior and exterior boundaries are rough. Now, however, consider a surface code consisting of

$$L = (m+1)a + m(b-2) \tag{11.34}$$

Now place m^2 equally spaced holes in the surface code, with the distance between each hole equal to a, and the width of each hole equal to b. The case of $m = 1$ is given in the right half of Figure 11.10. Deduce the values of n, k, d_X and d_Z for this code. Given your earlier work, a clever argument will suffice – you do not need to justify it in full detail.

E: Argue using the BPT bound (Theorem 11.7) that there cannot be a qualitatively better way to encode extra logical qubits in a patch of surface code – namely, if you wish to maintain a given distance $d = \min(d_X, d_Z)$, you can't encode too many more logical qubits k than you did in the construction of **D**.

12 Fault-tolerant quantum memory

Section 11 constructed the (rotated) surface code (Figure 11.7). Our goal is now to use syndrome measurement and feedforward to protect its logical qubit for a long time. We begin with a single round of noisy syndrome extraction, and then explain why a genuine quantum memory must use the full history of many syndrome measurements. Where the ideas we describe here generalize directly to other quantum LDPC codes (Definition 9.7), we will emphasize this point explicitly.

12.1 One round of data and measurement errors

We use a simple phenomenological noise model. On each qubit, the X- and Z-components of the Pauli error occur independently with probability p_{e}. Thus the single-qubit Pauli probabilities are

$$\mathbb{P}(P_i = P) = \begin{cases} (1-p_{\mathrm{e}})^2, & P = \mathbb{I}, \\ p_{\mathrm{e}}(1-p_{\mathrm{e}}), & P = X, \\ p_{\mathrm{e}}(1-p_{\mathrm{e}}), & P = Z, \\ p_{\mathrm{e}}^2, & P = Y. \end{cases} \tag{12.1}$$

Equivalently, in the binary formalism the error is represented by $e_X, e_Z \in \mathbb{F}_2^n$, with

$$\mathbb{P}\big((e_X)_i = 1\big) = \mathbb{P}\big((e_Z)_i = 1\big) = p_{\mathrm{e}}. \tag{12.2}$$

For a CSS code, the ideal syndromes are

$$s_X = H_X e_Z, \tag{12.3a}$$
$$s_Z = H_Z e_X. \tag{12.3b}$$

The syndrome measurements are themselves imperfect, so the observed syndromes are

$$u_X = s_X + t_X, \tag{12.4a}$$
$$u_Z = s_Z + t_Z, \tag{12.4b}$$

where all additions are in $\mathbb{F}_2$, and each individual measurement is flipped independently with probability

$$\mathbb{P}\big((t_X)_\alpha = 1\big) = \mathbb{P}\big((t_Z)_\alpha = 1\big) = p_{\mathrm{m}}. \tag{12.5}$$

Even before addressing repeated measurements, it is useful to understand why a single round of weak data noise is unlikely to create a dangerous macroscopic error.

Proposition 12.1 (Single-round suppression of logical failure)**:** Consider a qLDPC code of distance d under the independent data-error model above, followed by minimum-weight decoding with an

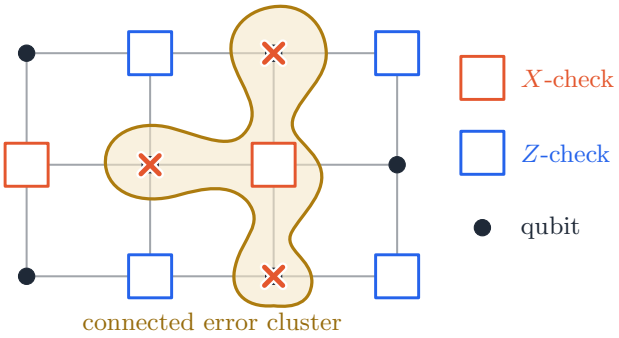

Figure 12.1: A connected cluster of errors in a surface code patch.

ideal syndrome. For sufficiently small fixed p_{e}, there is a constant $p_0 < 1$ such that

$$p_{\mathrm{fail}} \leq \mathrm{poly}(n)\, p_0^d. \tag{12.6}$$

Consequently, if $d \gg \log n$, then

$$p_{\mathrm{fail}} = \mathrm{e}^{-\Omega(d)}. \tag{12.7}$$

Given the surface code parameters $[[n = L^2, k = 1, d = L]]$, we see that p_{fail} is negligible as $L \to \infty$.

Proof. The proof is a connected-cluster counting argument, illustrated in Figure 12.1. For either the X- or Z-decoding problem, declare two qubits adjacent when they participate in a common check. Let Δ_{B} be the maximum number of relevant checks containing one qubit, and let Δ_{C} be the maximum number of qubits in one check. The resulting qubit-adjacency graph has maximum degree at most

$$D := \Delta_{\mathrm{B}}(\Delta_{\mathrm{C}} - 1). \tag{12.8}$$

Lemma 12.2 (Counting connected clusters)**:** Let $N_\ell(v)$ be the number of connected sets of ℓ qubits containing a fixed qubit v. Then

$$N_\ell(v) \leq (\mathrm{e}D)^{\ell-1} = \left[\mathrm{e}\Delta_{\mathrm{B}}(\Delta_{\mathrm{C}} - 1)\right]^{\ell-1}. \tag{12.9}$$

Proof sketch. At most D qubits are adjacent to any fixed qubit. Every connected set containing v contains a rooted spanning tree with root v. One can use generating functions to bound the number of such rooted trees of a given size to obtain (12.9). □

For a fully occupied connected error cluster of size ℓ containing a fixed qubit v, Lemma 12.2

gives

$$\mathbb{P}(v \text{ lies in an error cluster of size } \ell) \le p_{\mathrm{e}}^{\ell} N_{\ell}(v) \le \left(\mathrm{e} D p_{\mathrm{e}}\right)^{\ell}. \tag{12.10}$$

A decoder failure is slightly more general: the connected failure witness need only have at least 50% physical-error density. To see why, let e denote the physical error and r a minimum-weight recovery with the same syndrome. If decoding fails, then $e + r$ contains a connected component c that represents a nontrivial logical operator. Each such component has zero syndrome, because no check can meet two distinct components of the qubit-adjacency graph. Such a component must therefore have size at least the code distance:

$$|c| \ge d. \tag{12.11}$$

On c, the supports of e and r are complementary. Since replacing r by $r + c$ cannot reduce the weight of a minimum-weight recovery,

$$|e + c| \ge |r + c| \ge |e|. \tag{12.12}$$

Given that $|c| \ge d$ by the definition of code distance, we have

$$|e| \ge \frac{d}{2}. \tag{12.13}$$

The probability that at least half of the qubits in c had an error obeys

$$\mathbb{P}\left(|E \cap C| \ge \frac{\ell}{2}\right) \le \sum_{j=\lceil \ell/2 \rceil}^{\ell} \binom{\ell}{j} p_{\mathrm{e}}^{j} (1 - p_{\mathrm{e}})^{\ell - j} \le 2^{\ell} p_{\mathrm{e}}^{\ell/2}. \tag{12.14}$$

Taking a union bound over the n possible starting qubits, the two Pauli sectors, and all $\ell \ge d$ gives

$$p_{\mathrm{fail}} \le 2n \sum_{\ell \ge d} N_{\ell}(v)\, 2^{\ell} p_{\mathrm{e}}^{\ell/2} \le \frac{2n}{\mathrm{e} D} \sum_{\ell \ge d} \left(2\mathrm{e} D \sqrt{p_{\mathrm{e}}}\right)^{\ell}. \tag{12.15}$$

Absorbing the resulting combinatorial factor into a constant C_{LDPC} that depends only on the qLDPC degrees, one obtains the schematic bound

$$p_{\mathrm{fail}} \le 2n \left(C_{\mathrm{LDPC}} p_{\mathrm{e}}\right)^{d/2}. \tag{12.16}$$

For sufficiently small fixed p_{e}, this is exponentially small in d whenever $d \gg \log n$. □

This suggests the property we would like to preserve under repeated noisy correction: small error clusters should not accumulate into a logical error.

Definition 12.3 (Fault tolerance)**:** A code family and decoding procedure are **fault-tolerant** under a specified noise model if there is a nonzero region of fixed physical error rates for which the

probability of logical failure tends to zero as $n \to \infty$.

As we'll see later, a complete definition will also specify the decoder, the syndrome-extraction circuit, the desired storage time, and the metric used to quantify failure.

12.2 Quantum memory

Definition 12.4 (Quantum memory)**:** A quantum memory protects a logical state for a time $T(n)$ that diverges as the code size increases:

$$\lim_{n\to\infty} T(n) = \infty, \tag{12.17}$$

Let $\mathcal{E}$ denote one round of physical errors, $\widetilde{\mathcal{R}}$ a noisy recovery based on imperfect syndrome data, and $\mathcal{R}_f$ a final ideal recovery. Starting from a logical state ρ_{L}, we would like

$$\mathcal{R}_f \circ \left(\widetilde{\mathcal{R}} \circ \mathcal{E}\right)^{T(n)}(\rho_{\mathrm{L}}) \simeq \rho_{\mathrm{L}}. \tag{12.18}$$

12.2.1 Quantum MAP decoder

Given the noisy syndromes in Eqs. (12.4), what recovery should be applied? For the independent X- and Z-component noise model, the two CSS decoding problems can be treated separately. We focus on X errors. Two errors differing by an X-stabilizer are equivalent:

$$e_X \sim e_X + H_X^{\mathsf{T}} a, \qquad [e_X] = e_X + \mathrm{im}(H_X^{\mathsf{T}}). \tag{12.19}$$

The posterior probability of the equivalence class is

$$\mathbb{P}([e_X] \mid u_Z) = \sum_{\widetilde{e}\in \mathrm{im}(H_X^{\mathsf{T}})} \frac{\mathbb{P}(u_Z \mid e_X + \widetilde{e})\mathbb{P}(e_X + \widetilde{e})}{\mathbb{P}(u_Z)}. \tag{12.20}$$

Definition 12.5 (Maximum-a-posteriori decoder)**:** The MAP decoder chooses the error class

$$[\widehat{e}_X]_{\mathrm{MAP}} := \arg\max_{[e_X]} \mathbb{P}([e_X] \mid u_Z) \tag{12.21}$$

and applies any representative of that class.

The sum in Eq. (12.20) is generally difficult to evaluate. As in the classical Ising-code example, it is useful to rewrite the posterior as a disordered Ising model. We will then see that finding the ground state of a certain Ising model gives us a natural decoding strategy.

12.2.2 A disordered Ising model for one noisy syndrome snapshot

For an X-error configuration e_X, define Ising spins

$$\chi_i := 1 - 2(e_X)_i \in \{\pm 1\}. \tag{12.22}$$

The independent prior becomes

$$\mathbb{P}(e_X) = \prod_{i=1}^{n} p_{\mathrm{e}}^{(e_X)_i} (1-p_{\mathrm{e}})^{1-(e_X)_i} \propto \exp\left(h \sum_{i=1}^{n} \chi_i\right), \tag{12.23}$$

where h was defined in (4.36). Let m_Z be the number of Z-checks, and define the observed-syndrome signs

$$S_\alpha := 1 - 2(u_Z)_\alpha. \tag{12.24}$$

If $i \sim \alpha$ means that qubit i participates in the Z-check α, then

$$\mathbb{P}(u_Z \mid e_X) = \prod_{\alpha=1}^{m_Z} \begin{cases} p_{\mathrm{m}}, & (u_Z)_\alpha = 1 + (H_Z e_X)_\alpha, \\ 1 - p_{\mathrm{m}}, & (u_Z)_\alpha = (H_Z e_X)_\alpha \end{cases} \propto \exp\left(J \sum_{\alpha=1}^{m_Z} S_\alpha \prod_{i \sim \alpha} \chi_i\right) \tag{12.25}$$

where J was defined in (4.39).

The exact MAP decoder must compare the total posterior weight of whole stabilizer-equivalence classes. A weaker decoder instead chooses the most likely individual error configuration. This is the ground-state problem for

$$H_{\mathrm{easy},X} = -h \sum_{i=1}^{n} \chi_i - J \sum_{\alpha=1}^{m_Z} S_\alpha \prod_{i \sim \alpha} \chi_i. \tag{12.26}$$

The class sum in Eq. (12.20) makes exact MAP decoding harder than minimizing Eq. (12.26). Equivalently, the posterior weight of the logical class $[e_X]$ is the partition function

$$Z_{[e_X]}(u_Z) := \sum_{\widetilde{e} \in \mathrm{im}(H_X^{\mathsf{T}})} \exp\left[-H_{\mathrm{easy},X}(e_X + \widetilde{e}; u_Z)\right], \tag{12.27}$$

and the MAP decoder selects the class with maximal $Z_{[e_X]}(u_Z)$. Thus exact MAP decoding compares free energies, while the weaker decoder retains only the most likely individual configuration.

12.2.3 Problems with decoding using a single snapshot of syndromes

As depicted in Figure 12.2 single noisy syndrome snapshot can be misleading. A short physical error string produces nearby syndrome defects and is likely to be identified correctly. By contrast, if two observed defects are separated by a distance r, then the decoder compares probabilities of

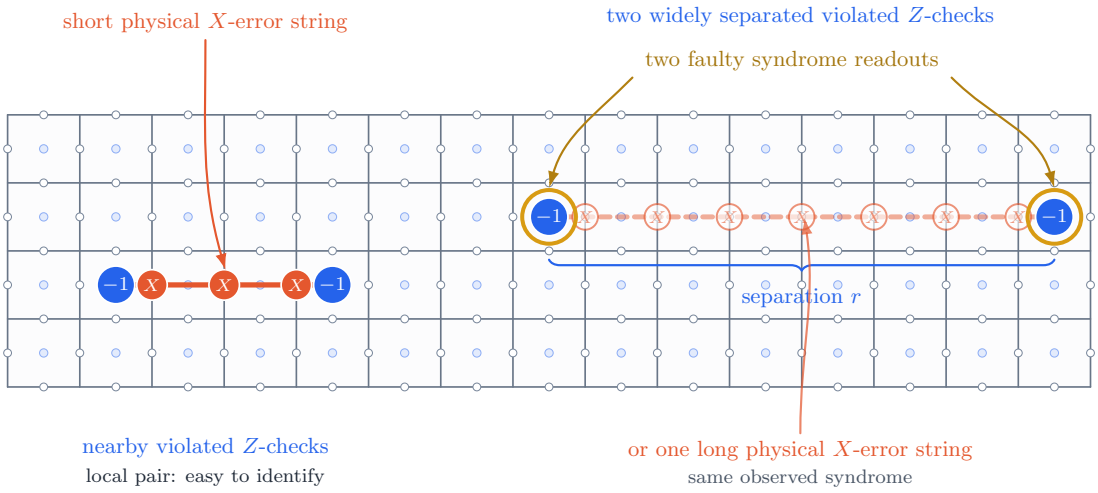

Figure 12.2: Easy and hard error configurations to decode with a MAP decoder using a single round of syndrome extraction.

the rough form

$$\mathbb{P}(\text{two bad syndrome measurements}) \sim p_{\mathrm{m}}^2, \tag{12.28a}$$

$$\mathbb{P}(\text{physical error string of length } r) \sim \left[\mathrm{O}(1)p_{\mathrm{e}}\right]^r. \tag{12.28b}$$

For sufficiently large r, a single-snapshot decoder therefore attributes the isolated defects to measurement errors and ignores them.

In a system with n qubits, one expects rare clusters whose diameter is of order $\log n$. A particular such cluster may be more likely to arise from a faulty syndrome snapshot than from data errors, so the single-round decoder ignores it. Repeating this process without retaining history allows residual clusters to accumulate and eventually produce a logical error.

12.3 Spacetime decoding and minimum-weight perfect matching

The remedy is to repeat noisy syndrome extraction many times and decode the entire spacetime history. Let $e_{X,t} \in \mathbb{F}_2^n$ denote the cumulative physical X error present at time t, and let $t_{Z,t} \in \mathbb{F}_2^{m_Z}$ denote the measurement fault at that time. The measured syndrome is

$$u_{Z,t} = t_{Z,t} + H_Z e_{X,t}. \tag{12.29}$$

Define

$$\chi_{i,t} := 1 - 2(e_{X,t})_i, \tag{12.30a}$$

$$r_{\alpha,t} := 1 - 2(u_{Z,t})_\alpha. \tag{12.30b}$$

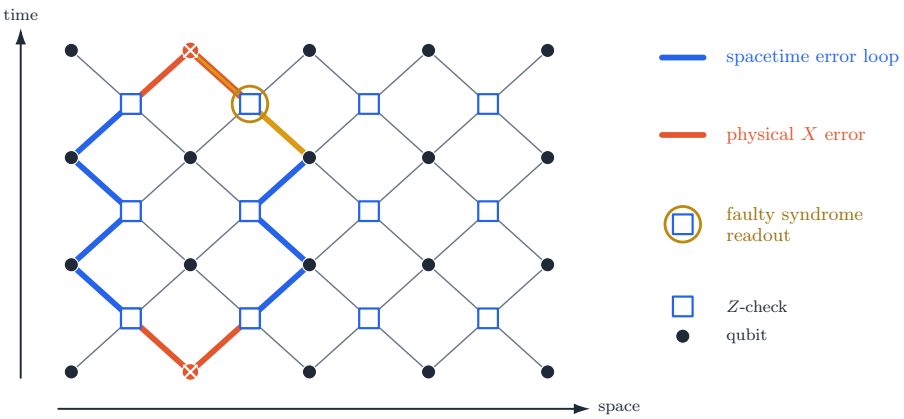

Figure 12.3: The spacetime decoding graph for (one of the two spatial dimensions of) the surface code, together with time. The MWPM decoder looks for the smallest way to close all loops.

Then the product between adjacent times detects whether a new data error occurred,

$$\chi_{i,t}\chi_{i,t-1} = 1 - 2\big(e_{X,t} + e_{x,t-1}\big)_i, \tag{12.31}$$

while the check term detects whether the syndrome was measured incorrectly,

$$r_{\alpha,t} \prod_{i\sim\alpha} \chi_{i,t} = 1 - 2(t_{Z,t})_\alpha. \tag{12.32}$$

Consequently, the most-likely-history Hamiltonian is

$$H^{\text{spacetime}}_{\text{easy},X} = -\sum_{t=1}^{T} \left[h \sum_{i=1}^{n} \chi_{i,t}\chi_{i,t-1} + J \sum_{\alpha=1}^{m_Z} r_{\alpha,t} \prod_{i\sim\alpha} \chi_{i,t} \right]. \tag{12.33}$$

For a clean initial state, one may take $\chi_{i,0} = 1$. Different values of p_{e} and p_{m} simply produce different weights h and J for spatial and temporal faults.

The detection events used by a matching decoder are changes in the measured syndrome. Defining $f_{Z,t} := u_{Z,t} + u_{Z,t-1}$ gives

$$f_{Z,t} = H_Z\big(e_{X,t} + e_{x,t-1}\big) + t_{Z,t} + t_{Z,t-1}. \tag{12.34}$$

Thus a new data fault creates spatially separated detection events in one time slice, whereas a measurement fault creates detection events at the same check in adjacent time slices. The observed events are the boundary of a fault chain in the spacetime decoding graph.

For the surface code, the decoder seeks a minimum-total-weight collection of spatial and temporal faults with the observed boundary. Equivalently, it adds a minimum-weight set of edges that closes the spacetime error chains. This is the **minimum-weight perfect-matching (MWPM)** decoder: see Figure 12.3.

Theorem 12.6 (Surface-code quantum memory below threshold)**:** For physical error rates below threshold, there are constants $\alpha, \beta > 0$ such that decoding for a time

$$T \leq \mathrm{poly}(d)\, \mathrm{e}^{\beta d} \tag{12.35}$$

leads to logical failure probability

$$\mathbb{P}(\text{logical error by time } T) \leq \mathrm{e}^{-\alpha d}. \tag{12.36}$$

Proof sketch. The spacetime decoding graph has bounded degree. If decoding produces a logical error, the union of the physical fault history and the inferred recovery contains a large nontrivial spacetime path or cycle. A Peierls argument compares the exponentially growing number of possible connected chains with the exponentially decreasing probability that at least half of a long chain is occupied by physical faults. Below threshold the latter wins, so a logical chain of length $\Omega(d)$ is exponentially unlikely. A union bound over the T possible time locations gives the range in Eq. (12.35). □

The outcome of decoding is therefore a surface code containing only correctable residual error clusters. The logical information remains protected, and the active protocol realizes a quantum memory.

Interestingly, it turns out that the MWPM decoder can run in polynomial time for the 2d surface code. In general, finding the ground state of $H_{\mathrm{easy},X}$ is a hard classical computational problem, but under our independent X and Z error model, it is surprisingly easy for the surface code. This is one advantage to using the surface code in experiment. Unfortunately, if Y errors are just as likely as X or Z errors, solving the MWPM problem turns out to be NP-hard.

Definition 12.7 (Below-threshold region)**:** For a fixed code family, decoder, noise model, and prescribed storage-time scaling $T(n)$, the below-threshold region is the set of pairs $(p_{\mathrm{e}}, p_{\mathrm{m}})$ for which

$$\lim_{n\to\infty} \mathbb{P}\big(\text{logical error by time } T(n)\big) = 0. \tag{12.37}$$

The two-dimensional surface code has one of the largest known thresholds among practical scalable quantum codes. Representative values for the phenomenological noise model are

noise model and decoder	threshold
$p_{\mathrm{m}} = 0$, MAP decoding	$p_{\mathrm{e}} \approx 0.109$
$p_{\mathrm{e}} = p_{\mathrm{m}}$, MAP decoding	$p_{\mathrm{e}} \approx 0.031$
$p_{\mathrm{e}} = p_{\mathrm{m}}$, MWPM decoding	$p_{\mathrm{e}} \approx 0.029$

(12.38)

Experimental and circuit-level analyses often quote lower values because they include a more detailed noise model for the entire syndrome-extraction circuit. Understanding how syndrome

extraction itself can spread or correlate errors is an additional part of fault-tolerant quantum error correction, which we will return to in Section 14.3.

12.4 Thresholds exist for general quantum LDPC codes

The spacetime cluster argument is not special to the surface code; it applies to any quantum LDPC code (Definition 9.7). Indeed, by similar counting to Proposition 12.1, the probability of seeing large connected error clusters in the spacetime decoding graph (generalizing Figure 12.3) is exponentially suppressed. Then we obtain:

Theorem 12.8: Consider a qLDPC code family with distance

$$d \gg \log n. \tag{12.39}$$

Under sufficiently weak local data and syndrome-measurement noise, MAP or "minimum-weight" spacetime decoding yields a quantum memory with logical failure probability exponentially small in d.

Theorem 12.8 is an information-theoretic statement and does not by itself imply an efficient decoder. The decoding problem can be much harder for general quantum codes. Still, combined with the possibility of measuring bounded-weight checks efficiently, as we discuss in Section **??**, this makes qLDPC codes leading candidates for scalable fault-tolerant quantum memories and, together with suitable logical gates and hardware architecture, for fault-tolerant quantum computation.

13 Clifford circuits

We have seen how to store logical qubits safely. We now turn toward how this is done in an experiment and, eventually, how logical gates can be applied safely. To get there, we first need a useful formalism for a particularly important family of quantum circuits, known as Clifford circuits.

13.1 Clifford group

13.1.1 Definition and elementary examples

Definition 13.1 (Clifford group)**:** Given the set $\mathrm{U}(2^n)$ of unitaries acting on n qubits, the n-qubit Clifford group is

$$\mathcal{C}_n := \left\{ U \in \mathrm{U}(2^n) : U^\dagger \mathcal{P}_n U = \mathcal{P}_n \right\}. \tag{13.1}$$

A unitary $U \in \mathcal{C}_n$ is called a **Clifford gate**.

Clifford gates preserve the structure of Pauli stabilizer codes, so they form a natural class of gates to study in quantum error correction.

Proposition 13.2: The set $\mathcal{C}_n$ is a group.

Proof. If $U_1, U_2 \in \mathcal{C}_n$, then

$$(U_2 U_1)^\dagger \mathcal{P}_n (U_2 U_1) = U_1^\dagger (U_2^\dagger \mathcal{P}_n U_2) U_1 = U_1^\dagger \mathcal{P}_n U_1 = \mathcal{P}_n. \tag{13.2}$$

The identity and inverse properties follow directly from Eq. (13.1). □

Proposition 13.3: Every Pauli operator is a Clifford gate:

$$\mathcal{P}_n \subseteq \mathcal{C}_n. \tag{13.3}$$

Proof. For Hermitian Pauli operators $P, Q \in \mathcal{P}_n$, as in Proposition 7.9,

$$Q^\dagger P Q = \eta P, \qquad \eta \in \{\pm 1\}. \tag{13.4}$$

Hence conjugation by a Pauli maps the Pauli group to itself. □

Example 13.4 (Hadamard gate)**:** On one qubit, define the **Hadamard gate**

$$H := \frac{X + Z}{\sqrt{2}}. \tag{13.5}$$

This operator is unitary, and

$$H^\dagger X H = \frac{1}{2}(X + Z) X (X + Z) = Z, \tag{13.6a}$$

$$H^\dagger Z H = X. \tag{13.6b}$$

Thus H permutes Pauli operators and belongs to $\mathcal{C}_1$.

13.1.2 Symplectic action on Pauli operators

Recall that, after discarding an overall Pauli phase, an n-qubit Pauli operator is represented by a binary vector (Definition 8.1); multiplication of Pauli operators becomes addition of their binary vectors:

$$\varphi(P_1 P_2) = \varphi(P_1) + \varphi(P_2). \tag{13.7}$$

If $U \in \mathcal{C}_n$, then

$$(U^\dagger P_1 U)(U^\dagger P_2 U) = U^\dagger (P_1 P_2) U. \tag{13.8}$$

Combining (13.7) and (13.8), we see that conjugation by U acts linearly on the binary representation:

$$\varphi(U^\dagger P U) = M_U \varphi(P), \qquad M_U \in \mathbb{F}_2^{2n\times 2n}. \tag{13.9}$$

The commutation relation between two Pauli operators is encoded by the symplectic form J, defined in (8.8):

$$P_1 P_2 = (-1)^{\varphi(P_1)^{\mathsf{T}} J \varphi(P_2)} P_2 P_1, \tag{13.10}$$

Unitary conjugation preserves commutation and anticommutation, so

$$M_U^{\mathsf{T}} J M_U = J. \tag{13.11}$$

Thus we arrive at:

Proposition 13.5: M_U is an element of the **symplectic group** of matrices over $\mathbb{F}_2$:

$$M_U \in \mathrm{Sp}(2n, \mathbb{F}_2) := \left\{ M \in \mathbb{F}_2^{2n\times 2n} : M^{\mathsf{T}} J M = J \right\}. \tag{13.12}$$

For the Hadamard gate,

$$M_H = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \in \mathrm{Sp}(2, \mathbb{F}_2). \tag{13.13}$$

Example 13.6 (S gate)**:** A second single-qubit Clifford gate is

$$S := \mathrm{e}^{-\mathrm{i}\pi Z/4}, \tag{13.14a}$$

$$M_S = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}. \tag{13.14b}$$

Indeed,

$$S^\dagger X S = -Y, \tag{13.15a}$$
$$S^\dagger Y S = X, \tag{13.15b}$$
$$S^\dagger Z S = Z. \tag{13.15c}$$

Example 13.7 (CNOT gate on 2 qubits): For $n > 1$, there are also entangling Clifford gates. With qubit i as the control and qubit j as the target, define

$$\mathrm{CNOT}_{i\to j} := \frac{\mathbb{I} + Z_i}{2} + \frac{\mathbb{I} - Z_i}{2} X_j. \tag{13.16}$$

For example, it creates a Bell pair from a product state:

$$\mathrm{CNOT}_{1\to 2}\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle\right) = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{13.17}$$

A direct calculation gives

$$\mathrm{CNOT}_{i\to j}^\dagger\, Z_i\, \mathrm{CNOT}_{i\to j} = Z_i, \tag{13.18a}$$
$$\mathrm{CNOT}_{i\to j}^\dagger\, X_i\, \mathrm{CNOT}_{i\to j} = X_i X_j, \tag{13.18b}$$
$$\mathrm{CNOT}_{i\to j}^\dagger\, X_j\, \mathrm{CNOT}_{i\to j} = X_j, \tag{13.18c}$$
$$\mathrm{CNOT}_{i\to j}^\dagger\, Z_j\, \mathrm{CNOT}_{i\to j} = Z_i Z_j. \tag{13.18d}$$

For two qubits, ordered as (x_1, x_2, z_1, z_2), this becomes

$$M_{\mathrm{CNOT}_{1\to 2}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{13.19}$$

A closely related gate is the CZ gate (Problem 13.1).

13.1.3 Generators and classification

We identify Clifford unitaries that differ by a Pauli operator and a global phase. Equivalently,

$$U \sim V \quad\Longleftrightarrow\quad U = \mathrm{e}^{\mathrm{i}\theta} P V, \qquad P \in \mathcal{P}_n, \tag{13.20}$$
$$\widehat{\mathcal{C}}_n := \mathcal{C}_n / (\mathrm{U}(1)\mathcal{P}_n). \tag{13.21}$$

Theorem 13.8 (Classification of Clifford gates)**:** The quotient $\widehat{\mathcal{C}}_n$ is generated by the images of $\langle H_i,\ S_i,\ \mathrm{CNOT}_{i\to j}\rangle$. Moreover,

$$\widehat{\mathcal{C}}_n \cong \mathrm{Sp}(2n, \mathbb{F}_2). \tag{13.22}$$

In other words, every binary symplectic transformation is realized by a Clifford circuit.

Proof sketch. The proof is a little tedious; we only show part of it. The first step is to show that if $M_U = \mathbb{I}$, then $U \in \mathrm{U}(1)\mathcal{P}_n$.

Lemma 13.9: Suppose that $U \in \mathcal{C}_n$ obeys

$$U^\dagger P(v) U = (-1)^{\ell(v)} P(v) \tag{13.23}$$

for every Hermitian Pauli representative $P(v)$, where $v \in \mathbb{F}_2^{2n}$. Then U is a Pauli operator up to a global phase.

Proof. Compatibility with Pauli multiplication makes ℓ linear. Nondegeneracy of the symplectic form then gives an $r \in \mathbb{F}_2^{2n}$ such that

$$\ell(v+w) = \ell(v) + \ell(w), \qquad \ell(v) = r^{\mathsf{T}} J v. \tag{13.24}$$

Conjugation by $P(r)$ produces exactly the same signs. Hence $P(r)^\dagger U$ commutes with every Pauli operator and is therefore a proportional to $\mathbb{I}$, i.e. $U \in \mathrm{U}(1)\mathcal{P}_n$. □

The second step of the proof is to show that every $M \in \mathrm{Sp}(2n, \mathbb{F}_2)$ can be generated by the gates H_i, S_i, and $\mathrm{CNOT}_{i\to j}$. Since we know that every $U \in \mathcal{C}_n$ corresponds to an $M_U \in \mathrm{Sp}(2n, \mathbb{F}_2)$, showing that all symplectic matrices can be built both shows that H, S and CNOT generate the Clifford group, and (13.22). The essence of the proof (which is tedious, thus omitted) is to perform Gaussian elimination over $\mathbb{F}_2$, using a set of elementary steps $A_1, \ldots, A_k \in \{H_i, S_i, \mathrm{CNOT}_{i\to j}\}$ to reduce an arbitrary $M \in \mathrm{Sp}(2n, \mathbb{F}_2)$ to the identity by a sequence of transformations:

$$M_k M_{k-1} \cdots M_1 M = \mathbb{I}_{2n}. \tag{13.25}$$

Reversing the sequence constructs a Clifford circuit realizing M. □

13.2 Gottesman–Knill theorem

Definition 13.10 (Stabilizer mixed state)**:** Let

$$\mathcal{S} = \langle S_1, \ldots, S_m \rangle \tag{13.26}$$

be generated by m independent, mutually commuting Hermitian Pauli operators. For $\boldsymbol{\eta} \in \{\pm 1\}^m$, define

$$\rho_{\boldsymbol{\eta}} := 2^{m-n} \prod_{i=1}^{m} \frac{\mathbb{I} + \eta_i S_i}{2}. \tag{13.27}$$

The product in Eq. (13.27) projects onto the joint eigenspace with S_i-eigenvalue η_i. That eigenspace has dimension 2^{n-m}, so the factor 2^{m-n} normalizes the state. When $m = n$, the state is pure; when $m < n$, it is maximally mixed within the joint eigenspace.

Theorem 13.11 (Gottesman–Knill theorem)**:** There is a polynomial-time classical algorithm for simulating the dynamics of stabilizer states under Clifford circuits, including Pauli measurements and classical feedforward.

Proof. We construct this algorithm directly from the binary formalism.

Algorithm 13.12 (Stabilizer update under Clifford gate)**:** A Clifford gate U evolves the stabilizer state as

$$U \rho_{\boldsymbol{\eta}} U^\dagger = 2^{m-n} \prod_{i=1}^{m} \frac{\mathbb{I} + \eta_i (U S_i U^\dagger)}{2}. \tag{13.28}$$

Store the stabilizer generators in the binary matrix

$$\mathsf{S} := \begin{pmatrix} \varphi(S_1) & \cdots & \varphi(S_m) \end{pmatrix} \in \mathbb{F}_2^{2n \times m}, \tag{13.29}$$

together with the eigenvalue vector $\boldsymbol{\eta} \in \{\pm 1\}^m$. For each generator, write

$$U S_i U^\dagger = \sigma_i S_i', \qquad \varphi(S_i') = M_{U^\dagger} \varphi(S_i), \qquad \sigma_i \in \{\pm 1\}. \tag{13.30}$$

The update rule is therefore

$$\mathsf{S} \longmapsto M_{U^\dagger} \mathsf{S}, \tag{13.31a}$$

$$\eta_i \longmapsto \sigma_i \eta_i. \tag{13.31b}$$

The signs must be retained: for example, a Clifford conjugation may map a Hermitian Pauli generator to minus another Hermitian Pauli generator.

Algorithm 13.13 (Stabilizer update under a Pauli measurement)**:** If P is generated by the current stabilizers up to sign, write

$$P = \sigma \prod_{i=1}^{m} S_i^{a_i}, \qquad \sigma \in \{\pm 1\}, \qquad a_i \in \mathbb{F}_2. \tag{13.32}$$

Then the outcome is deterministic, with eigenvalue

$$\eta_P = \sigma \prod_{i=1}^{m} \eta_i^{a_i}. \tag{13.33}$$

Otherwise, perform both of the following steps.

1. Reorder the generators so that $S_1, \ldots, S_r$ anticommute with P and $S_{r+1}, \ldots, S_m$ commute with P, where r is allowed to be zero. If $r > 0$, discard S_1 and replace every other anticommuting generator by $S_1 S_j$. In binary form,

$$\mathsf{S}' := (\varphi(S_2) + \varphi(S_1), \ldots, \varphi(S_r) + \varphi(S_1), \varphi(S_{r+1}), \ldots, \varphi(S_m)), \tag{13.34}$$

   and

$$\boldsymbol{\eta}' := (\eta_2 \eta_1, \ldots, \eta_r \eta_1, \eta_{r+1}, \ldots, \eta_m)^{\mathsf{T}}. \tag{13.35}$$

   If $r = 0$, this first step does nothing, and we set $\mathsf{S}' = \mathsf{S}$ and $\boldsymbol{\eta}' = \boldsymbol{\eta}$.

2. Append the measured Pauli as a new stabilizer:

$$\mathsf{S}_{\text{new}} = (\mathsf{S}' \mid \varphi(P)), \tag{13.36a}$$

$$\boldsymbol{\eta}_{\text{new}} = \begin{pmatrix} \boldsymbol{\eta}' \\ \eta_P \end{pmatrix}, \qquad \mathbb{P}(\eta_P = +1) = \mathbb{P}(\eta_P = -1) = \frac{1}{2}. \tag{13.36b}$$

The two update steps in Algorithm 13.13 are independent. If $r = 0$, the measurement simply adds a new commuting stabilizer. If $r > 0$, one old generator is removed before P is appended, so the number of independent stabilizers is unchanged. Multiplying S_j by the pivot S_1 makes it commute with P and changes its eigenvalue from η_j to $\eta_1 \eta_j$.

Apply Algorithms 13.12 and 13.13 as needed to simulate a Clifford circuit with measurements. Moreover, there is no reason that subsequent gates/measurements cannot depend on the knowledge of prior measurement outcomes – the algorithms work exactly the same way. All of these operations on S and $\boldsymbol{\eta}$ are binary matrix manipulations and require only polynomial classical resources. □

The important point is that even highly entangled stabilizer states, especially those relevant for studying the dynamics of Pauli stabilizer codes, can be simulated efficiently. It is helpful to see the abstract procedure above done in practice, so we present a simple example here:

Example 13.14 (Measuring one qubit of a Bell pair)**:** Start from $|00\rangle$, whose stabilizers are Z_1 and Z_2. Consider the following sequence of gates, which prepare an entangled Bell pair and subsequently

collapse the state via measurement:

$$|\psi_0\rangle = |00\rangle\,, \qquad \mathsf{S}_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad (S_1, S_2) = (Z_1, Z_2) \tag{13.37a}$$

$$H_1|\psi_0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle, \qquad \mathsf{S}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}, \qquad (S_1, S_2) = (X_1, Z_2). \tag{13.37b}$$

$$\mathrm{CNOT}_{1\to 2} H_1|\psi_0\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \qquad \mathsf{S}_2 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, \qquad (S_1, S_2) = (X_1X_2, Z_1Z_2). \tag{13.37c}$$

Measuring Z_2 removes the anticommuting generator X_1X_2 and appends Z_2, giving

$$|00\rangle \text{ or } |11\rangle, \qquad \mathsf{S} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}, \qquad (S_1, S_2) = (Z_1Z_2, Z_2), \tag{13.38a}$$

$$\boldsymbol{\eta}_{\text{out}} = \begin{pmatrix} 1 \\ \eta_{Z_2} \end{pmatrix}, \qquad \mathbb{P}(\eta_{Z_2} = \pm 1) = \frac{1}{2}. \tag{13.38b}$$

The two outcomes produce $|00\rangle$ and $|11\rangle$ if $\eta_{Z_2} = +1$ or -1, respectively.

Ultimately, the Gottesman–Knill theorem, together with the "digitization" of errors into Pauli errors when measuring the checks of a stabilizer code, explains why large Clifford syndrome-extraction circuits can be simulated classically. Until we measure stabilizers, coherent errors may require a complicated simulation to track; usually we assume that the errors are independent and simple, as in (12.1), so the simulation is tractable. If the rows of a Pauli parity-check matrix H describe the stabilizer checks of a quantum code, then

$$\mathsf{S} = H^{\mathsf{T}}, \qquad \eta_a = (-1)^{s_a}, \tag{13.39}$$

and the measured eigenvalue vector records the syndrome $\boldsymbol{s}$. This permits large-scale simulations of QEC dynamics, for example when estimating an error threshold. Using only the $n-k$ check generators describes a state that is maximally mixed on the k logical qubits. To simulate a particular encoded stabilizer state, one appends k commuting logical stabilizers.

13.3 Pauli frame tracking

The Gottesman–Knill formalism suggests a useful experimental simplification.

Definition 13.15 (Pauli frame tracking)**:** After noisy syndrome extraction, a decoder selects Pauli recovery R. Rather than applying R physically, one stores it in classical software and reinterprets all later Clifford gates and measurement outcomes relative to this **Pauli frame**.

This is the same idea that was used implicitly in spacetime decoding in Section 12.3. The frame records the recovery selected by the decoder, not the unknown physical error itself. Let

$$r := \varphi(R) \in \mathbb{F}_2^{2n} \tag{13.40}$$

be the binary representation of the current Pauli frame. Under a Clifford gate U, the virtual recovery transforms as

$$R \longmapsto URU^\dagger, \tag{13.41a}$$

$$r \longmapsto M_{U^\dagger} r. \tag{13.41b}$$

If a Hermitian Pauli P is then measured, conjugation by the frame gives

$$R^\dagger P R = (-1)^{\varphi(P)^\mathsf{T} J r} P. \tag{13.42}$$

Consequently the corrected interpretation of a raw measurement eigenvalue is

$$\eta_{\text{corr}} = \eta_{\text{raw}} (-1)^{\varphi(P)^\mathsf{T} J r}. \tag{13.43}$$

Thus a known Pauli correction need not be implemented as a physical gate: it can be propagated through the Clifford circuit and absorbed into classical post-processing of later measurements.

Problems

Problem 13.1 (CZ gate)**:** In Rydberg atom arrays, the most natural entangling gate corresponds to a CZ gate. Between qubits i and j, the CZ gate is defined as:

$$\mathrm{CZ}_{ij} = \mathbb{I} - 2|1\rangle\langle 1|_i \otimes |1\rangle\langle 1|_j. \tag{13.44}$$

A: Show that CZ_{ij} is a Clifford gate. Show explicitly how CZ_{ij} transforms all single-qubit Pauli X and Z operators. In addition and equivalently, write the 4×4 symplectic $\mathbb{F}_2$ matrix $M_{\mathrm{CZ}_{12}}$.

B: We will want to understand how to prepare the GHZ state

$$|\mathrm{GHZ}\rangle := \frac{|000\cdots\rangle + |111\cdots\rangle}{\sqrt{2}} \tag{13.45}$$

with Rydberg atoms. Find an explicit set of n commuting stabilizers that stabilize $|\text{GHZ}\rangle$.

C: Using only experimentally accessible single-qubit rotation gates (H and S on any one qubit), and CZ gates, find a circuit that prepare the GHZ state, starting from the initial state $|00\cdots\rangle$. Explain how to use the stabilizer formalism in your analysis to track the evolution of the state through the circuit.

14 Syndrome extraction circuits

This lecture explains how to measure the syndromes of a CSS code using quantum circuits, and how faults inside the syndrome-extraction circuit complicate fault tolerance.

14.1 Quantum circuit notation

Quantum circuit diagrams are often more useful than long products of unitaries. The basic conventions used below are summarized in Figure 14.1. The most common gates that we use in a quantum circuit are single-qubit rotation gates and CNOT gates (Example 13.7), and the notation for these specific gates is depicted in Figure 14.2.

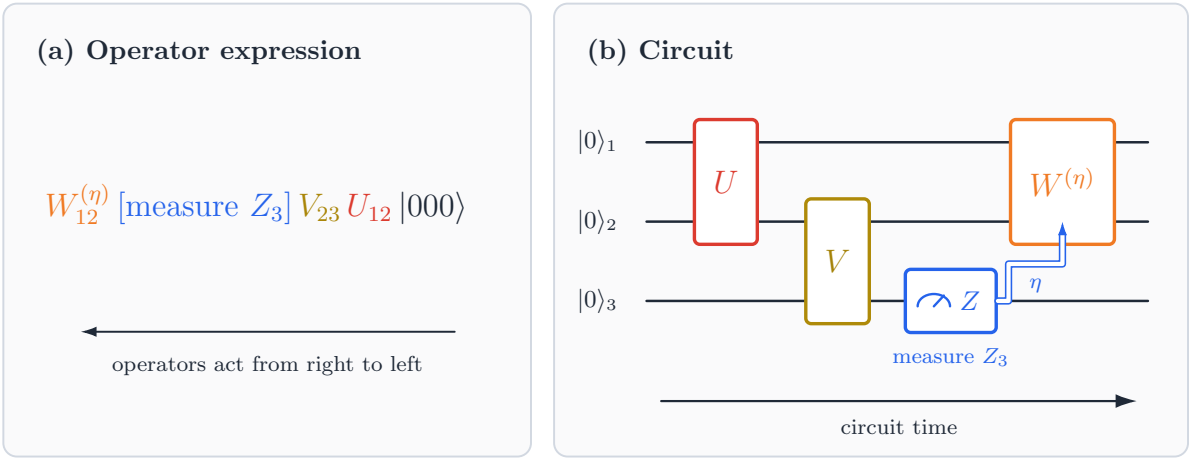

Figure 14.1: Converting a product of unitary operators (read right to left, when acting on a ket), into a quantum circuit (where time is read from left to right). Here $\eta \in \{\pm 1\}$ denotes the measurement outcome after measuring Z_3. Quantum circuits can involve unitary operators/gates, measurement, and feedforward of the measurement outcome into future unitaries.

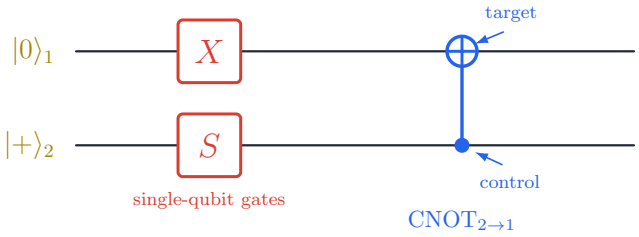

Figure 14.2: Denoting single-qubit vs. the entangling CNOT gate in a quantum circuit.

14.2 Ancilla-assisted syndrome extraction

To measure a parity-check such as $Z_1Z_2Z_3Z_4$, we must not measure the four operators Z_1, Z_2, Z_3, Z_4 separately: doing so would reveal more information than the syndrome and would destroy coherence within a fixed syndrome subspace. We will instead use an ancilla qubit and assume that the native entangling operations are two-qubit gates. By entangling the physical qubits with an ancilla, we will effectively be able to measure a higher-weight Pauli operator than we may natively have access to in our quantum hardware. We will often use the following jargon:

Definition 14.1 (Data/ancilla qubits)**:** A **data qubit** (or physical qubit) is one of the n qubits whose entangled state makes up the quantum CSS code. An **ancilla qubit** is anything else.

Usually, ancilla qubits will be repeatedly measured, and as such "reset" into simple states unentangled with the data qubits. We'll see how that works explicitly soon.

We will repeatedly use the propagation of Pauli operators through a CNOT gate. For a CNOT with control qubit 1 and target qubit 2, we recall from Example 13.7 that

$$\mathrm{CNOT}_{1\to 2}\begin{pmatrix} X_1 \\ X_2 \\ Z_1 \\ Z_2 \end{pmatrix}\mathrm{CNOT}_{1\to 2}^{\dagger} = \begin{pmatrix} X_1X_2 \\ X_2 \\ Z_1 \\ Z_1Z_2 \end{pmatrix}. \tag{14.1}$$

14.2.1 Measuring a Z-type check

Algorithm 14.2 (Z-check syndrome extraction)**:** To measure the check $Z_1Z_2Z_3Z_4$, prepare an ancilla a in $|0\rangle_a$, apply CNOTs from each data qubit to the ancilla, and finally measure Z_a, as shown in the circuit in Figure 14.3.

Let

$$U_Z = \mathrm{CNOT}_{4\to a}\mathrm{CNOT}_{3\to a}\mathrm{CNOT}_{2\to a}\mathrm{CNOT}_{1\to a} \tag{14.2}$$

be the unitary part of the extraction circuit. The ancilla begins in a known state stabilized by Z_a. Evolving the measured ancilla operator backward through the circuit gives

$$U_Z^{\dagger} Z_a U_Z = Z_1Z_2Z_3Z_4Z_a. \tag{14.3}$$

Since the initial ancilla has $Z_a = +1$, the final measurement of Z_a reports the eigenvalue of $Z_1Z_2Z_3Z_4$ on the data qubits. Equivalently, outcome $m \in \mathbb{F}_2$ applies the projective measurement operator

$$P_m^{(Z)} = \frac{\mathbb{I} + (-1)^m Z_1Z_2Z_3Z_4}{2} \tag{14.4}$$

to the data. Thus the circuit measures only the check eigenvalue and preserves coherence within

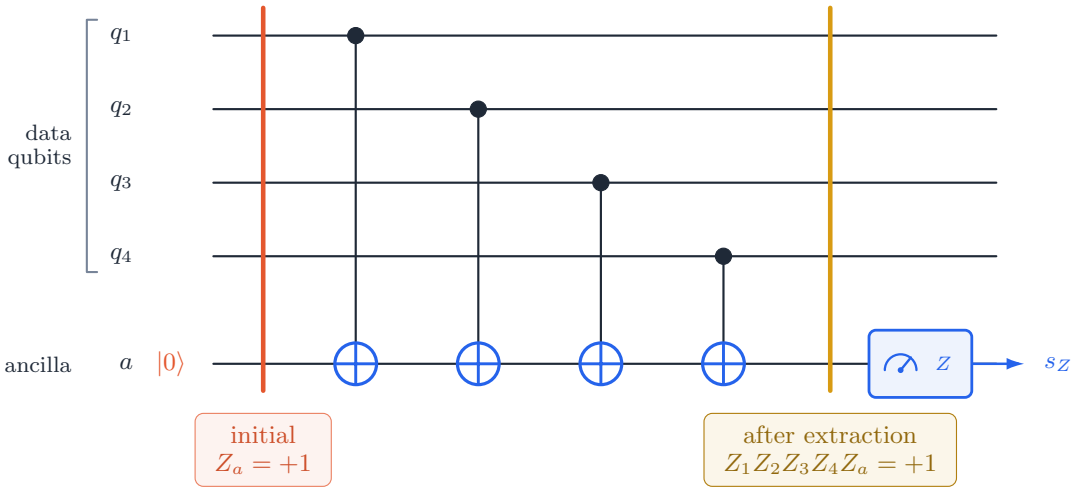

Figure 14.3: Quantum circuit using only CNOT gates and measurement to measure the stabilizer $Z_1Z_2Z_3Z_4$.

each of its two eigenspaces.

Example 14.3: Consider the initial state

$$|\psi_0\rangle = \frac{1}{2}\left(|0000\rangle + |1111\rangle + |0001\rangle + |1110\rangle\right) \otimes |0\rangle_a \,. \tag{14.5}$$

The first two data states have check eigenvalue $+1$, while the last two have check eigenvalue -1. After measuring the ancilla, the final state is

$$|\psi_{\text{final}}\rangle = \begin{cases} \frac{1}{\sqrt{2}}\left(|0000\rangle + |1111\rangle\right) \otimes |0\rangle_a \,, & \text{with probability } \frac{1}{2}, \\ \frac{1}{\sqrt{2}}\left(|0001\rangle + |1110\rangle\right) \otimes |1\rangle_a \,, & \text{with probability } \frac{1}{2}. \end{cases} \tag{14.6}$$

This is error digitization: the measurement collapses the data into a definite syndrome eigenspace without resolving the individual computational-basis states inside that eigenspace. The same circuit measures a Z-type check of any weight by adding one data-to-ancilla CNOT for each qubit in its support.

Lastly, although we do not write it out explicitly, it is straightforward to modify Algorithm 14.2 to extract the Z-check syndrome for a Z-check of any weight, by adding one CNOT gate for each physical qubit involved in the check.

14.2.2 Measuring an X-type check

Algorithm 14.4 (X-check syndrome extraction)**:** Prepare an ancilla a' in $|+\rangle_{a'}$, and use the ancilla as the control of four CNOTs whose targets are the data qubits, measuring $X_{a'}$ at the end: see

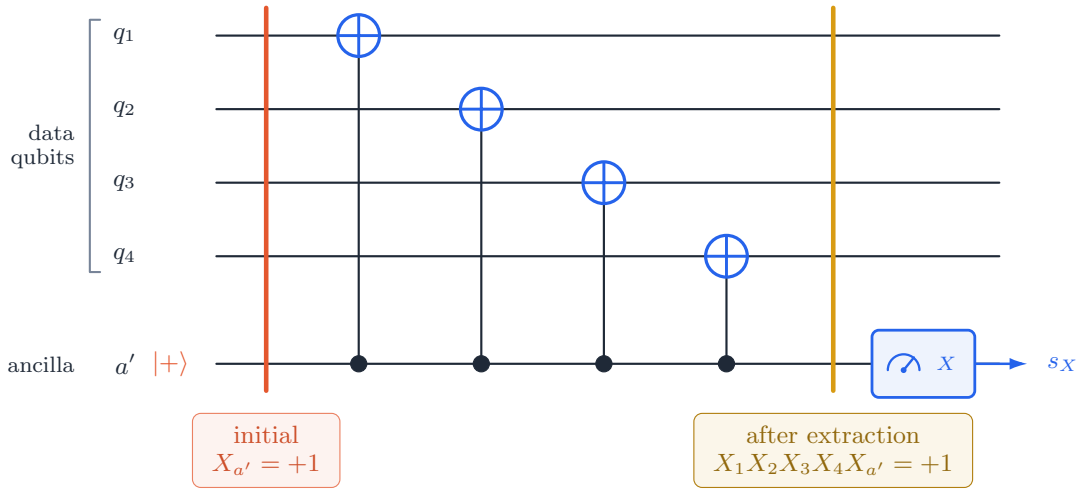

Figure 14.4: Quantum circuit using only CNOT gates and measurement to measure the stabilizer $X_1X_2X_3X_4$.

Figure 14.4.

This is the "dual" of the Z-check syndrome extraction circuit in Algorithm 14.2.

Writing

$$U_X = \mathrm{CNOT}_{a'\to 4}\mathrm{CNOT}_{a'\to 3}\mathrm{CNOT}_{a'\to 2}\mathrm{CNOT}_{a'\to 1}, \tag{14.7}$$

we find

$$U_X^\dagger X_{a'} U_X = X_1X_2X_3X_4X_{a'}. \tag{14.8}$$

The initial ancilla is stabilized by $X_{a'}$, so the final measurement of $X_{a'}$ reports the eigenvalue of $X_1X_2X_3X_4$ and collapses the data into the corresponding syndrome eigenspace. Importantly, both the X- and Z-check extraction circuits use only two-qubit entangling gates.

14.3 Circuit-level fault tolerance

We now ask how faults propagate through these circuits. It is enough to focus on the Z-check circuit; the X-check story follows by exchanging X and Z and reversing the CNOT direction.

14.3.1 Circuit-level noise

Consider the candidate errors occurring on either physical/data or ancilla qubits in Figure 14.5. Suppose that an error E_1 acts on data qubit 1 immediately after that qubit has interacted with the ancilla. If $E_1 = Z_1$, it commutes with the Z-type check and therefore has no effect on this check outcome, although it can be detected later by an X-type syndrome measurement. If $E_1 = X_1$, the

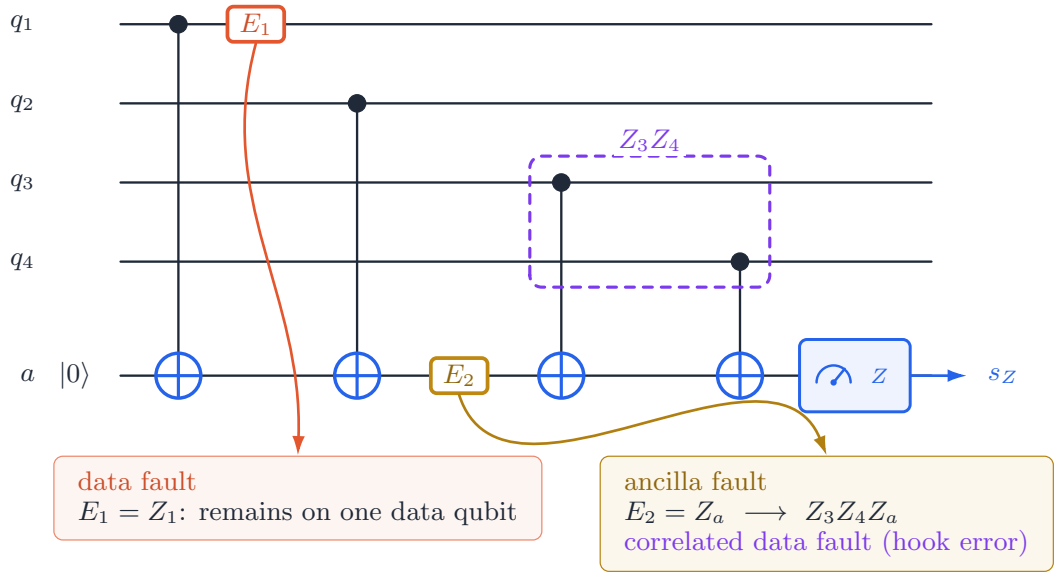

Figure 14.5: The Z-check syndrome extraction circuit from Figure 14.3, but now with some errors occuring on data (E_1) or ancilla (E_2) qubits.

ancilla has already interacted with qubit 1, so this fault is not detected in the present extraction round, but it can be detected in the next round of syndrome-extraction; therefore it is also not very dangerous (we can't tell it apart from a genuine error that occurred in between the two rounds of syndrome extraction). A useful way to see the latter statement is to push the error backward through the preceding CNOT:

$$X_1 \mathrm{CNOT}_{1\to a} = \mathrm{CNOT}_{1\to a} X_1 X_a. \tag{14.9}$$

Thus the same faulty circuit can be represented as an X_1 data error together with an X_a ancilla error at the beginning of the circuit.

Now suppose that an error E_2 acts on the ancilla during extraction. An X_a error on the CNOT target does not propagate to the data, but it flips the final Z_a measurement outcome. By contrast, a Z_a error propagates through each subsequent CNOT and can generate a correlated Z error on several data qubits.

Definition 14.5 (Hook error)**:** A correlated data qubit error produced by one otherwise uncorrelated error during the syndrome extraction circuit is called a **hook error**.

For example, if the CNOT order is $1 \to 2 \to 3 \to 4$ and a Z_a fault occurs after the second interaction, propagation through the two remaining CNOTs gives

$$Z_a \longmapsto Z_3 Z_4 Z_a. \tag{14.10}$$

The ancilla factor is harmless on the final Z-basis measurement, while the data error Z_3Z_4 is equivalent to Z_1Z_2 modulo the measured stabilizer $Z_1Z_2Z_3Z_4$. This middle-of-the-circuit fault is the genuinely dangerous weight-two hook. A Y_a fault contains both dangerous components: it can produce a hook error and flip the syndrome outcome.

More generally, consider a weight-w Z-check measured in the order $1, 2, \ldots, w$. If a Z_a fault occurs after the kth interaction, its data component at the end of the circuit is

$$E_k = Z_{k+1}Z_{k+2}\cdots Z_w. \tag{14.11}$$

Multiplying by the measured stabilizer gives the equivalent representative

$$E_k \sim (Z_1Z_2\cdots Z_w)\, E_k = Z_1Z_2\cdots Z_k. \tag{14.12}$$

Hence the effective weight of the hook satisfies

$$\mathrm{wt}_{\mathrm{eff}}(E_k) \le \min\{k, w-k\} \le \left\lfloor \frac{w}{2} \right\rfloor. \tag{14.13}$$

For a check of maximum weight Δ_{C}, a single ancilla fault can therefore produce a correlated data error of effective weight as large as $\lfloor \Delta_{\mathrm{C}}/2 \rfloor$.

Circuit-level fault tolerance is consequently an engineering problem involving the code, extraction circuit, noise model, and decoder. Thresholds are usually estimated by Monte Carlo sampling of circuit faults, with Clifford propagation efficiently simulated using the stabilizer formalism. For a standard depolarizing circuit-noise model and MWPM decoding, the surface-code threshold is of order

$$p_{\mathrm{th}}^{\mathrm{circuit}} \sim 0.01, \tag{14.14}$$

rather than the roughly 0.03 threshold of the phenomenological model – recall (12.38). The precise numerical value depends on the extraction circuit, relative error rates of its locations, and decoder; the canonical "one percent" statement should be understood at this level of precision.

14.3.2 Hook orientation in the rotated surface code

Hook errors can reduce the number of circuit faults required to build a logical operator. Figure 14.6 recalls the two logical directions of the rotated surface code and compares two CNOT orders for a Z-type check.

For the ordering $1 \to 2 \to 3 \to 4$, a Z-hook can be aligned with the direction of a minimum-weight logical Z. A logical string of weight d can then be assembled from only about $d/2$ faulty circuit locations. More precisely, this bad orientation exhibits a logical fault path containing only

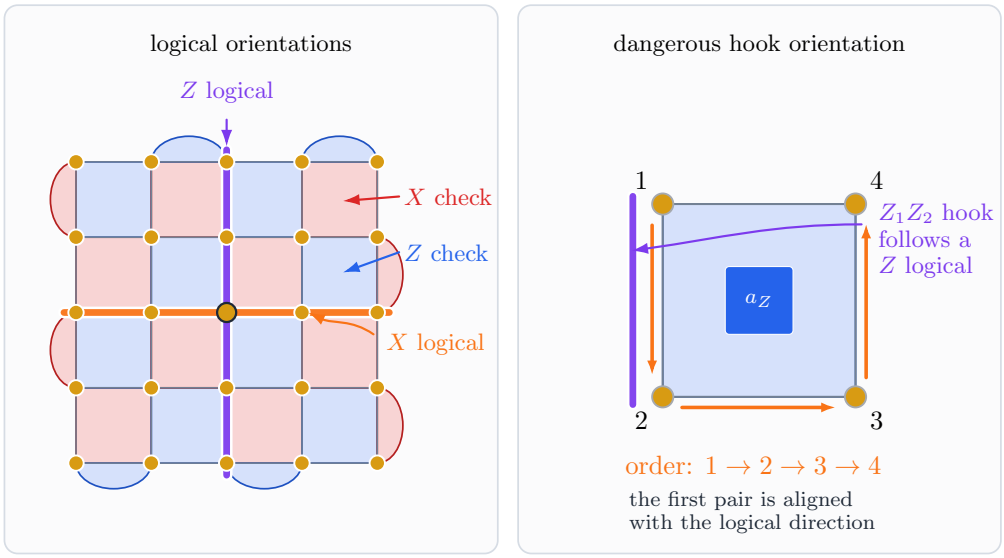

Figure 14.6: Hook errors in the rotated surface code.

$\lceil d/2 \rceil$ hook faults, so the circuit distance can obey

$$d_{\text{circ}} \leq \left\lceil \frac{d}{2} \right\rceil \tag{14.15}$$

even though the static code distance remains d. Choosing instead the order $1 \to 4 \to 3 \to 2$ rotates the two-qubit Z-hook so that it is not aligned with any minimum-weight logical Z. The X-check circuits should be ordered by the dual rule: X-hooks must be transverse to minimum-weight logical X operators.

14.3.3 Global CNOT scheduling

There is a second constraint. Ideally, all weight-four checks are measured in four CNOT layers, with no qubit participating in more than one gate in a given layer. The local orders around different checks cannot be chosen independently, because neighboring X- and Z-checks can share two data qubits.

Figure 14.7 shows a bad and a good ordering for one such pair of checks. In the bad circuit, the relative order of the X-check and Z-check interactions is opposite on the two shared data qubits. The final Z-ancilla measurement then depends on the state of the X ancilla and does not cleanly measure Z_1Z_2. In the good circuit, the X ancilla flips both shared data qubits together, leaving their Z_1Z_2 parity invariant.

Pulling the final Z-ancilla observable backward makes the distinction explicit:

$$U_{\text{bad}}^{\dagger} Z_{a_Z} U_{\text{bad}} = Z_1 Z_2 Z_{a_X} Z_{a_Z}, \tag{14.16a}$$

$$U_{\text{good}}^{\dagger} Z_{a_Z} U_{\text{good}} = Z_1 Z_2 Z_{a_Z}. \tag{14.16b}$$

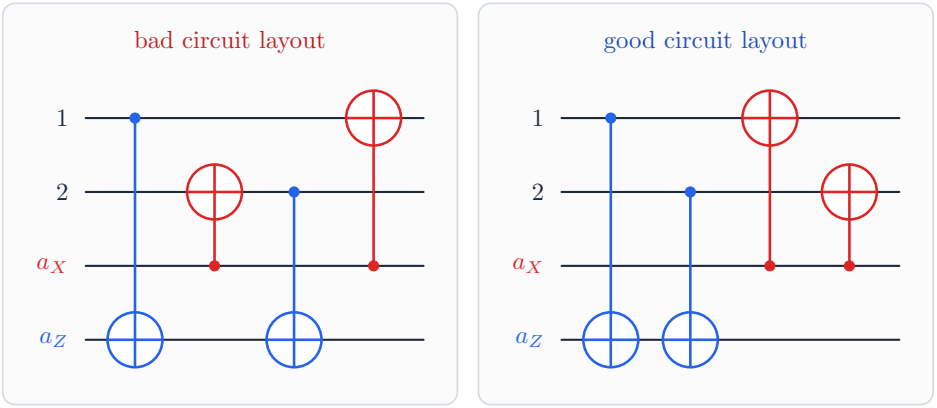

Figure 14.7: Good vs. bad schedules for CNOT gates when extracting the syndrome between two qubits 1 and 2 that have both an X-check and a Z-check in common.

The extra Z_{a_X} in the bad schedule is not a stabilizer of the initial $|+\rangle_{a_X}$ state. A useful general rule is that, for two commuting checks sharing multiple data qubits, their ancillas should interact in the same relative order on every shared qubit. Finding an optimized circuit is a design problem: see Problem 14.1.

One may try to extract all X syndromes and then all Z syndromes. This avoids the interleaving constraint, at the cost of a deeper circuit and additional idle fault locations. Practical threshold estimates near Eq. (14.14) refer to optimized, interleaved extraction circuits.

14.3.4 Bounded-weight checks and general qLDPC codes

The same considerations suggest why fault-tolerant extraction is possible in principle for a general qLDPC code. Let Δ_B be the maximum number of checks incident on a data qubit, and let Δ_C be the maximum check weight. Bounded Δ_B permits all checks to be measured in a bounded number of CNOT layers, limiting errors accumulated while qubits idle. Bounded Δ_C limits the number of data qubits reached by a single ancilla fault.

The correlated hook should not be modeled as $O(\Delta_C)$ independent errors: the entire correlated event still arises from one fault with probability $O(p_{\text{circ}})$. Rather, one fault has bounded spread $r = O(\Delta_C)$, so a logical operator of weight d requires, potentially, only d/Δ_C suitably arranged faults. Schematically, under local stochastic circuit noise and with a suitable circuit-level decoder,

$$\mathbb{P}[\text{logical failure}] \lesssim \text{poly}(n) p_{\text{circ}}^{d/\Delta_C}. \tag{14.17}$$

For a qLDPC family with fixed Δ_B and Δ_C and distance $d \gtrsim \log n$, we expect exponential suppression in d. Thus bounded check weight is essential not only for measurement depth but also for controlling fault propagation. Increasing Δ_C generally worsens the circuit-level constants and threshold.

This schematic discussion assumes that the Tanner graph interactions can be implemented without routing depth that grows with n and that the decoder accounts for correlated data and measurement faults. These are non-trivial issues to sort out in actual experiments.

Problems

Problem 14.1 (Optimizing the syndrome extraction circuit)**:** Show how to measure the X-checks and Z-checks of a rough surface code with the shortest possible depth circuit. Use the rough surface code as drawn in Figure 14.6. Your circuit should avoid hook errors and the bad circuit scheduling depicted in Figure 14.7, and ensure that each data qubit is involved in at most one CNOT during each round.

15 Transversal gates

So far, we have developed circuit-level fault-tolerant quantum memories that protect stored logical qubits. Fault-tolerant quantum computation must also manipulate the logical information. We cannot safely unencode, apply an ordinary physical gate, and re-encode: during the unencoded step, a single physical error can directly corrupt the logical qubit. Logical gates must therefore act directly on the encoded information.

15.1 Fault-tolerant preparation and measurement

Preparing a general encoded state is nontrivial. Simple computational-basis states are much easier to prepare in CSS codes.

15.1.1 Preparation of a logical codeword

Proposition 15.1 (Fault-tolerant preparation of a logical zero state)**:** Suppose that an LDPC CSS code admits fault-tolerant repeated syndrome extraction and decoding. Then the state $|0_L\rangle^{\otimes k}$ can be prepared fault tolerantly from a physical product state.

Proof. Initialize the n physical qubits in

$$|\psi_0\rangle = |0\rangle^{\otimes n}. \tag{15.1}$$

This state has eigenvalue $+1$ under every Z-type check and every chosen logical Z operator:

$$Z(h_z)\,|\psi_0\rangle = |\psi_0\rangle \qquad \text{for every row } h_z \text{ of } H_Z, \tag{15.2a}$$

$$\overline{Z}_a\,|\psi_0\rangle = |\psi_0\rangle \qquad \text{for } a = 1, \ldots, k. \tag{15.2b}$$

Repeatedly measure both the X- and Z-type checks. The first X-check outcomes project the product state into definite X-syndrome sectors. A decoder then chooses a likely Z-type recovery, which may either be applied physically or stored in a Pauli frame. The Z-check measurements diagnose physical X errors that occur during preparation. Once decoding succeeds, all stabilizer eigenvalues are $+1$, while the logical Z eigenvalues in Eq. (15.2b) remain fixed. The resulting encoded state is therefore $|0_L\rangle^{\otimes k}$. □

The first X syndrome is intrinsically random, so the corresponding Z-recovery or Pauli-frame representative can have large weight. This does not represent a macroscopic physical error produced by the preparation: it selects the $+1$ representatives of the newly measured X checks. A logical Z ambiguity in this first frame is harmless on $|0_L\rangle^{\otimes k}$. Repeated syndrome rounds are needed to suppress faults in the extraction circuit and to infer the frame reliably.

This is essentially the same fault-tolerant procedure used for quantum memory, now with a simple physical initial condition.

15.1.2 Destructive logical Z measurement

Proposition 15.2 (Fault-tolerant logical measurement)**:** Any CSS code has a fault-tolerant way to measure the Z-logical operators.

Proof. Measure every physical Z_i simultaneously and record the bit string

$$m \in \mathbb{F}_2^n, \qquad Z_i = (-1)^{m_i}. \tag{15.3}$$

Let $m_0 \in \ker(H_Z)$ denote the ideal readout string. A pre-existing physical X error e_x and a faulty Z_i readout pattern f_z enter in exactly the same way. The observed record is

$$\widetilde{e}_x := e_x + f_z, \tag{15.4a}$$

$$m = m_0 + \widetilde{e}_x, \tag{15.4b}$$

$$s := H_Z m = H_Z \widetilde{e}_x. \tag{15.4c}$$

Choose any reference error $r_x(s)$ satisfying

$$H_Z r_x(s) = s. \tag{15.5}$$

Errors with the measured syndrome are divided into logical equivalence classes of the form

$$r_x(s) + \mathrm{im}(H_X^{\mathsf{T}}) + \ell_x, \qquad \ell_x \in \mathrm{im}(G_X^{\mathsf{T}}). \tag{15.6}$$

A MAP decoder (Definition 12.5) chooses the most likely logical class,

$$\widehat{\ell}_x := \underset{\ell_x \in \mathrm{im}(G_X^{\mathsf{T}})}{\arg\max}\ \mathbb{P}\Big[\widetilde{e}_x \in r_x(s) + \mathrm{im}(H_X^{\mathsf{T}}) + \ell_x \,\Big|\, s\Big]. \tag{15.7}$$

We correct the classical measurement record according to

$$m_{\mathrm{corr}} := m + r_x(s) + \widehat{\ell}_x. \tag{15.8}$$

When the decoder selects the correct logical class,

$$m_{\mathrm{corr}} = m_0 + c_x, \qquad c_x \in \mathrm{im}(H_X^{\mathsf{T}}), \qquad m_{\mathrm{corr}} \in \ker(H_Z). \tag{15.9}$$

For each encoded qubit, choose any binary representative $\ell_z^{(a)}$ of its logical Z operator and report

$$\eta_a := \ell_z^{(a)} \cdot m_{\mathrm{corr}} \in \mathbb{F}_2, \qquad a = 1, \ldots, k. \tag{15.10}$$

The inferred logical state is the computational-basis state $|\eta_1 \cdots \eta_k\rangle_L$. If another representative of

the same logical operator is chosen, then

$$\ell_z^{(a)\prime} = \ell_z^{(a)} + H_Z^{\mathsf{T}} b \quad \Longrightarrow \quad (\ell_z^{(a)\prime} - \ell_z^{(a)}) \cdot m_{\text{corr}} = b \cdot H_Z m_{\text{corr}} = 0. \tag{15.11}$$

Thus every representative of a given logical Z gives the same answer. Fault tolerance again follows from successful decoding: at sufficiently low physical and readout error rates, a wrong logical result requires the decoder to select the wrong logical X class. □

No repeated measurement of each individual Z_i is assumed here. A faulty final readout is absorbed into the effective error $\widetilde{e}_x$ and is handled by the same MAP-style logical-class decoder. If earlier syndrome rounds are available, they can be included as additional data in the posterior probability in Eq. (15.7).

15.2 Logical gates

We next ask how to apply gates directly to encoded information.

Definition 15.3 (Logical gate)**:** Let $\mathcal{C}$ be a codespace. A physical unitary U implements a **logical gate** if

$$U\mathcal{C} = \mathcal{C}. \tag{15.12}$$

Equivalently, if $P_{\mathcal{C}}$ denotes the codespace projector, then

$$U P_{\mathcal{C}} U^{\dagger} = P_{\mathcal{C}}. \tag{15.13}$$

The identity is a logical gate, so the physically interesting case is one in which the restriction of U to the codespace is not merely a scalar. One convenient condition is

$$P_{\mathcal{C}} U P_{\mathcal{C}} \not\propto P_{\mathcal{C}}. \tag{15.14}$$

Equivalently, there exist orthogonal states $|\psi_a\rangle, |\psi_b\rangle \in \mathcal{C}$ for which

$$\langle\psi_a|\psi_b\rangle = 0, \qquad \langle\psi_a|U|\psi_b\rangle \neq 0. \tag{15.15}$$

For a Pauli stabilizer code,

$$P_{\mathcal{C}} = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} S. \tag{15.16}$$

Hence every logical unitary obeys

$$U \left(\sum_{S \in \mathcal{S}} S \right) U^{\dagger} = \sum_{S \in \mathcal{S}} S. \tag{15.17}$$

This does not require U to permute the individual Pauli stabilizers. For a generic non-Clifford logical gate, $USU^\dagger$ can be a non-Pauli operator, although the conjugated operators still form a commuting group that fixes the same codespace.

Proposition 15.4 (Clifford logical gates normalize the stabilizer group)**:** Let U be both a physical Clifford gate and a logical gate for a Pauli stabilizer code with stabilizer group $\mathcal{S}$. Then

$$U\mathcal{S}U^\dagger = \mathcal{S}. \tag{15.18}$$

Proof. For $S \in \mathcal{S}$ and $|\psi\rangle \in \mathcal{C}$, the state $U^\dagger |\psi\rangle$ also lies in $\mathcal{C}$, and therefore

$$USU^\dagger |\psi\rangle = |\psi\rangle . \tag{15.19}$$

Because U is Clifford, $USU^\dagger$ is a Pauli string. Every Pauli string that fixes the full codespace belongs to $\mathcal{S}$, so $U\mathcal{S}U^\dagger \subseteq \mathcal{S}$. Applying the same argument to $U^\dagger$ gives equality. □

Generic non-Clifford logical gates need not leave the Pauli stabilizer group invariant. We now look for logical gates with especially benign error propagation.

15.3 Transversal gates and fault tolerance

Definition 15.5 (Transversal gate)**:** Partition the physical qubits into disjoint subsets

$$\{1, \dots, n\} = R_1 \sqcup \cdots \sqcup R_r, \qquad |R_j| \le q = O(1). \tag{15.20}$$

A gate is **transversal** with respect to this partition if

$$U = \prod_{j=1}^{r} U_{R_j}, \tag{15.21}$$

where U_{R_j} acts only on the qubits in R_j.

Some authors reserve the term **strictly transversal** for the smallest possible cells. For several code blocks, the standard strictly transversal condition is that each cell R_j contains at most one physical qubit from each block; for a single block, each factor is then a one-qubit gate.

Proposition 15.6 (Fault tolerance of transversal gates)**:** Consider an LDPC CSS code family with distance $d \gtrsim \log n$ and a decoder that yields fault-tolerant memory below threshold. Under sufficiently weak local stochastic noise, a bounded-size transversal gate followed by error correction is fault tolerant.

Proof sketch. The argument is analogous to the cluster argument for quantum memory. An error

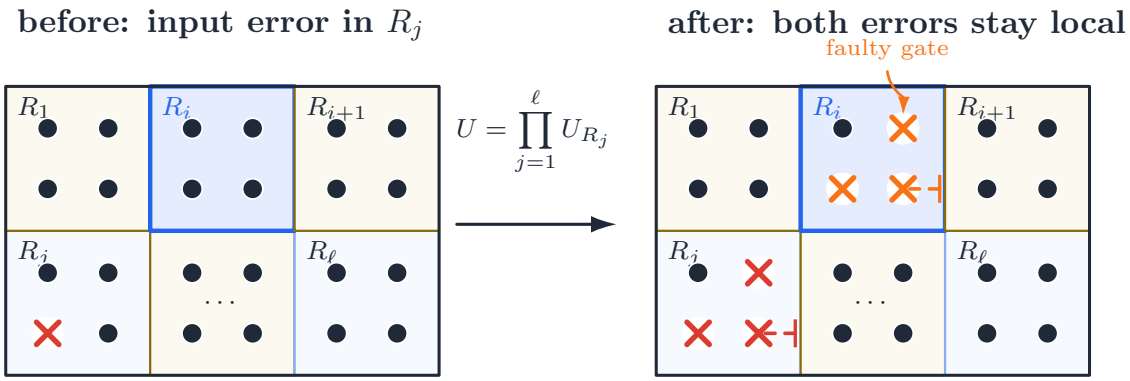

Figure 15.1: Errors cannot propagate very far in a transversal gate, making them manifestly fault-tolerant.

can spread only within the bounded cell containing its original support. In particular,

$$\operatorname{supp}(UEU^\dagger) \subseteq \bigcup_{j:\, R_j \cap \operatorname{supp}(E) \neq \varnothing} R_j, \qquad \operatorname{wt}(UEU^\dagger) \leq q \operatorname{wt}(E). \tag{15.22}$$

Likewise, an independent gate fault in one cell cannot propagate outside that cell. Hence small, disconnected fault clusters remain bounded-size clusters, and the same below-threshold decoder used for memory removes them in later rounds of error correction: see Figure 15.1. □

15.4 Transversal CNOT for every CSS code

Let h_X and h_Z be the check matrices of one CSS code, and let g_X and g_Z contain chosen logical Pauli representatives. For two identical code blocks,

$$H_X = \begin{pmatrix} h_X & 0 \\ 0 & h_X \end{pmatrix}, \qquad H_Z = \begin{pmatrix} h_Z & 0 \\ 0 & h_Z \end{pmatrix}, \tag{15.23a}$$

$$G_X = \begin{pmatrix} g_X & 0 \\ 0 & g_X \end{pmatrix}, \qquad G_Z = \begin{pmatrix} g_Z & 0 \\ 0 & g_Z \end{pmatrix}. \tag{15.23b}$$

Theorem 15.7 (Transversal CNOT for CSS codes)**:** For two copies of any CSS code, the bitwise gate

$$\overline{\mathrm{CNOT}} := \prod_{i=1}^{n} \mathrm{CNOT}_{i^{(1)} \to i^{(2)}} \tag{15.24}$$

implements a logical CNOT from the first code block to the second.

Proof. Order binary Pauli coordinates as $(x^{(1)}, x^{(2)} \mid z^{(1)}, z^{(2)})$. The symplectic action of the bitwise

CNOT is (Example 13.7)

$$M_{\mathrm{CNOT}} = \begin{pmatrix} \mathbb{I} & 0 & 0 & 0 \\ \mathbb{I} & \mathbb{I} & 0 & 0 \\ 0 & 0 & \mathbb{I} & \mathbb{I} \\ 0 & 0 & 0 & \mathbb{I} \end{pmatrix}. \tag{15.25}$$

The two-block stabilizer matrix transforms as

$$M_{\mathrm{CNOT}} \left(\begin{array}{cc|cc} h_X^{\mathsf{T}} & 0 & 0 & 0 \\ 0 & h_X^{\mathsf{T}} & 0 & 0 \\ \hline 0 & 0 & h_Z^{\mathsf{T}} & 0 \\ 0 & 0 & 0 & h_Z^{\mathsf{T}} \end{array}\right) = \left(\begin{array}{cc|cc} h_X^{\mathsf{T}} & 0 & 0 & 0 \\ h_X^{\mathsf{T}} & h_X^{\mathsf{T}} & 0 & 0 \\ \hline 0 & 0 & h_Z^{\mathsf{T}} & h_Z^{\mathsf{T}} \\ 0 & 0 & 0 & h_Z^{\mathsf{T}} \end{array}\right). \tag{15.26}$$

Equivalently, the new CSS check matrices are

$$H'_X = \begin{pmatrix} h_X & h_X \\ 0 & h_X \end{pmatrix}, \qquad H'_Z = \begin{pmatrix} h_Z & 0 \\ h_Z & h_Z \end{pmatrix}. \tag{15.27}$$

Adding the second block row to the first in H'_X, and the first block row to the second in H'_Z, returns the original matrices in Eq. (15.23a). These row operations only change the stabilizer generating set, so the codespace is invariant.

The same calculation for the logical representatives gives

$$G'_X = \begin{pmatrix} g_X & g_X \\ 0 & g_X \end{pmatrix}, \qquad G'_Z = \begin{pmatrix} g_Z & 0 \\ g_Z & g_Z \end{pmatrix}. \tag{15.28}$$

Thus, for each logical qubit a,

$$X_{L,a}^{(1)} \longmapsto X_{L,a}^{(1)} X_{L,a}^{(2)}, \tag{15.29a}$$
$$X_{L,a}^{(2)} \longmapsto X_{L,a}^{(2)}, \tag{15.29b}$$
$$Z_{L,a}^{(1)} \longmapsto Z_{L,a}^{(1)}, \tag{15.29c}$$
$$Z_{L,a}^{(2)} \longmapsto Z_{L,a}^{(1)} Z_{L,a}^{(2)}. \tag{15.29d}$$

This is precisely the logical CNOT action. If each block encodes $k > 1$ logical qubits, the physical gate implements CNOT simultaneously on the k corresponding logical pairs. □

The preceding proof is an encoded version of Gottesman–Knill stabilizer tracking: invariance of the stabilizers establishes that a logical gate was implemented, while the transformation of the logical Pauli operators identifies that gate.

15.5 Getting a full transversal Clifford gate set

There is a useful sufficient condition for obtaining the rest of the logical Clifford group transversally. We first state it for a code encoding one logical qubit.

Theorem 15.8 (Transversal Clifford gates for a self-dual CSS code)**:** Let a CSS code have parameters $[[n, 1, d]]$ and satisfy

$$H_X = H_Z = H_0. \tag{15.30}$$

Suppose that every row h of H_0 has weight

$$\mathrm{wt}(h) = 0 \pmod 4. \tag{15.31}$$

Then the code has the following transversal logical gates:

1. $U_H := H^{\otimes n}$ implements logical Hadamard;
2. either $U_S := S^{\otimes n}$ or $U_S^\dagger$ implements logical S;
3. bitwise CNOT between two code blocks implements logical CNOT.

Consequently, these gates generate the full logical Clifford group.

Proof. First consider U_H. On a weight-four check, for example,

$$U_H X_1 X_2 X_3 X_4 U_H^\dagger = Z_1 Z_2 Z_3 Z_4. \tag{15.32}$$

More generally, Hadamard exchanges every X-type and Z-type check. In the binary representation,

$$M_H \begin{pmatrix} H_0^\mathsf{T} & 0 \\ 0 & H_0^\mathsf{T} \end{pmatrix} = \begin{pmatrix} 0 & H_0^\mathsf{T} \\ H_0^\mathsf{T} & 0 \end{pmatrix}. \tag{15.33}$$

The columns, and hence the stabilizer generators, are merely permuted.

Because the code encodes one logical qubit, we may choose the same binary representative g_0 for logical X and logical Z:

$$X_L = X(g_0), \qquad Z_L = Z(g_0). \tag{15.34}$$

The logical matrix transforms in the same way,

$$M_H \begin{pmatrix} g_0^\mathsf{T} & 0 \\ 0 & g_0^\mathsf{T} \end{pmatrix} = \begin{pmatrix} 0 & g_0^\mathsf{T} \\ g_0^\mathsf{T} & 0 \end{pmatrix}, \tag{15.35}$$

so that

$$U_H X_L U_H^\dagger = Z_L, \tag{15.36a}$$
$$U_H Z_L U_H^\dagger = X_L. \tag{15.36b}$$

Thus U_H implements logical Hadamard.

Next consider U_S. Its binary action sends an X component to an equal X and Z component:

$$M_S \begin{pmatrix} H_0^\mathsf{T} & 0 \\ 0 & H_0^\mathsf{T} \end{pmatrix} = \begin{pmatrix} H_0^\mathsf{T} & 0 \\ H_0^\mathsf{T} & H_0^\mathsf{T} \end{pmatrix}. \tag{15.37}$$

This binary calculation does not by itself determine the stabilizer signs. If h is a row of H_0, then

$$U_S X(h) U_S^\dagger = \mathrm{i}^{\mathrm{wt}(h)} X(h) Z(h) = X(h) Z(h), \tag{15.38}$$

due to (15.31). The $Z(h)$ factor is itself a stabilizer, and the phase is $+1$, so all stabilizer eigenvalues remain unchanged.

For the logical X operator,

$$U_S X_L U_S^\dagger = \mathrm{i}^{\mathrm{wt}(g_0)} X_L Z_L = \begin{cases} Y_L, & \mathrm{wt}(g_0) = 1 \pmod 4, \\ -Y_L, & \mathrm{wt}(g_0) = 3 \pmod 4. \end{cases} \tag{15.39}$$

The weight of g_0 must be odd, since

$$g_0 \cdot g_0 = \mathrm{wt}(g_0) \pmod 2 = 1 \tag{15.40}$$

is the anticommutation condition between X_L and Z_L. Therefore U_S implements either logical S or logical $S^\dagger$; taking the inverse gives the other convention.

The transversal CNOT was proved in Theorem 15.7. □

If $k > 1$, the same hypotheses ensure that $H^{\otimes n}$ and $S^{\otimes n}$ are logical Clifford gates, but their action need not be an elementary Hadamard or phase gate on each chosen logical qubit. They can permute, mix, or entangle the logical qubits.

15.6 Two-dimensional color codes and the Steane code

Definition 15.9 (Two-dimensional color code)**:** Take a trivalent, face-three-colorable planar cellulation and place one qubit on every vertex. For every face f, define the two checks

$$X_f := \prod_{v \in f} X_v, \qquad Z_f := \prod_{v \in f} Z_v. \tag{15.41}$$

An example of such a color code is shown in Figure 15.2.

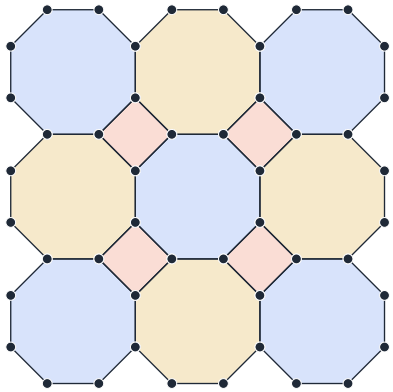

Figure 15.2: A patch of a 2d color code.

Proposition 15.10: The face operators in Eq. (15.41) define a CSS stabilizer code.

Proof. Two distinct faces in the cellulation share either no vertices or the two vertices of one common edge. Hence their supports overlap evenly. A face of one color is surrounded by faces whose colors alternate between the other two colors, so every face has an even number of edges and vertices. Thus the X_f and Z_f checks on the same face also overlap evenly. Consequently,

$$H_X = H_Z =: H_{\text{face}}, \qquad H_X H_Z^{\mathsf{T}} = 0, \tag{15.42}$$

and all X- and Z-type checks commute. □

Proposition 15.11 (Square–octagon color code)**:** A $k = 1$ square–octagon color-code patch has a transversal generating set for its logical Clifford group.

Proof. The X- and Z-check matrices coincide by Eq. (15.42). Every face has either four or eight vertices, so every check generator has weight divisible by four. The claim follows from Theorem 15.8. □

Color codes are two-dimensional topological codes, much like surface codes, and are subject to the same general tradeoff bounds. Their self-dual check geometry gives them more transversal gates and several other useful properties. We will return to color codes in Section **??**.

Example 15.12 (Steane code)**:** The smallest nontrivial triangular color code is the $[[7, 1, 3]]$ Steane code, as depicted in Figure 15.3. A convenient choice of check matrices is

$$H_X = H_Z = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{15.43}$$

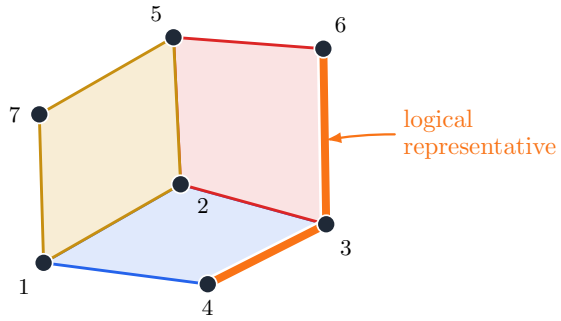

Figure 15.3: A depiction of the Steane code as a simple 2d color code.

The same binary vector can be used for logical X and logical Z:

$$G_X = G_Z = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}. \tag{15.44}$$

This representative has weight three. Therefore

$$H^{\otimes 7} \longmapsto \overline{H}, \tag{15.45a}$$

$$S^{\otimes 7} \longmapsto \overline{S}^\dagger, \tag{15.45b}$$

$$\prod_{i=1}^{7} \mathrm{CNOT}_{i^{(1)} \to i^{(2)}} \longmapsto \overline{\mathrm{CNOT}}. \tag{15.45c}$$

These gates give a transversal generating set for the full logical Clifford group.

Proposition 15.13: The Steane code is the smallest CSS code with distance $d = 3$.

Problems

Problem 15.1 (Logical gates via physical SWAP gates)**:** We may implement fault-tolerant logical gates by physically swapping qubits, whenever this operation has a very low error rate in hardware: usually in trapped ion or neutral atom hardware, where we simply physically move the qubits, rather than implementing SWAP directly. The SWAP gate simply exchanges two qubits, exactly what it sounds like:

$$\mathrm{SWAP}_{1,2} \begin{pmatrix} X_1 \\ X_2 \\ Z_1 \\ Z_2 \end{pmatrix} \mathrm{SWAP}_{1,2}^\dagger = \begin{pmatrix} X_2 \\ X_1 \\ Z_2 \\ Z_1 \end{pmatrix}. \tag{15.46}$$

A: Explain why a necessary and "sufficient" condition for a CSS code to have a logical gate

implemented by qubit permutation is the following. Let Σ be the $n \times n$ matrix that describes the permutation of qubits: namely, $\Sigma_{ij} = 1$ if and only if j is sent to i under the permutation. Then we require that there exist invertible $\mathbb{F}_2$-valued matrices M_X and M_Z such that

$$H_X = M_X H_X \Sigma, \tag{15.47a}$$

$$H_Z = M_Z H_Z \Sigma. \tag{15.47b}$$

B: It remains to find an example of a code that obeys (15.47), where the resulting transformation on on logical operators does not act as the identity. Here is one such example:

$$H_X = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{15.48a}$$

$$H_Z = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{15.48b}$$

Verify that this is a [[10,2,3]] code, and find (one choice of) G_X and G_Z.

C: Consider the permutation that swaps qubit $2k-1$ with qubit $2k$ for $k = 1, 2, 3, 4, 5$. Show that this obeys (15.47). Further show that this physical SWAP implements a non-trivial logical Clifford gate, and find what it is in the presentation of G_X and G_Z you found in **B**.

16 Universal gate sets

We have seen how to implement logical Clifford gates fault tolerantly and, for suitable codes, transversally. However, Clifford circuits with stabilizer inputs and Pauli measurements are also efficiently simulable on a classical computer. Universal quantum computation therefore requires access to an additional, non-Clifford resource.

16.1 The Clifford hierarchy

Definition 16.1 (Clifford hierarchy)**:** Let $\mathcal{P}_n$ be the n-qubit Pauli group, with all groups below understood projectively, so that overall phases are ignored. Define

$$\mathcal{C}_n^{(1)} := \mathcal{P}_n, \tag{16.1a}$$

$$\mathcal{C}_n^{(2)} := \mathcal{C}_n, \tag{16.1b}$$

and, recursively for $k \geq 2$,

$$\mathcal{C}_n^{(k)} := \left\{ U \in \mathrm{U}(2^n) : U\mathcal{P}_n U^\dagger \subseteq \mathcal{C}_n^{(k-1)} \right\}. \tag{16.2}$$

The second level is the Clifford group because Clifford gates conjugate every Pauli operator to another Pauli operator. In contrast, $\mathcal{C}_n^{(k)}$ is generally not a group for $k > 2$. This last point is essential. Although the elementary non-Clifford generators used below lie in $\mathcal{C}^{(3)}$, products of such gates and Clifford gates need not remain in $\mathcal{C}^{(3)}$. There is therefore no contradiction between the finiteness of each fixed hierarchy level, modulo phase, and the density of the group generated by Clifford and T gates.

Example 16.2 (T gate)**:** The standard single-qubit gate in the third level is

$$T := \begin{pmatrix} 1 & 0 \\ 0 & \mathrm{e}^{\mathrm{i}\pi/4} \end{pmatrix} = \mathrm{e}^{\mathrm{i}\pi/8}\mathrm{e}^{-\mathrm{i}\pi Z/8}, \qquad T^2 = S. \tag{16.3}$$

Its conjugation action on the Pauli generators is

$$TXT^\dagger = \mathrm{e}^{-\mathrm{i}\pi/4} SX \in \mathcal{C}_1^{(2)}, \tag{16.4a}$$

$$TZT^\dagger = Z. \tag{16.4b}$$

Thus $T \in \mathcal{C}_1^{(3)}$.

Example 16.3 (CCNOT gate)**:** A three-qubit example in the third level is the Toffoli, or CCNOT, gate. With qubits 1 and 2 as controls and qubit 3 as target,

$$\mathrm{CCNOT}_{12\to 3} = \mathbb{I} - 2\left(\frac{\mathbb{I} - Z_1}{2}\right)\left(\frac{\mathbb{I} - Z_2}{2}\right)\left(\frac{\mathbb{I} - X_3}{2}\right). \tag{16.5}$$

It lies in $\mathcal{C}_3^{(3)}$, because for example

$$\mathrm{CCNOT}_{12\to 3}\, X_1\, \mathrm{CCNOT}_{12\to 3}^{\dagger} = \mathrm{CNOT}_{2\to 3} \in \mathcal{C}_3^{(2)}. \tag{16.6}$$

The T gate, Toffoli, and closely related variants are the most common non-Clifford gates that one tries to implement fault tolerantly.

16.2 Universality and the Solovay–Kitaev theorem

A useful structural fact is that the Clifford group is already very large.

Theorem 16.4: Let $G \subseteq \mathrm{U}(2^n)$ be generated projectively by a discrete gate set G_0. If

$$\mathcal{C}_n \subset G, \tag{16.7}$$

i.e. Cliffords are a strict subset of G, then G is dense projectively in $\mathrm{U}(2^n)$. Equivalently, for every $U \in \mathrm{U}(2^n)$ and every $\epsilon > 0$, there are gates $g_1, \ldots, g_{\ell(\epsilon)} \in G_0$ such that

$$\left\| U - g_1 g_2 \cdots g_{\ell(\epsilon)} \right\| \leq \epsilon. \tag{16.8}$$

We will not prove this theorem. Its practical consequence is that a fault-tolerant implementation of one suitable non-Clifford gate is enough to complete the Clifford gates to a universal gate set.

Corollary 16.5: The gate set $\{H, T, \mathrm{CNOT}\}$ is universal for quantum computation.

The fact that $\mathcal{C}_n^{(3)}$ is not a group is crucial here: the group generated by Clifford and T gates is not confined to the third level of the hierarchy.

Density alone does not yet control the length of the approximating circuit. The Solovay–Kitaev theorem gives a polylogarithmic bound.

Theorem 16.6 (Solovay–Kitaev)**:** Let $G_0 \subset \mathrm{SU}(2)$ be a finite, inverse-closed gate set that generates a dense subgroup. There is a constant $c < 4$ such that every one-qubit unitary can be approximated to operator-norm error ϵ using a word of length

$$\ell(\epsilon) \lesssim \left(\log \frac{1}{\epsilon} \right)^{c}. \tag{16.9}$$

Proof sketch. Assume first that a finite lookup table gives a coarse approximation with small but fixed error. If a residual unitary Δ is close to the identity, a balanced-commutator decomposition writes

$$\Delta = U_A U_B U_A^{-1} U_B^{-1}, \qquad \|U_A - \mathbb{I}\|, \|U_B - \mathbb{I}\| = \mathrm{O}\left(\sqrt{\|\Delta - \mathbb{I}\|} \right). \tag{16.10}$$

Suppose the current residual error is ϵ_k. Recursively approximate U_A and U_B at the preceding scale and replace the residual by the commutator of the approximating words. The first-order errors

cancel inside the commutator, giving the schematic recursions

$$\epsilon_{k+1} = \mathrm{O}\big(\epsilon_k^{3/2}\big)\,, \tag{16.11a}$$

$$\ell_{k+1} \le 5\ell_k + \mathrm{O}(1). \tag{16.11b}$$

Taking the initial residual to be ϵ_0, one reaches errors of order

$$\epsilon_k \sim \epsilon_0^{(3/2)^k} \tag{16.12}$$

using a word of length of order $\ell_0 5^k$. Eliminating k gives

$$c \le \frac{\log 5}{\log(3/2)} \approx 3.97 \tag{16.13}$$

in (16.9). □

16.3 A transversal non-Clifford gate

Amazingly, there are quantum codes with a transversal non-Clifford gate.

Proposition 16.7 (Transversal T in the quantum Reed–Muller code)**:** The $[[15, 1, 3]]$ quantum Reed–Muller code is defined by

$$H_X = \begin{pmatrix} 1&1&1&1&1&1&1&1&0&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0&1&1&1&1&0&0&0 \\ 1&1&0&0&1&1&0&0&1&1&0&0&1&1&0 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \end{pmatrix}. \tag{16.14}$$

The Z-check matrix is

$$H_Z = \begin{pmatrix} 1&1&1&1&1&1&1&1&0&0&0&0&0&0&0 \\ 1&1&1&1&0&0&0&0&1&1&1&1&0&0&0 \\ 1&1&0&0&1&1&0&0&1&1&0&0&1&1&0 \\ 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 1&1&1&1&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 1&0&1&0&1&0&1&0&0&0&0&0&0&0&0 \\ 1&1&0&0&0&0&0&0&1&1&0&0&0&0&0 \\ 1&0&1&0&0&0&0&0&1&0&1&0&0&0&0 \\ 1&0&0&0&1&0&0&0&1&0&0&0&1&0&0 \end{pmatrix}. \tag{16.15}$$

A nonminimal choice of logical Pauli representatives is

$$G_X = G_Z = \begin{pmatrix} 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \end{pmatrix}. \tag{16.16}$$

This code admits a transversal logical $T^\dagger$ gate.

Proof. The four rows of H_X are independent, so $\mathrm{im}(H_X^\mathsf{T})$ contains 16 binary vectors. Every nonzero vector in this space has Hamming weight 8. The logical zero state can therefore be written as

$$|0_L\rangle = \frac{1}{4} \sum_{v \in \mathrm{im}(H_X^\mathsf{T})} |v\rangle \,. \tag{16.17}$$

Since $T^{\otimes 15}$ multiplies a computational-basis string of weight w by $\mathrm{e}^{\mathrm{i}\pi w/4}$, every string appearing in Eq. (16.17) has phase one. Hence

$$T^{\otimes 15} |0_L\rangle = |0_L\rangle \,. \tag{16.18}$$

The logical one state is

$$|1_L\rangle = X^{\otimes 15} |0_L\rangle \,. \tag{16.19}$$

Complementing a weight-0 or weight-8 string produces a weight-15 or weight-7 string. Both acquire the same phase,

$$\mathrm{e}^{15\mathrm{i}\pi/4} = \mathrm{e}^{7\mathrm{i}\pi/4} = \mathrm{e}^{-\mathrm{i}\pi/4}, \tag{16.20}$$

so

$$T^{\otimes 15} |1_L\rangle = \mathrm{e}^{-\mathrm{i}\pi/4} |1_L\rangle \,. \tag{16.21}$$

Therefore the physical transversal gate acts on the codespace as

$$P_{\mathcal{C}} T^{\otimes 15} P_{\mathcal{C}} = \overline{T}^\dagger P_{\mathcal{C}}. \tag{16.22}$$

Equivalently, $(T^\dagger)^{\otimes 15}$ implements logical T. □

The quantum Reed–Muller code does not have a transversal Hadamard gate. It does have a transversal phase gate, since squaring Eq. (16.22) gives

$$P_{\mathcal{C}} S^{\otimes 15} P_{\mathcal{C}} = \overline{S}^\dagger P_{\mathcal{C}}. \tag{16.23}$$

Thus the code supplies a useful non-Clifford gate but not a universal transversal gate set.

16.4 The Eastin–Knill theorem

Can one find a code that combines the transversal Clifford gates of the Steane code with the transversal T gate of the quantum Reed–Muller code? The Eastin–Knill theorem rules this out if all gates are transversal relative to one fixed correctable partition.

Theorem 16.8 (Eastin–Knill)**:** Let the physical qubits be partitioned into disjoint regions

$$\{1,\dots,n\} = R_1 \sqcup R_2 \sqcup \cdots \sqcup R_\ell, \tag{16.24}$$

and suppose that every region R_j is correctable. In particular, for a code of distance d, it is enough that

$$\max_j |R_j| < d. \tag{16.25}$$

Then the logical gates implemented by product unitaries

$$U = \bigotimes_{j=1}^{\ell} U_{R_j} \tag{16.26}$$

do not form a universal logical gate set.

Proof sketch. Suppose, toward a contradiction, that the product logical gates were universal. Taking the closure of this gate set inside the compact product group gives a compact Lie group. Its identity component would contain a one-parameter family of transversal logical gates,

$$U(\theta) = \bigotimes_{j=1}^{\ell} U_{R_j}(\theta), \qquad U(0) = \mathbb{I}. \tag{16.27}$$

By taking a unitary where θ is arbitrarily close to 0, we can look for a logical unitary that is arbitrarily close to the identity, which is generated by a sum of local terms:

$$U(\theta) = \mathbb{I} + \mathrm{i}\theta \sum_{j=1}^{\ell} A_j + \cdots . \tag{16.28}$$

Correctability of R_j and the Knill–Laflamme condition imply

$$P_{\mathcal{C}} A_j P_{\mathcal{C}} = a_j P_{\mathcal{C}} \tag{16.29}$$

for some scalar a_j. Therefore

$$P_{\mathcal{C}} \left(\sum_{j=1}^{\ell} A_j \right) P_{\mathcal{C}} = \left(\sum_{j=1}^{\ell} a_j \right) P_{\mathcal{C}}, \tag{16.30}$$

so every infinitesimal transversal logical gate acts only by an overall phase. The identity component is consequently trivial projectively. A compact Lie group has only finitely many connected components, so the projective logical image of the transversal group is finite and cannot be dense in the full logical unitary group. □

The theorem identifies a fundamental obstruction to universal computation using only a fixed

set of transversal logical gates. The most common way around the obstruction is to use measured ancillas and magic-state distillation, which will be the subject of the next lecture.

Eastin–Knill assumes one fixed partition. It does not rule out a collection of gates that are individually transversal relative to different correctable partitions, provided error correction is performed before switching from one partition to another: see Problem 17.2.

16.5 The Bravyi–König theorem

The Reed–Muller and Steane codes are small block codes. We now ask what restrictions arise for a family of Pauli stabilizer codes whose checks and logical circuits are geometrically local.

Theorem 16.9 (Bravyi–König)**:** Consider a family of geometrically local Pauli stabilizer codes in D spatial dimensions, with increasing code distance. A bounded-depth geometrically local circuit that preserves the codespace implements a logical gate in the Dth level of the Clifford hierarchy:

$$\overline{U} \in \mathcal{C}^{(D)}. \tag{16.31}$$

More quantitatively, the light-cone radius of the circuit and the diameter of the stabilizer generators must be small compared with the length scale on which the code distance becomes visible. The theorem therefore applies not only to strictly transversal gates, but also to any constant-depth circuit of bounded-range physical gates.

In two dimensions, every such protected gate is Clifford. A protected non-Clifford gate first becomes possible in three dimensions, where the third level includes gates such as T and CCNOT.

16.5.1 Proof sketch in two dimensions

The geometric construction is closely related to the proof of the Bravyi–Poulin–Terhal bound (Theorem 11.7). Partition the two-dimensional code into three regions A, B, and C, each formed from well-separated correctable pieces, as shown in Figure 16.1. The cleaning lemma allows arbitrary logical Pauli operators P_L and Q_L to be represented so that

$$\operatorname{supp}(P_L) \subseteq A \cup C, \tag{16.32a}$$

$$\operatorname{supp}(Q_L) \subseteq B \cup C. \tag{16.32b}$$

Let

$$Q_L := U X_L U^\dagger. \tag{16.33}$$

A bounded-depth local circuit can enlarge the support of Q_L (compared to X_L) by a bounded light-cone radius. Consider the group commutator

$$K_L := Z_L Q_L Z_L^\dagger Q_L^\dagger. \tag{16.34}$$

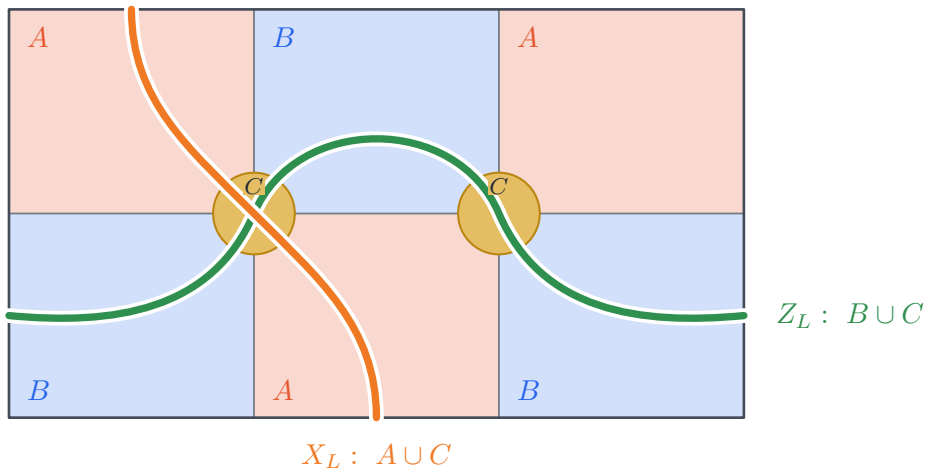

Figure 16.1: Partition of a 2d local region into A, B and C, with separate X_L and Z_L logicals cleaned out of B and A respectively.

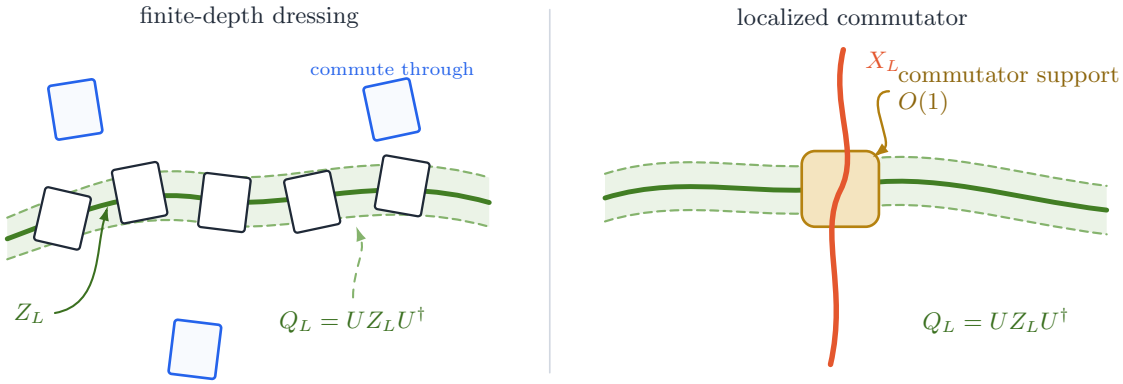

Figure 16.2: *Left:* under evolution by a local circuit, the local X_L cannot grow too much. *Right:* the commutator K_L must be localized near the intersection of Q_L and X_L.

Because the A- and B-regions are separated, the support of K_L is confined to a bounded thickening of C: see Figure 16.2. This is a baby circuit version of what are known as "Lieb-Robinson bounds". The union lemma makes this thickened region correctable, so the Knill–Laflamme condition gives

$$P_{\mathcal{C}} K_L P_{\mathcal{C}} = \lambda(P_L, Q_L) P_{\mathcal{C}}. \tag{16.35}$$

Since $P_L^2 = \mathbb{I}$, the phase is $\lambda(P_L, Q_L) = \pm 1$. Thus the logical operator $\overline{U}\,\overline{Q}_L\,\overline{U}^\dagger$ commutes or anticommutes with every logical Pauli $\overline{P}_L$. It must therefore be a logical Pauli itself. Since this holds for every X_L,

$$\overline{U} \in \mathcal{C}^{(2)}, \tag{16.36}$$

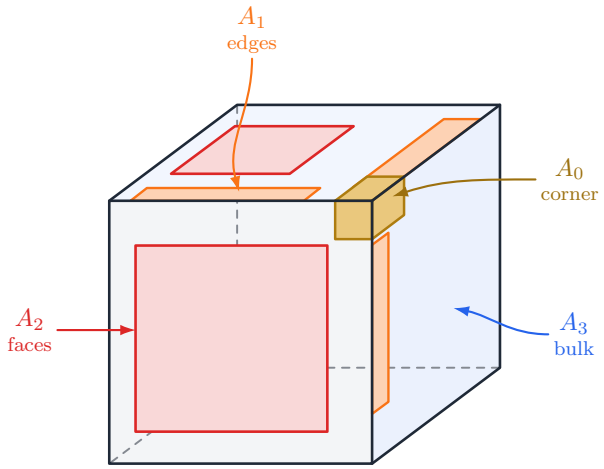

Figure 16.3: Partitioning a 3d region into A_0, A_1, A_2, A_3.

so U acts as a logical Clifford gate.

16.5.2 The hierarchy argument in three dimensions

In three dimensions, partition the code into four regions A_0, A_1, A_2, A_3 as shown in Figure 16.3, again built from separated correctable pieces. Choose logical Paulis Q_0, Q_1, Q_2 cleaned away from A_0, A_1, A_2, respectively, and define

$$R_0 := UQ_0U^\dagger, \tag{16.37a}$$
$$R_1 := Q_1R_0Q_1^\dagger R_0^\dagger, \tag{16.37b}$$
$$R_2 := Q_2R_1Q_2^\dagger R_1^\dagger. \tag{16.37c}$$

The same support-cancellation argument localizes R_2 inside the remaining correctable region A_3, and hence

$$P_{\mathcal{C}}R_2P_{\mathcal{C}} \propto P_{\mathcal{C}}. \tag{16.38}$$

Because this holds for every choice of logical Paulis Q_1 and Q_2, one climbs back through the recursive definition of the hierarchy:

$$R_2 \propto \mathbb{I} \implies R_1 \in \mathcal{C}^{(1)} \implies R_0 \in \mathcal{C}^{(2)} \implies U \in \mathcal{C}^{(3)}. \tag{16.39}$$

The argument generalizes naturally: in D dimensions, a partition into $D+1$ correctable regions and $D-1$ nested commutators after $UQ_0U^\dagger$ forces the logical gate into $\mathcal{C}^{(D)}$.

Two-dimensional code families are already attractive for many experimental architectures. The standard approach is therefore not to search for a universal family of geometrically local

transversal gates, but to supplement protected Clifford operations with state injection and magic-state distillation, as we discuss in Section 17 and Section **??**. However, we do note the following results which are a recent area of research interest:

Theorem 16.10: There exist local quantum codes in two dimensions, based on Kitaev's quantum double model with non-Abelian group D_4 (a generalization of the toric code in Section 9), with transversal logical CCNOT gates.

A detailed discussion of such codes is beyond the scope of this class.

Problems

Problem 16.1: We stated without proof that the Reed-Muller code has distance 3. Find d_X and d_Z explicitly and provide minimal weight representatives for the logicals.

17 Magic state distillation

Universal quantum computation requires a non-Clifford gate, or equivalently a nonstabilizer "magic" state. In this lecture we sketch how to obtain this resource fault tolerantly. We will focus less on optimizing the procedure and more on the conceptual picture.

17.1 Quantum teleportation

Algorithm 17.1 (Quantum teleportation)**:** Let qubit 1 contain an unknown state $|\psi\rangle$, and prepare qubits $2, 3$ in the Bell state

$$\left|\Phi^{+}\right\rangle_{23} := \frac{|00\rangle_{23} + |11\rangle_{23}}{\sqrt{2}}. \tag{17.1}$$

Measure the commuting Pauli operators X_1X_2 and Z_1Z_2, obtaining outcomes

$$\eta_{XX}, \eta_{ZZ} \in \{+1, -1\}. \tag{17.2}$$

A Pauli correction on qubit 3, determined by these two outcomes, leaves qubit 3 in the state $|\psi\rangle$. The corresponding circuit is depicted in Figure 17.1.

We can verify the algorithm using stabilizer and logical-operator tracking. Order the binary Pauli coordinates as $(x_1, x_2, x_3 \mid z_1, z_2, z_3)$. The columns of

$$\mathsf{S}_{\Phi^+} = \left(\begin{array}{cc} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ \hline 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{array}\right) \tag{17.3}$$

represent the Bell-pair stabilizers X_2X_3 and Z_2Z_3. After the Bell measurement, a generating matrix

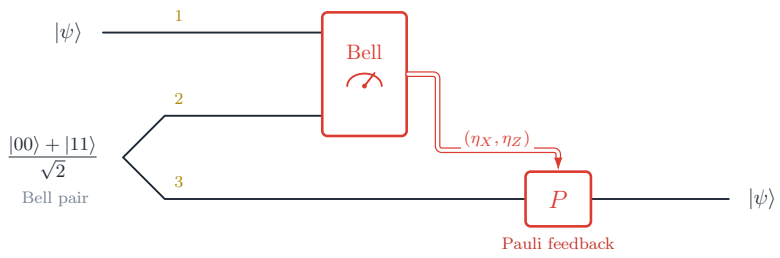

Figure 17.1: Circuit diagram for Algorithm 17.1.

for the measured stabilizers is

$$\mathsf{S}' = \left(\begin{array}{cc} 1 & 0 \\ 1 & 0 \\ \hline 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{array}\right), \tag{17.4}$$

with eigenvalues η_{XX} and η_{ZZ}, respectively.

Initially, logical X and Z for the unknown state are represented by

$$\mathsf{L} = \left(\begin{array}{cc} 1 & 0 \\ 0 & 0 \\ \hline 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{array}\right). \tag{17.5}$$

Modifying the two columns by the Bell stabilizers gives the stabilizer-equivalent representatives

$$\widetilde{\mathsf{L}} = \left(\begin{array}{cc} 1 & 0 \\ 1 & 0 \\ \hline 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{array}\right), \tag{17.6}$$

corresponding to $X_1X_2X_3$ and $Z_1Z_2Z_3$. In the post-measurement subspace, the measured operators X_1X_2 and Z_1Z_2 can be replaced by their eigenvalues. Therefore the logical data are represented on qubit 3 by

$$X_L = \eta_{XX} X_3, \tag{17.7a}$$

$$Z_L = \eta_{ZZ} Z_3. \tag{17.7b}$$

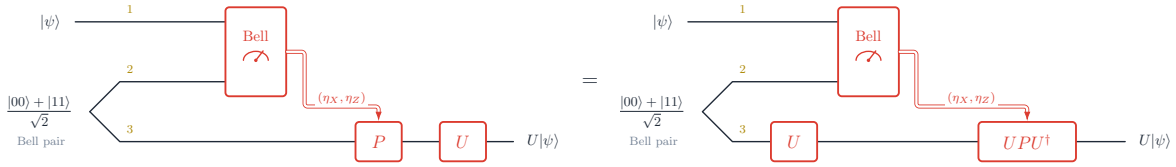

Figure 17.2: Circuit diagram for Algorithm 17.2.

The corresponding unsigned binary representatives are

$$\widetilde{\mathsf{L}}' = \left(\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ \hline 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{array}\right). \tag{17.8}$$

Thus qubit 3 must carry the original quantum data. The required Pauli feedback is, up to an irrelevant overall phase,

$$P(\eta_{XX}, \eta_{ZZ}) = X_3^{(1-\eta_{ZZ})/2} Z_3^{(1-\eta_{XX})/2} \sim \begin{cases} \mathbb{I}, & (\eta_{XX}, \eta_{ZZ}) = (+1, +1), \\ X_3, & (\eta_{XX}, \eta_{ZZ}) = (+1, -1), \\ Z_3, & (\eta_{XX}, \eta_{ZZ}) = (-1, +1), \\ Y_3, & (\eta_{XX}, \eta_{ZZ}) = (-1, -1). \end{cases} \tag{17.9}$$

17.2 Gate teleportation and magic states

Algorithm 17.2 (Gate teleportation)**:** As shown in Figure 17.2, we can modify Algorithm 17.1 to add the action of a gate U to the teleported qubit. If ordinary teleportation would have produced the Pauli byproduct P, then a Bell pair with U applied to its output half produces $UP|\psi\rangle$. Applying the conjugated correction gives

$$(UPU^\dagger)UP|\psi\rangle = U|\psi\rangle. \tag{17.10}$$

Equivalently, the resource state is the twisted Bell pair

$$|\Phi_U\rangle := (\mathbb{I} \otimes U)\left|\Phi^+\right\rangle. \tag{17.11}$$

For $U = T \in \mathcal{C}_1^{(3)}$, the correction obeys

$$TPT^\dagger \in \mathcal{C}_1^{(2)}, \tag{17.12}$$

so the correction is Clifford. The goal is therefore to teleport a gate on logical qubits encoded in a fault-tolerant code, using fault-tolerant Clifford operations.

The difficulty is that the original non-Clifford resource is noisy. Even when memory errors are exponentially suppressed by the code distance, the raw non-Clifford preparation is only physical-noise limited:

$$\mathbb{P}[\text{failure}] \sim \mathbb{P}[T \text{ preparation fails}] + \mathbb{P}[\text{memory fails}] \sim O(p_{\text{phys}}) + \mathrm{e}^{-\alpha d}. \tag{17.13}$$

The key question is how to obtain a very high-fidelity T resource.

Definition 17.3 (Magic state distillation)**:** The T magic state is

$$|T\rangle := T\,|+\rangle = \frac{|0\rangle + \mathrm{e}^{\mathrm{i}\pi/4}\,|1\rangle}{\sqrt{2}}. \tag{17.14}$$

Magic state distillation takes many noisy copies of $|T\rangle$ (often encoded in a first code) and probabilistically outputs one higher-fidelity copy (often encoded in an second code).

The reason why we often work directly with encoded magic states in the distillation procedure is so that we can do fault-tolerant error correction in the presence of noisy gates in the distillation circuits (which we'll describe shortly). The reason the distillation is probabilistic is because we will use a code to detect errors in the noisy input magic states and **postselect** on seeing no error:

Definition 17.4 (Postselection)**:** Postselection refers to a procedure where a quantum state is "rejected" if certain measurement outcomes are not obtained. We have to repeat the prior circuit and measure until we get the desired outcome.

17.3 Code concatenation

To implement the distillation procedure, we first formally define code concatenation.

Definition 17.5 (Concatenated CSS code)**:** Let $C_{\text{in}} = [[n_{\text{in}}, k_{\text{in}}, d_{\text{in}}]]$ and $C_{\text{out}} = [[n_{\text{out}}, k_{\text{out}}, d_{\text{out}}]]$ be CSS codes. Take k_{in} parallel copies of C_{out}. At each of the n_{out} outer coordinates, collect the corresponding k_{in} outer-code qubits and encode them as the k_{in} logical qubits of one block of C_{in}. The concatenated code has parameters $[[n_{\text{in}}n_{\text{out}}, k_{\text{in}}k_{\text{out}}, d]]$. More concretely, let $H_{X,\text{in}}$, $H_{Z,\text{in}}$ and $G_{X,\text{in}}$, $G_{Z,\text{in}}$ denote inner-code check and logical generator matrices, and use analogous notation for the outer code. With the physical qubits ordered block by block, the concatenated matrices are

$$H_X = \begin{pmatrix} \mathbb{I}_{n_{\text{out}}} \otimes H_{X,\text{in}} \\ H_{X,\text{out}} \otimes G_{X,\text{in}} \end{pmatrix}, \tag{17.15a}$$

$$H_Z = \begin{pmatrix} \mathbb{I}_{n_{\text{out}}} \otimes H_{Z,\text{in}} \\ H_{Z,\text{out}} \otimes G_{Z,\text{in}} \end{pmatrix}, \tag{17.15b}$$

$$G_X = G_{X,\text{out}} \otimes G_{X,\text{in}}, \tag{17.15c}$$

$$G_Z = G_{Z,\text{out}} \otimes G_{Z,\text{in}}. \tag{17.15d}$$

From the form of (17.15c) and (17.15d) we see that

$$d_X = d_{X,\text{in}} d_{X,\text{out}}, \qquad d_Z = d_{Z,\text{in}} d_{Z,\text{out}}. \tag{17.16}$$

Example 17.6 (Shor code): Take the inner bit-flip repetition code and outer phase-flip repetition code, with

$$H_{Z,\text{in}} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad H_{X,\text{in}} = 0, \qquad G_{X,\text{in}} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}, \quad G_{Z,\text{in}} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \tag{17.17a}$$

$$H_{Z,\text{out}} = 0, \qquad H_{X,\text{out}} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \quad G_{X,\text{out}} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \quad G_{Z,\text{out}} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}. \tag{17.17b}$$

The concatenated matrices $H_{X,Z}$ and $G_{X,Z}$ given by (17.15) reduce to (8.22), (8.32) and (8.35) as needed.

17.4 The 15-to-1 distillation protocol

For magic state distillation, we will choose the outer code to be the $[[15, 1, 3]]$ quantum Reed–Muller code (Section 16.3), which has a transversal T gate, and choose the inner code to be a surface code or another code with fault-tolerant Clifford operations: so schematically,

$$C_\text{out} = [[15, 1, 3]], \qquad C_\text{in} = [[L^2, 1, L]]. \tag{17.18}$$

Using the inverse convention established for the Reed–Muller code in the preceding lecture, the true logical T on the concatenated code is

$$T_{15L^2} = \left(T^\dagger_{L^2}\right)^{\otimes 15}, \tag{17.19}$$

where $T^\dagger_{L^2}$ denotes logical $T^\dagger$ on one inner-code block. The desired operation is therefore already transversal at the outer-code level.

The remaining problem is that the inner surface code does not have a transversal logical $T^\dagger$. We instead implement each of the fifteen logical $T^\dagger$ gates by gate teleportation, consuming a noisy magic state. The required twisted Bell state can be prepared using only a noisy $\left|T^\dagger\right\rangle := T^\dagger |+\rangle$ state and a Clifford CNOT:

$$(\mathbb{I} \otimes T^\dagger) \left|\Phi^+\right\rangle_{12} = \text{CNOT}_{2\to 1} \left(|0\rangle_1 \otimes \left|T^\dagger\right\rangle_2 \right). \tag{17.20}$$

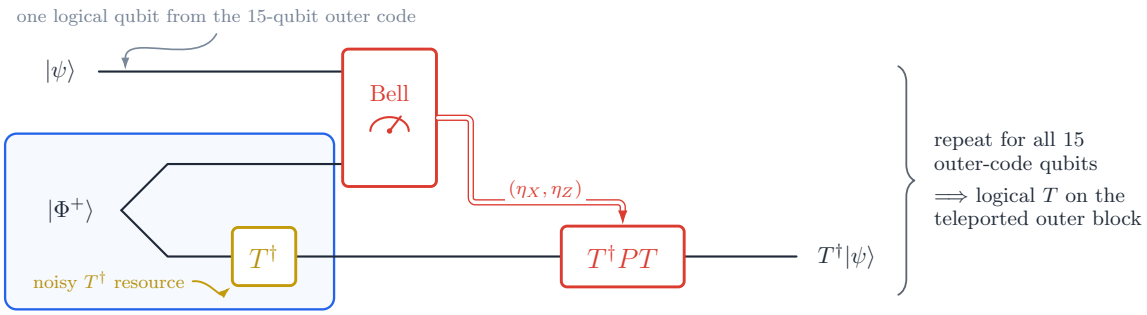

Figure 17.3: Repeating Figure 17.2, emphasizing how the gate teleportation is used to act with a logical $T^\dagger$ on the qubits representing one outer-code qubit of the Reed-Muller code. In practice, the part highlighted in blue is done via (17.20).

The inverse-convention resource is equivalent to the usual magic state under a Clifford gate:

$$\left|T^\dagger\right\rangle = S^\dagger \left|T\right\rangle . \tag{17.21}$$

Thus distilling $|T\rangle$ or $\left|T^\dagger\right\rangle$ has the same essential cost.

The Clifford couplings and Bell measurements in this injection gadget must be implemented fault tolerantly; lattice surgery or code deformation can be used for this purpose. We will discuss such planar implementations in Section 18. Repeating the gadget fifteen times applies the transversal outer-code operation in Eq. (17.19): see Figure 17.3.

Let $p_{\mathrm{circ},L}$ denote the failure probability of the protected Clifford circuitry and the outer-syndrome measurement. At distance L it is schematically

$$\mathbb{P}[\text{measurement of } H_{X,\mathrm{out}} \otimes G_{X,\mathrm{in}} \text{ fails}] \sim \mathrm{e}^{-\alpha L} =: p_{\mathrm{circ},L}. \tag{17.22}$$

The total accepted-output error has the form

$$\mathbb{P}[T_{15L^2} \left|+\right\rangle_{15L^2} \text{ is incorrect}] = O(p_{\mathrm{circ},L}) + p_T, \tag{17.23}$$

where p_T is the probability that errors in the raw magic states evade the outer Reed–Muller syndrome.

Proposition 17.7 (Leading error of 15-to-1 distillation)**:** Assume an independent stochastic noise model in which each of the fifteen raw magic states suffers a Z error with probability p_{phys}. Conditioned on accepting the trivial outer syndrome, and neglecting errors in the circuit operations, the output magic-state error is

$$p_T = 35 p_{\mathrm{phys}}^3 + O(p_{\mathrm{phys}}^4). \tag{17.24}$$

Proof. The Reed–Muller code has distance three, so one- and two-input errors are detected. To count the leading undetected errors, recall that the fifteen columns of $H_{X,\mathrm{out}}$ are the fifteen nonzero vectors in $\mathbb{F}_2^4$. A weight-three Z error supported on columns $a, b, c \in \mathbb{F}_2^4 \setminus \{0\}$ has trivial syndrome precisely when

$$H_{X,\mathrm{out}} e_Z = 0 \quad \Longleftrightarrow \quad a + b + c = 0 \quad \Longleftrightarrow \quad c = a + b. \tag{17.25}$$

Choose an ordered pair of distinct nonzero vectors a, b. There are $15 \cdot 14$ choices, and $c = a + b$ is then fixed and is distinct from both a and b. Each unordered triple is counted by all 3! permutations, so the number of undetected weight-three patterns is

$$N_3 = \frac{15 \cdot 14}{3!} = 35. \tag{17.26}$$

Each such weight-three operator is a logical error rather than a stabilizer. Indeed, the logical operator $\overline{X} = X^{\otimes 15}$ commutes with every Z-type stabilizer, so every Z stabilizer has even weight. An undetected weight-three Z operator has odd weight and anticommutes with $\overline{X}$; it therefore represents logical $\overline{Z}$. Conditioning on acceptance changes the result only by a relative factor $1 + O(p_{\mathrm{phys}})$ and does not alter the leading coefficient. The 35 weight-three logical patterns contribute $35 p_{\mathrm{phys}}^3 + O(p_{\mathrm{phys}}^4)$, proving Eq. (17.24). □

17.5 Teleporting the distilled state to one inner-code block

The distilled state initially occupies the full Reed–Muller/surface-code concatenated block. It is often useful to move it fault tolerantly into one ordinary L^2 inner-code block, so that subsequent stabilizer measurements again have the smaller inner-code weight.

Let $\overline{X}_{\mathrm{dst}}, \overline{Z}_{\mathrm{dst}}$ be the logical Paulis of a fresh destination block, and let $\overline{X}_{\mathrm{RM}}, \overline{Z}_{\mathrm{RM}}$ be the outer logical Paulis of the concatenated Reed–Muller block. Initialize the destination in $|+\rangle_{\mathrm{dst}}$ and start from $|+\rangle_{\mathrm{dst}} \otimes |T\rangle_{\mathrm{RM}}$. First fault tolerantly measure $\overline{Z}_{\mathrm{dst}} \overline{Z}_{\mathrm{RM}}$ with outcome $\eta_Z \in \{+1, -1\}$. A weight-three representative of $\overline{Z}_{\mathrm{RM}}$ involves only 3 of the 15 inner-code blocks. This measurement destroys the separate stabilizer $\overline{X}_{\mathrm{dst}}$, but preserves the commuting product $\overline{X}_{\mathrm{dst}} \overline{X}_{\mathrm{RM}}$. Next fault tolerantly measure $\overline{X}_{\mathrm{RM}}$, obtaining outcome η_X. The logical data are now represented entirely on the destination block as

$$X_L = \eta_X \overline{X}_{\mathrm{dst}}, \tag{17.27a}$$

$$Z_L = \eta_Z \overline{Z}_{\mathrm{dst}}. \tag{17.27b}$$

After applying or tracking the corresponding Pauli frame, the final state is schematically

$$|T\rangle_{\mathrm{dst}} \otimes |\mathrm{junk}\rangle_{\mathrm{RM}}\,. \tag{17.28}$$

This is the same state-teleportation procedure as in Algorithm 17.1, now performed on encoded

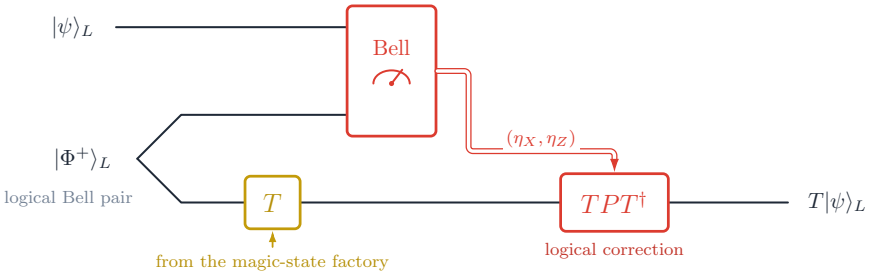

Figure 17.4: Sketch of how to use a magic state factory to perform a logical T gate on an encoded qubit.

logical qubits.

17.6 Magic state factories

The procedure can be iterated to produce increasingly accurate magic states. Let p_n denote the error after n rounds. Including the error floor from the protected circuitry gives the recursion

$$p_n \approx 35 p_{n-1}^3 + p_{\mathrm{circ},n}. \tag{17.29}$$

If the inner-code distance at every level is chosen so that

$$p_{\mathrm{circ},n} \lesssim 35 p_{n-1}^3, \tag{17.30}$$

then the idealized cubic recursion gives

$$p_n \approx 35^{(3^n-1)/2} p_0^{3^n} < \left(\sqrt{35}\, p_0\right)^{3^n}. \tag{17.31}$$

Thus later levels require larger-distance inner, for example surface, codes. For illustration, if $p_0 \approx 10^{-3}$, then $p_1 \approx 3.5 \times 10^{-8}$ and $p_2 \approx 1.5 \times 10^{-21}$.

Finally, a distilled state from the factory supplies the twisted Bell resource for a fault-tolerant gate-teleportation circuit acting on encoded data. Every logical T gate consumes one such resource state: see Figure 17.4.

This costly procedure is a major reason that estimates for the number of qubits required for fault-tolerant quantum computation are so large. In Section 18 we discuss how these logical measurements and state transfers can be implemented in a planar architecture.

Problems

Problem 17.1 (Alternative code concatenation scheme)**:** Find a generalization of the construction of concatenated CSS codes (Definition 17.5) to a situation where the inner code is $[[n_{\mathrm{in}}, k_{\mathrm{in}}, d_{\mathrm{in}}]]$ and the outer code is $[[mk_{\mathrm{in}}, k_{\mathrm{out}}, d_{\mathrm{out}}]]$ where $m > 1$ is an integer, and we wish to make a code

with parameters $[[mn_{\text{in}}, k_{\text{out}}, d]]$. How would you do it? What is the bound on d that you find?

Problem 17.2 (Universal "transversal" computation)**:** Consider a concatenated CSS code where the 7-qubit Steane code is the outer code and the 15-qubit Reed-Muller code is the inner code. We will show how to implement logical H and T fault-tolerantly and "transversally" within this single code, in a sense that we will make concrete below. Before we do this, it is useful to organize the 105 qubits of the code into an array (α, i), where $\alpha = 1, \ldots, 7$ denotes the outer index of the Steane code, and $i = 1, \ldots, 15$ denotes the inner index of the Reed-Muller code.

A: What are k and d for the concatenated code, which has $n = 105$ qubits?

B: Let H_{15} denote a (non-transversal) logical Hadamard gate on the Reed-Muller code. Explain why the 105-qubit code has a "transversal" logical Hadamard

$$H_{\text{L}} = \bigotimes_{\alpha=1}^{7} (H_{15})_{\alpha}. \tag{17.32}$$

Further explain why any one fault in this process is a correctable error.

C: Explain why the gate

$$T_{\text{L7}} := \text{CNOT}_{5\to 6}\text{CNOT}_{6\to 7} T_7 \text{CNOT}_{6\to 7}\text{CNOT}_{5\to 6} \tag{17.33}$$

implements a logical T gate on the 7-qubit Steane code (given the stabilizers in Example 15.12).

D: Use the method above to show how to implement a logical T (or $T^\dagger$) gate on the 105-qubit code by a sequence of "transversal" gates. Be sure to explain what partition of the 105 qubits is being used when defining transversal here. Explain why you can correct for any single fault that occurs during this process.

E: Conclude that if we perform stabilizer error-correction on the 105-qubit code after applying each logical T or H gate found above, if the probability p of any single-qubit error anywhere in the process is sufficiently small, then the probability of logical errors in *either* gate is $\text{O}(p^2)$. You do not need to optimize the prefactor; the point is that this procedure has successfully suppressed the logical error rate below the physical one for small enough p.

F: Identify the "loophole" in the Eastin-Knill Theorem we are taking advantage of here, and give a physical picture for why we can implement (arbitrarily good approximations to) any logical gate using this code without violating the intuition from the proof sketch of Theorem 16.8.

18 Lattice surgery

So far, we have seen how to implement several logical gates transversally in CSS codes. A difficulty arises when we try to implement a transversal CNOT between two copies of a two-dimensional code: every qubit in the control block must interact with the corresponding qubit in the target block. This requires the two two-dimensional architectures to be aligned, which may be possible in some hardware architectures but is difficult in a strictly planar layout, such as a superconducting-qubit architecture: see Figure 18.1. This motivates an alternative implementation of a logical CNOT that preserves planarity.

Definition 18.1 (Lattice surgery)**:** Lattice surgery is a non-transversal method for applying fault-tolerant joint logical Pauli measurements; in particular, a logical CNOT, in a two-dimensional planar code. In turn, state injection and magic-state factories can then be assembled.

We will focus on the logical CNOT. The same entangling operation will also be useful for magic-state injection and distillation. The essential geometric operations are a *merge*, in which two surface-code patches become one, and a *split*, in which one patch is divided into two: see Figure 18.2.

18.1 Merging and splitting surface-code patches

18.1.1 A rough merge

Consider two surface-code patches, as illustrated in Figure 18.3, whose rough boundaries face one another. To merge them, introduce a strip of interface qubits initialized in $|0\rangle$. The new Z-type checks are compatible with this initialization. We then repeatedly measure the new X-type checks across the interface. The product of their outcomes measures the joint logical operator

$$\eta = \prod_{a \in \mathrm{seam}_X} \eta_a \in \{+1, -1\}, \qquad \overline{X}_1 \overline{X}_2 = \eta \quad \text{on the post-measurement state.} \tag{18.1}$$

A smooth merge is analogous, with X and Z interchanged.

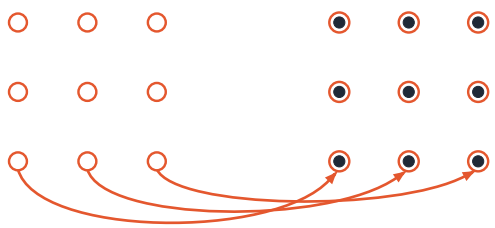

Figure 18.1: To implement a transversal logical CNOT in two dimensions, if gates are spatially local, one array of qubits (red) need to be physically moved to be adjacent to the black code.

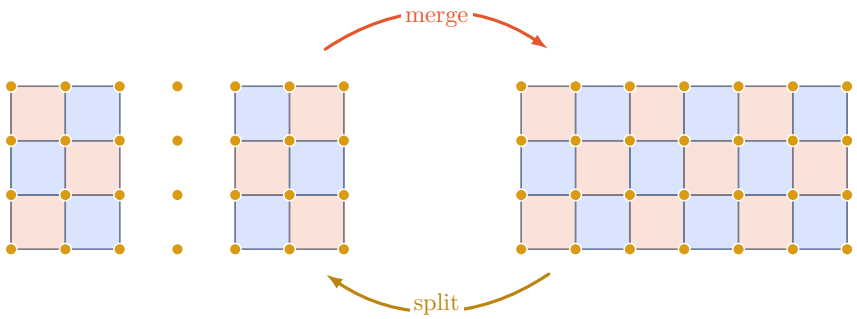

Figure 18.2: The two main operations of lattice surgery are the merge and split.

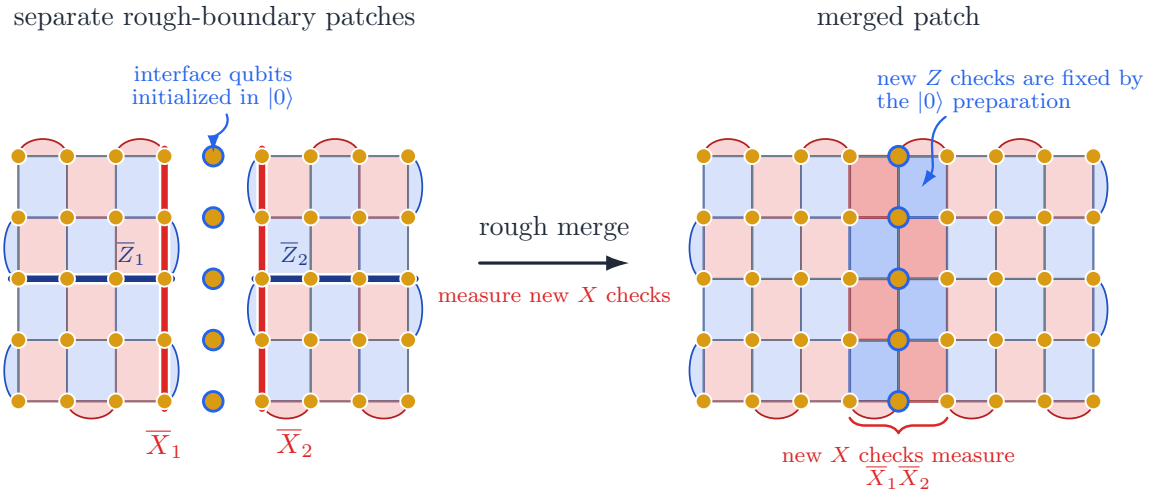

Figure 18.3: The merge operation in lattice surgery between two rough boundaries.

Let the two logical input states be arbitrary. Before the merge,

$$|\psi_{\text{before}}\rangle = (\alpha_1 |\overline{0_1}\rangle + \beta_1 |\overline{1_1}\rangle) \otimes |0\rangle_{\text{merge}} \otimes (\alpha_2 |\overline{0_2}\rangle + \beta_2 |\overline{1_2}\rangle). \tag{18.2}$$

Here $|0\rangle_{\text{merge}}$ is shorthand for all interface qubits being initialized in the computational-basis state $|0\rangle$. It is useful to define the logical Bell states

$$|\Phi_\pm\rangle = \frac{|\overline{0_1}\rangle |\overline{0_2}\rangle \pm |\overline{1_1}\rangle |\overline{1_2}\rangle}{\sqrt{2}}, \tag{18.3a}$$

$$|\Psi_\pm\rangle = \frac{|\overline{0_1}\rangle |\overline{1_2}\rangle \pm |\overline{1_1}\rangle |\overline{0_2}\rangle}{\sqrt{2}}. \tag{18.3b}$$

Conditioned on the decoded joint-measurement outcome η, the state after the merge is

$$|\psi_{\text{after}}\rangle \propto \begin{cases} (\alpha_1\alpha_2 + \beta_1\beta_2)\,|\Phi_+\rangle + (\alpha_1\beta_2 + \alpha_2\beta_1)\,|\Psi_+\rangle\,, & \eta = +1, \\ (\alpha_1\alpha_2 - \beta_1\beta_2)\,|\Phi_-\rangle + (\alpha_1\beta_2 - \alpha_2\beta_1)\,|\Psi_-\rangle\,, & \eta = -1. \end{cases} \tag{18.4}$$

The proportionality symbol in Eq. (18.4) is important: the conditional state must still be normalized according to Born's rule; the normalization depends on the probability of the observed value of η.

For the $\eta = +1$ outcome, the merged patch L' may be assigned the logical basis

$$\left|\overline{0}'\right\rangle = |\Phi_+\rangle\,, \tag{18.5a}$$

$$\left|\overline{1}'\right\rangle = |\Psi_+\rangle\,. \tag{18.5b}$$

The joint operator $\overline{X}_1\overline{X}_2$ is now a stabilizer of the merged code. For the $\eta = -1$ outcome, one can retain the same logical basis and record a Pauli byproduct E satisfying

$$E\left|\overline{0}'\right\rangle = |\Phi_-\rangle\,, \tag{18.6a}$$

$$E\left|\overline{1}'\right\rangle = |\Psi_-\rangle\,. \tag{18.6b}$$

The byproduct need not be physically applied; it can be retained in the Pauli frame. With the basis convention in Eq. (18.5), one may choose

$$E = \overline{Z}_1. \tag{18.7}$$

Modulo the new stabilizer, convenient logical representatives of the merged patch are

$$\overline{X}' \sim \overline{X}_1 \sim \eta\,\overline{X}_2, \qquad \overline{Z}' = \overline{Z}_1\overline{Z}_2. \tag{18.8}$$

Fault tolerance requires d rounds of syndrome extraction and decoding for a distance-d code. Thus η should be understood as the decoded logical parity inferred from the spacetime syndrome history, rather than merely the product of one noisy round of check measurements.

The merge has removed one logical qubit and collapsed the input state according to a fault-tolerant measurement of $\overline{X}_1\overline{X}_2$.

18.1.2 A rough split

Lattice splitting reverses the geometric procedure: see Figure 18.4. Measure the interface qubits transversally in the Z basis. This destroys the X-type stabilizers that crossed the interface, while preserving the information in the Z-type stabilizers. For example, if an old stabilizer crossing the

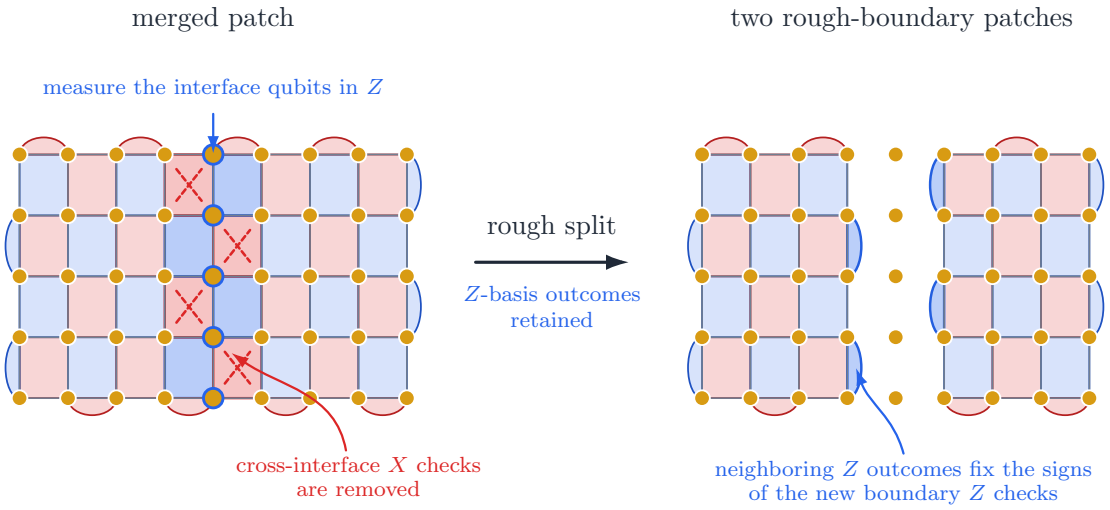

Figure 18.4: The split operation in lattice surgery creating two rough boundaries.

cut is $Z_aZ_bZ_cZ_d$ and the measured qubits give outcomes η_c and η_d, then

$$Z_aZ_bZ_cZ_d = +1, \qquad Z_c = \eta_c, \qquad Z_d = \eta_d \quad \Longrightarrow \quad Z_aZ_b = \eta_c\eta_d, \tag{18.9}$$

up to error correction and Pauli-frame tracking.

Suppose the merged patch initially stores

$$|\psi_{\text{before}}\rangle = \alpha \left|\overline{0'}\right\rangle + \beta \left|\overline{1'}\right\rangle . \tag{18.10}$$

Within the merged codespace,

$$\overline{X}_2 = \overline{X}_1 = \overline{X}', \tag{18.11}$$

and these representatives are unaffected by the Z-basis measurements. The two daughter patches are therefore stabilized, up to frame tracking, by $\overline{X}_1\overline{X}_2$. Their final state is

$$|\psi_{\text{after}}\rangle = \frac{\alpha+\beta}{\sqrt{2}} |\overline{+_1}\rangle\, |\overline{+_2}\rangle + \frac{\alpha-\beta}{\sqrt{2}} |\overline{-_1}\rangle\, |\overline{-_2}\rangle , \tag{18.12}$$

again up to Pauli-frame tracking. The information has not been destroyed by the split; instead, it has been distributed into two entangled logical qubits.

To retain distance d, the parent patch must be wide enough that both daughter patches have distance d. The interface measurements must also be included in the d-round spacetime syndrome history used by the decoder.

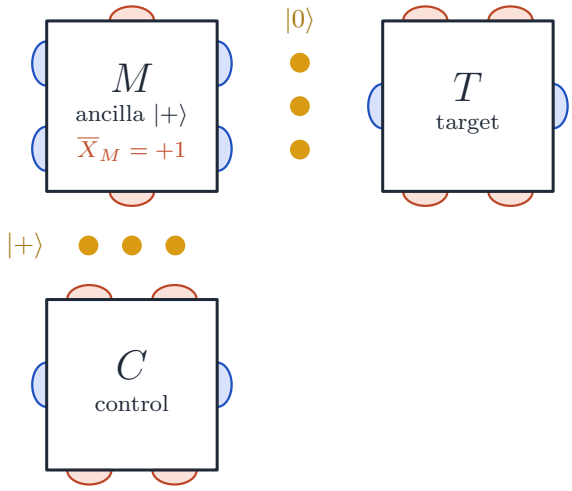

Figure 18.5: The patches C, M and T with their smooth and rough boundaries carefully aligned in preparation for a logical $\overline{\mathrm{CNOT}}_{C\to T}$ gate to be applied.

18.2 A logical CNOT from merge and split operations

We now combine these operations to implement a fault-tolerant logical $\overline{\mathrm{CNOT}}_{C\to T}$. Prepare three surface-code patches: a control patch C, a target patch T, and an ancilla patch M, with rough and smooth boundaries carefully aligned, as in Figure 18.5. Initialize the ancilla in $|\overline{+}_M\rangle$, equivalent so that this code patch has stabilizer $\overline{X}_M$. The protocol is illustrated in Figure 18.6:

1. merge C and M so as to measure $\overline{Z}_C\overline{Z}_M$, obtaining outcome η_1;
2. split C and M;
3. merge M and T so as to measure $\overline{X}_M\overline{X}_T$, obtaining outcome η_3;
4. destructively measure $\overline{Z}_M$, obtaining outcome η_4.

We can verify the protocol by tracking stabilizers and logical operators.

Step 1. The initial ancilla stabilizer $\overline{X}_M$ anticommutes with the measured operator. It is replaced by the post-measurement stabilizer

$$S_1 = \eta_1\overline{Z}_C\overline{Z}_M. \tag{18.13}$$

A convenient pair of logical representatives for the control is

$$(\overline{X}_C, \overline{Z}_C) \longmapsto (\overline{X}_C\overline{X}_M, \overline{Z}_C), \tag{18.14}$$

while the target logicals are clearly unaffected.

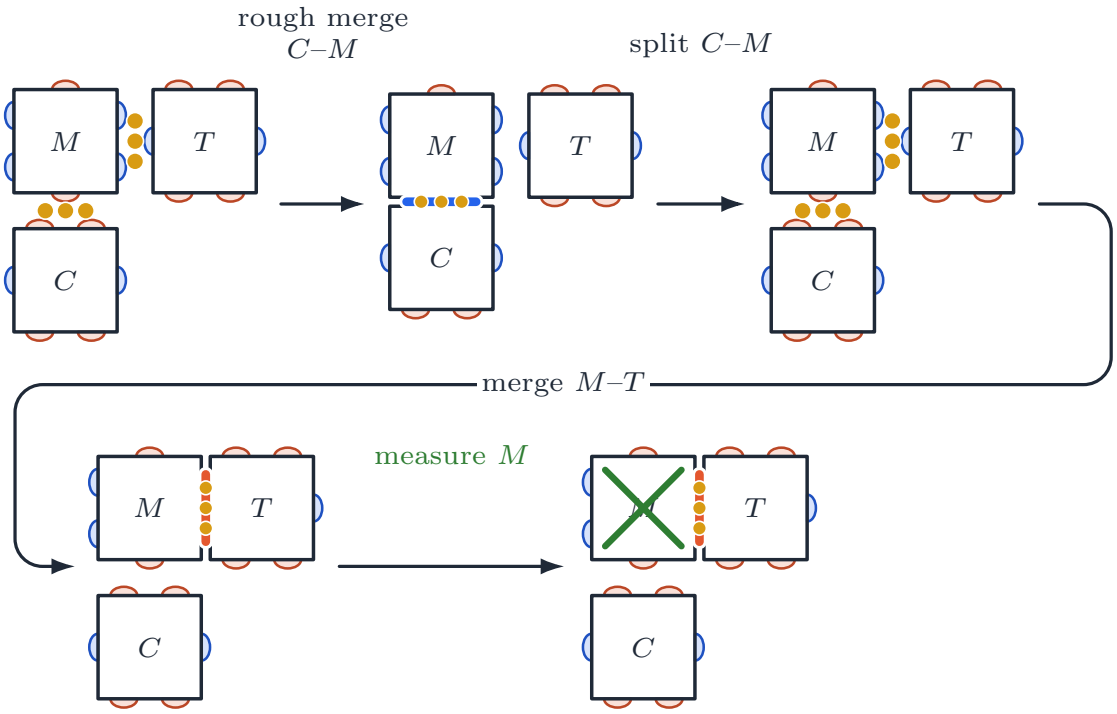

Figure 18.6: Steps 1-4 used to implement logical $\overline{\text{CNOT}}_{C\to T}$ between C and T, mediated by M.

Step 2. After splitting C and M, the stabilizer in Eq. (18.13) remains. The target logicals may be represented as

$$\overline{X}_T^{(2)} = \overline{X}_T, \tag{18.15a}$$

$$\overline{Z}_T^{(2)} = \eta_1 \overline{Z}_T \overline{Z}_M \overline{Z}_C. \tag{18.15b}$$

The second expression differs from the original $\overline{Z}_T$ only by the stabilizer S_1.

Step 3. The rough merge of M and T introduces the new stabilizer

$$S_3 = \eta_3 \overline{X}_M \overline{X}_T. \tag{18.16}$$

Using this stabilizer, the X-type logical representatives become

$$(\overline{X}_C \overline{X}_M, \overline{X}_T) \sim (\eta_3 \overline{X}_C \overline{X}_T, \overline{X}_T). \tag{18.17}$$

The corresponding Z-type representatives are

$$(\overline{Z}_C, \eta_1 \overline{Z}_T \overline{Z}_M \overline{Z}_C). \tag{18.18}$$

The representative of the target Z logical in Eq. (18.18) commutes with the new stabilizer because

it contains both $\overline{Z}_M$ and $\overline{Z}_T$. Here $\sim$ denotes equality modulo the current stabilizer group.

Step 4. Finally, measure the ancilla patch transversally in the Z basis and decode the logical outcome

$$\overline{Z}_M = \eta_4, \qquad \eta_4 \in \{+1, -1\}. \tag{18.19}$$

The logical operators on the surviving control and target patches transform as

$$\overline{X}_C \longmapsto \eta_3 \overline{X}_C \overline{X}_T, \tag{18.20a}$$
$$\overline{X}_T \longmapsto \overline{X}_T, \tag{18.20b}$$
$$\overline{Z}_C \longmapsto \overline{Z}_C, \tag{18.20c}$$
$$\overline{Z}_T \longmapsto \eta_1 \eta_4 \overline{Z}_C \overline{Z}_T. \tag{18.20d}$$

Ignoring the known signs, Eq. (18.20) is exactly the Heisenberg action of $\mathrm{CNOT}_{C\to T}$. The measurement outcomes are retained in the Pauli frame.

18.3 State injection

A lattice-surgery CNOT, together with suitable logical single-qubit Clifford gates and a high-quality non-Clifford resource state, gives a universal fault-tolerant gate set. Magic-state injection supplies the non-Clifford resource.

One begins with a single physical qubit in the state

$$|\psi\rangle_C = \alpha\,|0\rangle_C + \beta\,|1\rangle_C\,. \tag{18.21}$$

Initialize nearby qubits in $|0\rangle$, use CNOT gates to incorporate them into a small encoded region, and begin measuring the new surface-code checks. Any known syndrome signs are corrected or retained in the Pauli frame. The small patch can then be rough-merged with a larger surface-code patch initialized in $|\overline{0}\rangle$. Repeating the same growth procedure produces a full-distance logical patch carrying the injected state. This process is depicted in Figure 18.7.

This initial injection is not fault tolerant: a fault on the physical seed can become a logical fault. If the physical error rate is sufficiently small, the remaining logical error is therefore limited by the injection quality and can be suppressed by magic-state distillation. Growing the patch protects the state against subsequent faults, but it cannot remove an error that was already imprinted on the logical degree of freedom at the moment of injection. Distillation is what suppresses that inherited error.

There are also planar code-deformation constructions for a logical Hadamard gate, but we will not develop them here.

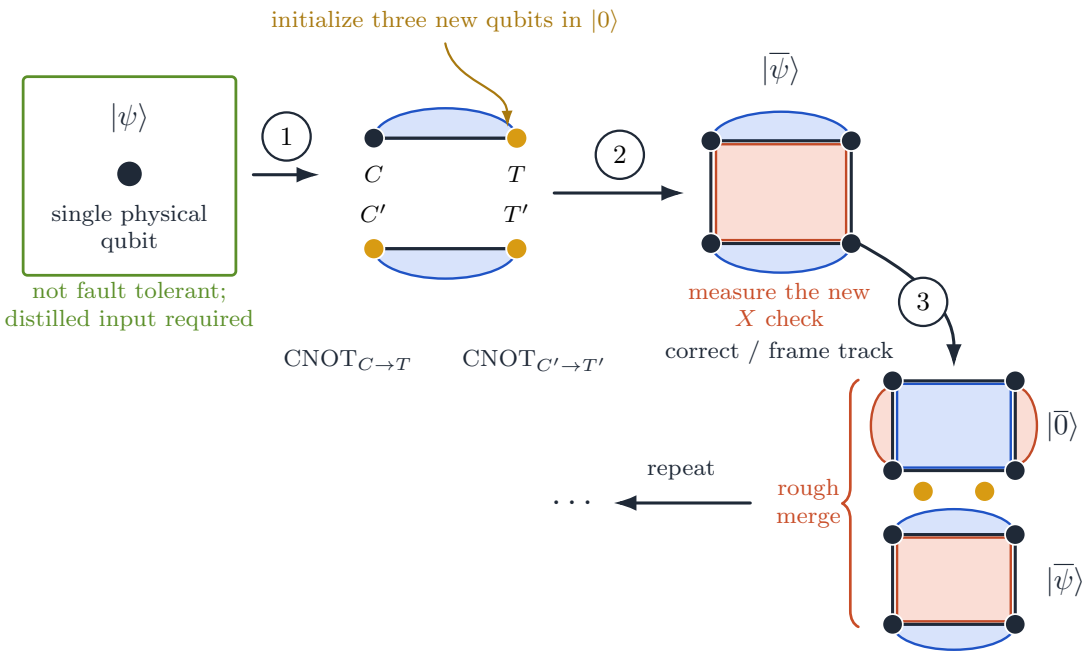

Figure 18.7: A non-optimized strategy for growing a surface code patch encoding a single logical qubit. We stress that this process is *not* fault-tolerant; distillation is necessary in order to make it fault-tolerant.

18.4 A schematic resource estimate

We now have the ingredients needed to build a useful fault-tolerant quantum computer. Let us make a schematic, deliberately unoptimized estimate of the required resources. Suppose that the computation uses k logical qubits, contains M logical T gates, has physical error rate p, and requires a logical gate and distillation error rate of order ϵ. We lay out surface code patches as depicted in Figure 18.8: some patches of surface code are used to store logical qubits, while others are used for magic state injection and distillation, along with gate teleportation (Section 17) in order to apply the logical non-Clifford gates.

A commonly used heuristic estimate for a distance-d surface code is

$$\epsilon \sim (100p)^{(d+1)/2}. \tag{18.22}$$

Solving for the distance gives

$$d \sim \frac{2\log(1/\epsilon)}{|\log(100p)|} - 1. \tag{18.23}$$

For $p \sim 10^{-3}$, this is approximately $d \sim 2\log_{10}(1/\epsilon) - 1$. If the circuit needs to run for t time steps, and ϵ is the logical error rate for both the logical memory and distillation (take it to be the larger

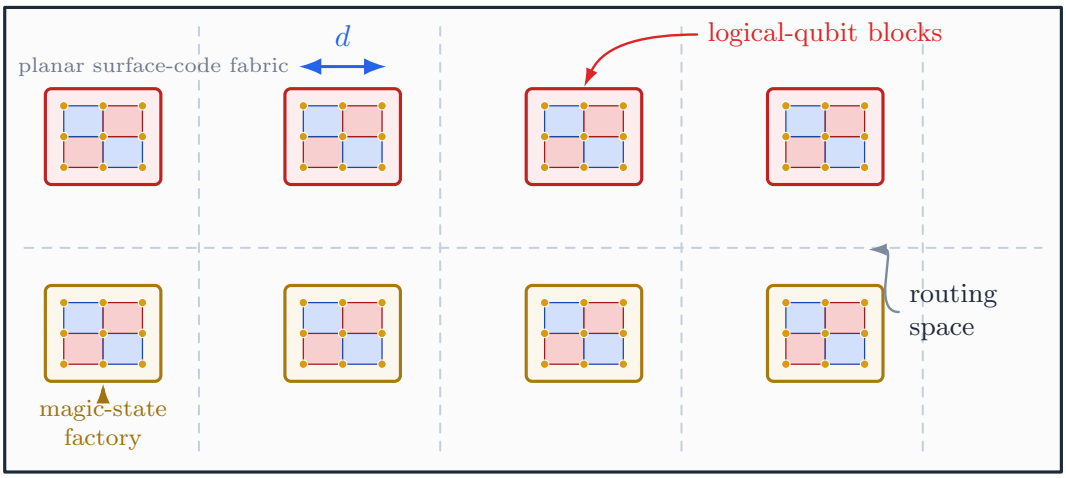

Figure 18.8: A schematic layout of a planar two-dimensional quantum computer based on the surface code. Logical qubits are routed via the same merge operations from Section 18.1.1 through the gaps in the architecture between logical blocks and magic state factories. d represents the code distance of each surface code patch, and should be set as in (18.23).

of the two for convenience), then the simple requirement

$$\epsilon(M + kt) \ll 1 \tag{18.24}$$

keeps the total probability of logical failure arising anywhere in the circuit small. Taking $\epsilon \sim 10^{-15}$, $M \sim 10^8$, and $p \sim 10^{-3}$ gives $d \sim 29$. For $k \sim 10^3$ logical qubits, a rough estimate of the physical-qubit count is

$$n \sim \mathrm{O}(1) \cdot kd^2 \sim 2 \times 10^6. \tag{18.25}$$

Here d^2 is the characteristic area of a logical patch and the O(1) factor, which we estimated conservatively at around 2, allows extra space for routing and magic-state factories.

For the illustrative parameters above, two levels of 15-to-1 magic-state distillation are sufficient: recall from Section 17.6 that two levels of distillation with $p \sim 10^{-3}$ give error probabilities of 10^{-21}, which would be below our target ϵ and likely limited by logical failures in any event.

Ignoring rejected batches, producing one twice-distilled state requires $15^2 = 225$ raw injected states. Thus the total number of raw injections is $N_{\mathrm{raw}} = 225M \sim 2 \times 10^{10}$. The first distillation level can be carried out in smaller surface-code patches. The number of first-level 15-to-1 blocks whose outputs must feed the large-distance level is $N_{\mathrm{level\,1}} = 15M \sim 10^9$. Suppose that the architecture can process γk of these first-level outputs in parallel, where $\gamma = \mathrm{O}(1)$. The number of

sequential large-block factory batches is then approximately

$$N_{\text{batches}} \sim \frac{15M}{\gamma k} \sim \frac{10^9}{\gamma k} \sim 10^6. \tag{18.26}$$

A fault-tolerant distillation batch has a depth of order

$$D_{\text{batch}} \sim d \times \mathrm{O}(10) \tag{18.27}$$

in surface-code cycles, where the constant depends on the detailed circuit used for distillation; much work goes into optimizing such constants! This gives a total runtime of order

$$t \sim 3 \times 10^8 \text{ cycles}. \tag{18.28}$$

If one cycle takes $\Delta t \sim 10\,\mu$s, then the runtime is approximately 1 hour. Thus this deliberately schematic estimate gives a "fast" computation time using on the order of a million physical qubits.

This is an enormous hardware cost, although the possibility of breaking RSA has been enough to motivate substantial interest in the subject. More modern quantum LDPC (Section **??**) codes may drastically reduce these costs, particularly the physical-qubit count. The estimate above is intended only as an order-of- magnitude illustration, not as a precise resource count for RSA factoring; such estimates continue to improve rapidly anyway.

19 Subsystem codes and code switching

After discussing magic-state distillation, we now turn to a different route to a universal fault-tolerant gate set: switch between codes with complementary transversal gates. Schematically, we seek a code A which has a transversal $\overline{T}$ logical gate, but not transversal $\overline{H}$; code B with transversal $\overline{H}$ but not $\overline{T}$; and a fault-tolerant way to switch between code A and code B while protecting the logical information.

19.1 The 9-qubit Bacon–Shor code

A natural entry point comes from the 9-qubit Shor code. In the qubit ordering shown in the figure below, its stabilizer group is

$$\begin{aligned}\mathcal{S}_{\text{Shor}} := \langle & X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9, \\ & Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9\rangle.\end{aligned} \tag{19.1}$$

The two weight-six X-checks are comparatively cumbersome to measure directly with a circuit-level fault-tolerant gadget. We will instead infer their outcomes from weight-two measurements.

Define the X- and Z-type gauge-generator subgroups

$$\mathcal{G}_X := \langle X_1X_4, X_4X_7, X_2X_5, X_5X_8, X_3X_6, X_6X_9\rangle\,, \tag{19.2a}$$

$$\mathcal{G}_Z := \langle Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9\rangle\,, \tag{19.2b}$$

$$\mathcal{G}_{\text{BS}} := \langle \mathcal{G}_X, \mathcal{G}_Z\rangle \subseteq \mathcal{P}_9. \tag{19.2c}$$

The group $\mathcal{G}_{\text{BS}}$ is non-Abelian: an XX bond and a ZZ bond anticommute whenever they overlap on exactly one qubit. A useful way to organize $\mathcal{G}_X$ and $\mathcal{G}_Z$ is as horizontal and vertical bonds in a 3×3 grid, as shown in Figure 19.1.

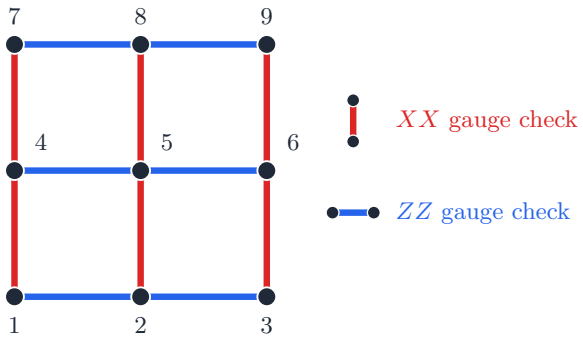

Figure 19.1: $\mathcal{G}_X$ and $\mathcal{G}_Z$ for the 9-qubit Bacon-Shor code.

19.1.1 Logical operators

Definition 19.1 (Centralizer): For a subset H of a group K, its centralizer in K is

$$C_K(H) := \{g \in K : gh = hg \text{ for every } h \in H\}. \tag{19.3}$$

It is convenient to denote the three full-row X operators and three full-column Z operators by

$$R_1^X := X_1X_2X_3, \qquad R_2^X := X_4X_5X_6, \qquad R_3^X := X_7X_8X_9, \tag{19.4a}$$

$$C_1^Z := Z_1Z_4Z_7, \qquad C_2^Z := Z_2Z_5Z_8, \qquad C_3^Z := Z_3Z_6Z_9. \tag{19.4b}$$

The centralizer of the gauge group is

$$C_{\mathcal{P}_9}(\mathcal{G}_{\mathrm{BS}}) = \left\langle R_1^X, R_2^X, R_3^X, C_1^Z, C_2^Z, C_3^Z \right\rangle. \tag{19.5}$$

Its intersection with the gauge group is generated by the four extended checks

$$\begin{aligned} \mathcal{S}_{\mathrm{BS}} := C_{\mathcal{P}_9}(\mathcal{G}_{\mathrm{BS}}) \cap \mathcal{G}_{\mathrm{BS}} = \langle & R_1^X R_2^X, R_2^X R_3^X, \\ & C_1^Z C_2^Z, C_2^Z C_3^Z \rangle. \end{aligned} \tag{19.6}$$

In particular, the ordinary Shor code is obtained by fixing the Z-type gauge generators:

$$\mathcal{S}_{\mathrm{Shor}} = \langle \mathcal{S}_{\mathrm{BS}}, \mathcal{G}_Z \rangle. \tag{19.7}$$

If we treated only $\mathcal{S}_{\mathrm{BS}}$ as the stabilizer group of an ordinary stabilizer code, the four independent checks would leave five logical qubits. We choose

$$\overline{X} := R_1^X = X_1X_2X_3, \qquad \overline{Z} := C_1^Z = Z_1Z_4Z_7 \tag{19.8}$$

as representatives of the one protected logical qubit. The other four logical pairs are declared to be gauge degrees of freedom. For example, X_1X_4 and Z_1Z_2 form one anticommuting gauge pair. We do not encode protected information in these four gauge qubits.

19.1.2 Error correction modulo the gauge group

In the usual Shor gauge, measuring the six generators of $\mathcal{G}_Z$ detects single-qubit X errors. We can then measure the six generators of $\mathcal{G}_X$ to diagnose Z errors. Only the two products of X-gauge outcomes corresponding to the central stabilizers $R_1^X R_2^X$ and $R_2^X R_3^X$ are syndrome data for the protected qubit; the remaining four independent outcomes describe the gauge state.

After measuring the X-gauge generators, X errors in the same column differ by gauge transformations:

$$X_1 \sim_{\mathcal{G}} X_4 \sim_{\mathcal{G}} X_7, \qquad X_2 \sim_{\mathcal{G}} X_5 \sim_{\mathcal{G}} X_8, \qquad X_3 \sim_{\mathcal{G}} X_6 \sim_{\mathcal{G}} X_9. \tag{19.9}$$

Thus the only meaningful single-X-error classes may be represented by X_1, X_2, X_3: the protected syndrome needs to determine only which column is in error. Individual X-gauge outcomes can appear to distinguish, for example, gauge-equivalent Z_1 and Z_2 errors, but those extra outcomes are random gauge information rather than protected syndrome information.

When we measure the Z-gauge generators again, the symmetric equivalences are

$$Z_1 \sim_{\mathcal{G}} Z_2 \sim_{\mathcal{G}} Z_3, \qquad Z_4 \sim_{\mathcal{G}} Z_5 \sim_{\mathcal{G}} Z_6, \qquad Z_7 \sim_{\mathcal{G}} Z_8 \sim_{\mathcal{G}} Z_9. \tag{19.10}$$

Only the three classes represented by Z_1, Z_4, Z_7 are meaningful: the central syndrome determines which row contains the error. Alternating the X- and Z-gauge measurements therefore changes the gauge state, but not the protected logical state.

A dressed logical X operator must contain an X error in each of the three columns, while a dressed logical Z operator must contain a Z error in each of the three rows. Consequently,

$$\mathrm{wt}(\overline{X}_{\mathrm{dressed}}) \geq 3, \qquad \mathrm{wt}(\overline{Z}_{\mathrm{dressed}}) \geq 3, \qquad d = 3. \tag{19.11}$$

Every single-qubit Pauli error is therefore correctable modulo the gauge group.

The five-qubit codespace of the four central stabilizers factorizes as

$$\mathcal{C}(\mathcal{S}_{\mathrm{BS}}) \cong \mathcal{H}_{\mathrm{L}} \otimes \mathcal{H}_{\mathrm{G}} \cong \mathbb{C}^2 \otimes (\mathbb{C}^2)^{\otimes 4}. \tag{19.12}$$

The alternating measurements have the schematic action

$$|\psi\rangle_{\mathrm{L}} \otimes |g_1\rangle_{\mathrm{G}} \longrightarrow |\psi\rangle_{\mathrm{L}} \otimes |g_2\rangle_{\mathrm{G}} \longrightarrow |\psi\rangle_{\mathrm{L}} \otimes |g_3\rangle_{\mathrm{G}} \longrightarrow \cdots . \tag{19.13}$$

The gauge qubits are allowed to change because they do not contain the protected information.

19.2 Pauli subsystem stabilizer codes

The Bacon–Shor example motivates a general extension of the Pauli stabilizer formalism, which we introduced in Section 7.

Definition 19.2 (Center)**:** The center of a group K is

$$Z(K) := C_K(K), \tag{19.14}$$

namely the set of elements commuting with every element of K.

Definition 19.3 (Pauli subsystem stabilizer code)**:** Let $\mathcal{G} \subseteq \mathcal{P}_n$ be a possibly non-Abelian Pauli gauge group. Suppressing scalar phases when writing generators, its stabilizer group is the center

$$Z(\mathcal{G}) = \Phi_{\mathrm{ph}}\mathcal{S} \subseteq \mathcal{G} \subseteq \mathcal{P}_n. \tag{19.15}$$

Recall the definition of Φ_{ph} in (7.64). The protected codespace is the simultaneous $+1$ eigenspace of the phase-free stabilizer generators in $\mathcal{S}$. The dressed logical operators are Pauli operators commuting with the stabilizers, modulo operators that act only on gauge degrees of freedom. A convenient set of nontrivial representatives is

$$\mathcal{L}_{\mathrm{dressed}} := C_{\mathcal{P}_n}(\mathcal{S}) \setminus \Phi_{\mathrm{ph}}\mathcal{G}, \tag{19.16}$$

and the subsystem distance is

$$d := \min_{P \in \mathcal{L}_{\mathrm{dressed}}} \mathrm{wt}(P). \tag{19.17}$$

We denote a subsystem code by $[[n, k, r, d]]$, where k qubits are protected and r are gauge qubits.

Restoring scalar phases, the relevant group sizes are

$$|\Phi_{\mathrm{ph}}\mathcal{S}| = 4 \cdot 2^{n-k-r}, \qquad |\Phi_{\mathrm{ph}}\mathcal{G}| = 4 \cdot 2^{n-k+r}. \tag{19.18}$$

Equivalently, there are $n-k-r$ independent stabilizer generators and r independent anticommuting gauge pairs.

The effective logical Pauli group and the bare logical Pauli group are respectively

$$\mathcal{P}_{\mathrm{logical}} := \frac{C_{\mathcal{P}_n}(\mathcal{S})}{\Phi_{\mathrm{ph}}\mathcal{G}}, \tag{19.19a}$$

$$\mathcal{P}_{\mathrm{logical,bare}} := \frac{C_{\mathcal{P}_n}(\mathcal{G})}{\Phi_{\mathrm{ph}}\mathcal{S}}. \tag{19.19b}$$

Bare logical operators commute with every gauge transformation. Dressed logical representatives are more important for the distance because a logical action may be accompanied by an arbitrary gauge transformation.

We state without proof the subsystem analogue of the canonical form for stabilizer codes.

Proposition 19.4 (Canonical form for a Pauli subsystem code)**:** There exists a Clifford transformation after which the gauge group, stabilizer group, and bare logical Pauli group can be generated as

$$\mathcal{G} = \langle X_1, Z_1, \ldots, X_r, Z_r, Z_{r+1}, \ldots, Z_{n-k} \rangle, \tag{19.20a}$$

$$\mathcal{S} = \langle Z_{r+1}, \ldots, Z_{n-k} \rangle, \tag{19.20b}$$

$$\mathcal{P}_{\mathrm{logical,bare}} = \langle X_{n-k+1}, Z_{n-k+1}, \ldots, X_n, Z_n \rangle. \tag{19.20c}$$

Consequently,

$$\mathcal{C}(\mathcal{S}) \cong (\mathbb{C}^2)^{\otimes k} \otimes (\mathbb{C}^2)^{\otimes r} =: \mathcal{H}_{\mathrm{L}} \otimes \mathcal{H}_{\mathrm{G}}. \tag{19.21}$$

The canonical form makes the harmless measurement backaction explicit. On the codespace, every

Hermitian gauge operator has the form

$$g\big|_{\mathcal{C}(\mathcal{S})} = \mathbb{I}_{\mathrm{L}} \otimes g_{\mathrm{G}}. \tag{19.22}$$

If its measurement outcome is discarded, the state evolves by

$$\rho \longmapsto \frac{1}{2}\left(\rho + g\rho g\right), \tag{19.23}$$

which acts only on $\mathcal{H}_{\mathrm{G}}$. Conditioned measurement outcomes likewise change only the gauge state. Thus gauge measurement can scramble the physical state without corrupting the protected logical qubits.

Example 19.5 (Bacon-Shor code as a subsystem code)**:** Clearly, the 9-qubit Bacon-Shor code from Section 19.1 is a Pauli subsystem code with parameters [[9,1,4,3]].

19.3 CSS subsystem codes

We will usually be interested in subsystem CSS codes. Suppose the X-type and Z-type gauge generators are specified by binary matrices M_X and M_Z, respectively. In contrast to an ordinary CSS stabilizer code, in general

$$M_X M_Z^{\mathsf{T}} \neq 0, \tag{19.24}$$

because the gauge group need not be Abelian.

Introduce the classical spaces

$$C_X^{\perp} := \mathrm{im}(M_X^{\mathsf{T}}), \qquad C_X := \ker(M_X), \tag{19.25a}$$

$$C_Z^{\perp} := \mathrm{im}(M_Z^{\mathsf{T}}), \qquad C_Z := \ker(M_Z). \tag{19.25b}$$

The X- and Z-type stabilizer subspaces are the gauge operators commuting with the full opposite-type gauge group:

$$\mathrm{im}(H_X^{\mathsf{T}}) = C_X^{\perp} \cap C_Z, \tag{19.26a}$$

$$\mathrm{im}(H_Z^{\mathsf{T}}) = C_Z^{\perp} \cap C_X. \tag{19.26b}$$

The dressed logical spaces are therefore

$$L_X := \frac{\ker(H_Z)}{\mathrm{im}(M_X^{\mathsf{T}})}, \tag{19.27a}$$

$$L_Z := \frac{\ker(H_X)}{\mathrm{im}(M_Z^{\mathsf{T}})}. \tag{19.27b}$$

The quotients mod out by both stabilizers and noncentral gauge operators.

19.4 Code switching by gauge fixing

Definition 19.6 (Code switching)**:** Code switching is a fault-tolerant procedure for moving between two quantum codes while preserving the encoded logical information, up to known logical or Pauli-frame transformations.

19.4.1 Gauge fixing

Gauge fixing is the special case relevant to this lecture. Two stabilizer codes arise as different commuting extensions of the center of one common gauge group:

$$\mathcal{S}_0 := Z(\mathcal{G}) \subseteq \mathcal{S}_A, \mathcal{S}_B \subseteq \mathcal{G}, \tag{19.28}$$

with common bare logical representatives in $C_{\mathcal{P}_n}(\mathcal{G})$. The same protected state is represented schematically as

$$\left|\overline{\psi}\right\rangle_A \cong |\psi\rangle_{\mathrm{L}} \otimes |g_A\rangle_{\mathrm{G}}\,, \qquad \left|\overline{\psi}\right\rangle_B \cong |\psi\rangle_{\mathrm{L}} \otimes |g_B\rangle_{\mathrm{G}}\,. \tag{19.29}$$

The gauge states can differ, while the logical factor is preserved by every operator in $\mathcal{G}$.

To switch from code A to code B, measure the generators of $\mathcal{S}_B$ that are not already fixed by $\mathcal{S}_0$. Then apply an appropriate gauge correction, or track it in a Pauli/gauge frame, while performing the necessary QEC. Reversing the roles of A and B reverses the switch.

19.4.2 A useful 15-qubit example

We now choose the two gauge fixings so that one exposes a transversal Hadamard gate and the other exposes a transversal T gate. Start with H_1 as the 4×15 matrix corresponding to the X-checks of the quantum Reed-Muller code in (16.14). The $[[15, 7, 3]]$ quantum Hamming code (recall Problem 8.1) has

$$H_X^{\mathrm{Ham}} = H_Z^{\mathrm{Ham}} = H_1. \tag{19.30}$$

Because the X- and Z-check matrices coincide, physical $H^{\otimes 15}$ acts as a logical Hadamard on each of the seven encoded qubits, in a suitable paired logical basis.

The $[[15, 1, 3]]$ quantum Reed–Muller code instead has

$$H_X^{\mathrm{RM}} = H_1, \qquad H_Z^{\mathrm{RM}} = \begin{pmatrix} H_1 \\ H_2 \end{pmatrix}, \qquad H_2 \in \mathbb{F}_2^{6\times 15}. \tag{19.31}$$

H_2 contains the remaining 6 rows of the Z-parity-check matrix in (16.15). It has the transversal logical phase gate

$$\overline{T}\left|\overline{\psi}\right\rangle = (T^\dagger)^{\otimes 15}\left|\overline{\psi}\right\rangle, \qquad \left|\overline{\psi}\right\rangle \in \mathcal{C}(\mathcal{S}_{\mathrm{RM}}), \tag{19.32}$$

with the inverse convention established earlier for this Reed–Muller code in Section 16.3.

To connect the two codes, reinterpret the Hamming code as a subsystem CSS code with gauge-generator matrices

$$M_X^{\mathrm{Ham}} = M_Z^{\mathrm{Ham}} = \begin{pmatrix} H_1 \\ H_2 \end{pmatrix}. \tag{19.33}$$

The H_1 rows generate the center, while the six H_2 rows in each Pauli sector form six gauge pairs. Thus this is a $[[15, 1, 6, 3]]$ subsystem code. The Reed–Muller code is obtained by fixing the Z-type H_2 gauge operators:

$$\mathcal{S}_{\mathrm{RM}} = \langle H_{1,X}, H_{1,Z}, H_{2,Z} \rangle . \tag{19.34}$$

Here $H_{j,X}$ and $H_{j,Z}$ denote the subgroups generated by the X- and Z-type Pauli operators associated with the rows of H_j.

Now start with $\left|\overline{\psi}\right\rangle$ encoded in the Reed–Muller gauge fixing. As a codeword of the seven-logical-qubit Hamming code, it can be viewed as one protected logical qubit with the six gauge qubits fixed in the logical Z basis. Therefore

$$H^{\otimes 15} \left|\overline{\psi}\right\rangle_{\mathrm{RM}} = H^{\otimes 15} \left(\left|\overline{\psi}\right\rangle_{\mathrm{L}} \otimes |\overline{0}\rangle_{\mathrm{G}}^{\otimes 6} \right) = \overline{H} \left|\overline{\psi}\right\rangle_{\mathrm{L}} \otimes |\overline{+}\rangle_{\mathrm{G}}^{\otimes 6} . \tag{19.35}$$

The protected logical Hadamard has been applied, but the six gauge qubits are now fixed in the complementary basis. To return to the Reed–Muller code, measure the six $H_{2,Z}$ stabilizers and apply, or frame-track, an $H_{2,X}$ gauge transformation with the required syndrome. This correction is defined only up to multiplication by the $H_{1,X}$ stabilizers. It is somewhat interesting how to actually perform this error correction: see Problem 19.1

The complete group structure, which is illustrated in Figure 19.2, is

$$\mathcal{G} = \langle H_{1,X}, H_{2,X}, H_{1,Z}, H_{2,Z} \rangle , \tag{19.36a}$$

$$\mathcal{S}_0 = \langle H_{1,X}, H_{1,Z} \rangle , \tag{19.36b}$$

$$\mathcal{S}_{\mathrm{RM}} = \langle H_{1,X}, H_{1,Z}, H_{2,Z} \rangle , \tag{19.36c}$$

$$\mathcal{S}_B = \langle H_{1,X}, H_{1,Z}, H_{2,X} \rangle . \tag{19.36d}$$

They obey

$$\mathcal{S}_0 \subseteq \mathcal{S}_{\mathrm{RM}}, \mathcal{S}_B \subseteq \mathcal{G}, \qquad H^{\otimes 15} : \mathcal{S}_{\mathrm{RM}} \longleftrightarrow \mathcal{S}_B. \tag{19.37}$$

A quick way to see that the protected logical qubit is common to every gauge fixing is to choose

$$\overline{X} = X^{\otimes 15}, \qquad \overline{Z} = Z^{\otimes 15}. \tag{19.38}$$

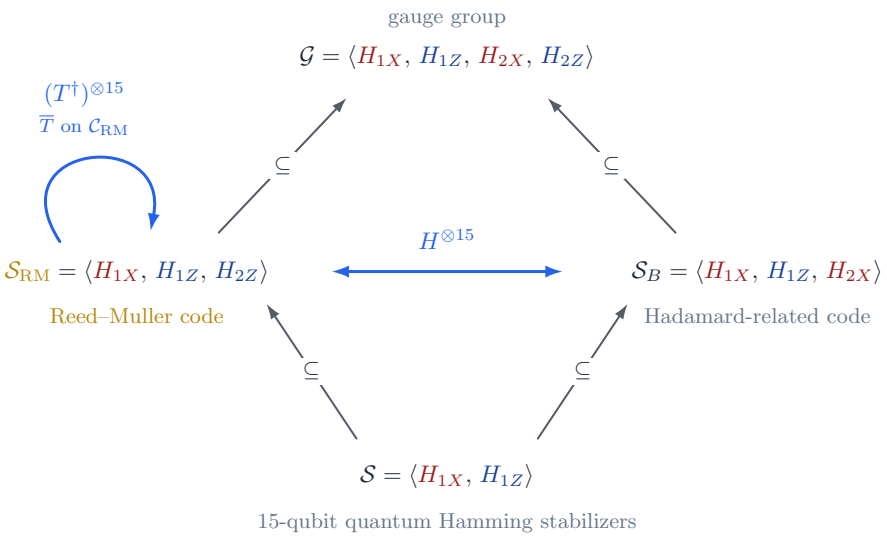

Figure 19.2: Understanding the universal gate set obtained in the 15-qubit (subsystem) codes via code switching.

Every row of H_1 and H_2 has even weight, so these operators commute with the entire gauge group. Since 15 is odd, they anticommute with one another, and

$$H^{\otimes 15} \overline{X} H^{\otimes 15} = \overline{Z}, \qquad H^{\otimes 15} \overline{Z} H^{\otimes 15} = \overline{X}. \tag{19.39}$$

Thus the desired logical Hadamard is visible without tracking the detailed motion of the six gauge qubits.

The subsystem distance $d = 3$ guarantees, at the code-capacity level, that any single-qubit data error remains correctable modulo the gauge group throughout the switch: $d = 3$ implies that every weight-one data Pauli error is correctable modulo $\mathcal{G}$. But of course, this is only a code-capacity statement. One-fault tolerance additionally requires a fault-tolerant circuit for measuring the $H_{2,Z}$ operators and a decoder that handles faulty measurement outcomes; it does not follow from the distance alone. Therefore, to obtain a scalable version of this strategy, in Section **??** we will seek topological code families in which complementary code choices or gauge fixings expose complementary transversal gates.

Problems

Problem 19.1 (Gauge fixing in the 15-qubit code)**:** Consider the code switching procedure in Section 19.4.2. Show that each potential error when gauge fixing from code $\mathcal{S}_B$ to the Reed-Muller code $\mathcal{S}_{\mathrm{RM}}$ corresponds to applying a correction (or frame tracking) associated to an element of $\mathrm{im}(H_{2,X}^{\mathsf{T}})$. To do this, think about the six rows of H_2 as labeled by a pair of 4 indices

$\{a, b\} \subset \{1, 2, 3, 4\}$, to think of $x \in \mathbb{F}_2^4 \setminus \{0\}$, and to define

$$(H_2)_{\{a,b\},x} := x_a x_b. \tag{19.40}$$

Calculate the matrix product $H_2 H_2^{\mathsf{T}}$ and ultimately establish the desired claim.