2023 Boulder lecture notes
Modern quantum error correction with 4 qubits
Victor V. Albert

0.1 Purpose and outline of these notes

QEC consists of subfields with distinct goals, including:

1. Construction of codes, either deterministic or random, that reach the boundary of what is possible.
   - MDS, perfect, random quantum, generalized homological product, good QLDPC, singleton-bound approaching approximate, covariant, locally testable, triorthogonal
2. Development of codes with near-term realization in mind.
   - 2-3D surface, 2-3D color, dynamically generated (Floquet, spacetime circuit), tetron Majorana, single-shot, self-correcting quantum, cluster-state, homological rotor
3. Direct realization of codes on a quantum device.
   - repetition, small distance block, 2D rotated surface, 2D color, two-component cat, square- and hexagonal-lattice GKP, dual-rail
4. Relating phases of quantum matter to error-correcting codes.
   - geometrically local Hamiltonian-based (topological, fracton, ETH, MPS)
5. Relating gravitational and other field theories to error-correcting codes.
   - holographic (HaPPY), renormalization group cat, matrix model
6. Development of codes for sensing/metrology.
   - Error-corrected sensing, metrological

These notes are not meant to be a deep dive into a particular topic of QEC, but instead a demonstration of a wide range of topics, gadgets, and ideas intended to achieve the various goals of the field. Particular attention is paid to ideas that can be conveyed at least partially using about four qubits. The intention is to make the reader more familiar with aspects of state-of-the-art quantum error correction so that they can more readily join the field. In particular, the latter part of the notes is meant to segue the reader to a more holistic framework that treats the entire error correction procedure as a dynamical process that does not necessarily revolve around a particular code.

These notes are not meant to be a proper historical overview of error correction; nor are they meant to be a review since they cover a nearly negligible fraction of the field.

0.2 Goal of error correction

The goal of error correction is to preserve messages sent through a noisy transmission channel by encoding the messages in an error-correcting code. More precisely, the goal is to make sure the rate of corruption of encoded (i.e., logical) information is lower than that of the same information sent without encoding.

A message ρ is encoded by the sender using an encoding map $\mathcal{E}$, passed through the noisy channel $\mathcal{N}$, and extracted by the receiver using a decoding map $\mathcal{D}$. If error correction succeeds, then the message should remain roughly intact such that noisy transmission is nearly equivalent to the noiseless case,

$$\mathcal{D}\mathcal{N}\mathcal{E}(\rho) \approx \rho\,. \tag{1}$$

In the classical case, a message can be represented by a binary string, and the set of messages can be thought to be sampled from some probability distribution. The state to be transmitted can be represented as a classical mixture ρ governed by the distribution.

Classical messages are encoded using an **alphabet** X, which can be the real numbers ($X = \mathbb{R}$), integers ($X = \mathbb{Z}$), a finite field ($X = \mathbb{F}_q$), or a ring ($X = R$). The simplest alphabet is the n-bit binary alphabet, where $X = \mathbb{Z}_2^n = \mathbb{F}_2^n$. For a one-bit message,

$$\rho = p_0|0\rangle\langle 0| + p_1|1\rangle\langle 1| \tag{2}$$

X	classical states (elements of X)	quantum states (functions on X)
$\mathbb{Z}_2^n = \mathbb{F}_2^n$	bits	qubits
$\mathbb{F}_q^n$	q-ary strings	Galois qudits
$\mathbb{Z}_q^n$	q-ary strings over $\mathbb{Z}_q$	modular qudits
$\mathbb{R}^n$	reals	oscillators
G	finite group	group-valued qudit

Table 1: Common classical alphabets and their corresponding quantum Hilbert spaces.

where $p_{0,1}$ are probabilities adding up to one.

Passing through a noisy channel can corrupt the state by exchanging the probabilities, $p_0 \leftrightarrow p_1$, via conjugation of the state by the Pauli matrix $X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, causing a logical bit flip. Without an encoding, there is no way of detecting that such an error occurred, much less reversing the error's effect.

An encoding maps the above logical information redundantly into bitstrings comprised of several physical bits. The repetition encoding into three physical bits, $0 \to 000$ and $1 \to 111$, allows for detection of at most one bit-flip event via majority vote. If two or more bits are flipped, majority vote yields the wrong result, and a logical bit-flip occurs.

Quantum messages are encoded using a **Hilbert space** of appropriate functions on an alphabet X, i.e., $\ell^2(X)$. The purpose of the ℓ^2 function space is to accommodate **superpositions** of classical codewords. The Hilbert space can be the space of a harmonic oscillator ($X = \mathbb{R}$), rotor ($X = \mathbb{Z}$), a modular qudit ($X = \mathbb{Z}_q$), a Galois qudit ($X = \mathbb{F}_q$), a finite group ($X = G$), or even a category ($X = \mathcal{C}$). The simplest Hilbert space is the n-qubit space, where $X = \mathbb{Z}_2^n = \mathbb{F}_2^n$. For a one-qubit message,

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle\,, \tag{3}$$

where $c_{0,1}$ are complex numbers whose norms add up to one. These coefficients represent the values of the **function** $\psi(x) \equiv \langle x|\psi\rangle$ at $x = 0, 1$, respectively.

Any classical encoding can be embedded into a quantum one. The encoding of the quantum version of the repetition code maps

$$|\psi\rangle \to |\psi_L\rangle = c_0|0_L\rangle + c_1|1_L\rangle = c_0|000\rangle + c_1|111\rangle \tag{4}$$

where $|\psi\rangle$ is called the **logical state**, and $|0_L\rangle, |1_L\rangle$ are called **logical codewords**.

Similar to the classical case, the quantum repetition code corrects single physical bitflip errors $\{XII, IXI, IIX\}$, where $XII = X \otimes I \otimes I$, and I is the two-by-two identity matrix. The majority-vote procedure can also be carried out using quantum circuits, but has to be performed carefully so as not to disturb the logical quantum superposition. The fact that this such a procedure is possible is one of the major advances of quantum error correction.

An error maps logical states to a different state, e.g.,

$$XII|\psi_L\rangle = XII\left(c_0|000\rangle + c_1|111\rangle\right) = c_0|010\rangle + c_1|101\rangle\,. \tag{5}$$

The error maps the codespace to an error space spanned by error words $|010\rangle$ and $|101\rangle$. Since the coefficients in the above superposition are the same as those in the original code state, the error has actually maintained the quantum information!

Since the above error space is orthogonal to the code space, there exist observables called **check operators** that can be used to distinguish the error space from the codespace without distinguishing states within the respective spaces. The error correction procedure consists of

1. detecting the bit-flip error via quantum measurement of check-operator eigenvalues, usually called syndromes, and
2. applying a correction operation that maps the state in the error space back into the codespace.

In order for the syndrome measurement not to collapse the logical quantum superposition, ancillary qubits have to be used, and every binary syndrome measurement requires access to one ancilla qubit. The correction operation is conditional on the syndrome measurement outcome: XII is applied when the error is detected, and similarly for the other bit-flip errors, and nothing is applied when no error is detected.

The procedure described above is specific to the repetition code, but similar procedures exist for other codes. Active measurement and correction has to be applied periodically in order to protect the quantum information;

otherwise, the information will decohere at a rate faster than if the information had been encoded in the trivial code consisting of the first physical qubit.

The above is not sufficient to protect quantum systems from noise. The reason is that, in addition to bit-flip noise, quantum superpositions can also be corrupted via conjugation of the state ρ by phase-flip errors $\{ZII, IZI, IIZ\}$, where the Pauli matrix $Z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$. This causes a logical phase flip, which maps $c_0 \leftrightarrow c_1$ and thus changes the relative phase of the logical superposition. True quantum encodings, as opposed to classical encodings used as quantum ones, thus have to protect against two types of noise.

The quantum repetition code cannot detect (or correct) phase-flip events because such errors maintain the logical state within the codespace. For example,

$$ZII|\overline{\psi}\rangle = c_0\left(ZII|000\rangle\right) + c_1\left(ZII|111\rangle\right) = c_0|0_L\rangle + c_1|1_L\rangle \tag{6}$$

Since the quantum information has not left the codespace, there is no way to detect that an error occurred. Therefore, a logical phase-flip is induced.

- HW: Why can't we just copy $|\psi\rangle \to |\psi\rangle|\psi\rangle|\psi\rangle$? Prove the no-cloning theorem.

0.3 My first quantum code

Four-qubit code detects both bit and phase errors. Codespace is

$$|0_L\rangle \propto |\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\rangle \tag{7}$$

$$|1_L\rangle \propto |\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\rangle \tag{8}$$

1. Bit flips are protected because Hamming distance between basis states making up distinct codewords is 2.
2. Phase flips are protected because there are multiple basis stafates per codeword. Each codeword is now itself a quantum superposition of two canonical basis states.
   (a) Entanglement (w.r.t. canonical basis) is required for multi-qubit error correction (given standard noise model).

Flipping the first qubit maps the codewords into error words

$$\begin{smallmatrix} X & I \\ I & I \end{smallmatrix}|0_L\rangle \propto |\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\rangle \propto |0_X\rangle \tag{9}$$

$$\begin{smallmatrix} X & I \\ I & I \end{smallmatrix}|1_L\rangle \propto |\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\rangle \propto |1_X\rangle\,. \tag{10}$$

Just as with the repetition code, an arbitrary superposition is maintained. Flipping the third qubit results in error words

$$\begin{smallmatrix} I & I \\ X & I \end{smallmatrix}|0_L\rangle \propto |\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\rangle \propto |1_X\rangle \tag{11}$$

$$\begin{smallmatrix} I & I \\ X & I \end{smallmatrix}|1_L\rangle \propto |\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\rangle \propto |0_X\rangle \tag{12}$$

These are the same as the error words (9), meaning that the two errors map to the same error space. When this occurs, the code is said to be **degenerate**. Since this error space is distinguishable from the code space, both errors can be detected using the same syndrome measurement. A similar effect occurs for the remaining bit-flip errors $\begin{smallmatrix} I & X \\ I & I \end{smallmatrix}$ and $\begin{smallmatrix} I & I \\ I & X \end{smallmatrix}$.

A syndrome measurement cannot determine which bit-flip error occurred because the errors act on the superposition in different ways,

$$\begin{smallmatrix} X & I \\ I & I \end{smallmatrix}\left(c_0|0_L\rangle + c_1|1_L\rangle\right) = \left(c_0|0_X\rangle + c_1|1_X\rangle\right) \tag{13}$$

$$\begin{smallmatrix} I & I \\ X & I \end{smallmatrix}\left(c_0|0_L\rangle + c_1|1_L\rangle\right) = \left(c_0|1_X\rangle + c_1|0_X\rangle\right) = \begin{smallmatrix} X & I \\ I & I \end{smallmatrix} \cdot \begin{smallmatrix} X & I \\ X & I \end{smallmatrix}\left(c_0|0_L\rangle + c_1|1_L\rangle\right)\,. \tag{14}$$

The first error maps the codewords to their corresponding error words, while the second maps the codewords to permuted error words. Flipping qubit one is equivalent to first undergoing a logical bit-flip X_L and then flipping qubit two.

Without the ability to distinguish errors that act differently within the same error space, it is impossible to map the superposition back into the code space without causing a logical bit-flip error. In other words, it is impossible to **correct** against both bit-flip errors at the same time because their corresponding recovery operations are different. Figuring out which recovery operation to apply for a given error space is called **decoding**.

A similar effect occurs for single phase flips. A Z-error on 2nd (or 4th) qubit acts as

$$\begin{aligned} \begin{smallmatrix} I & Z \\ I & I \end{smallmatrix}|0_L\rangle &\propto |\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\rangle - |\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\rangle \propto |0_Z\rangle && (15) \\ \begin{smallmatrix} I & Z \\ I & I \end{smallmatrix}|1_L\rangle &\propto |\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\rangle - |\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\rangle \propto |1_Z\rangle && (16) \end{aligned}$$

This means that both such phase flips are detectable and correctable. However, there is a problem with the remaining phase flips. A Z-error on 1st (or 3rd) qubit acts as

$$\begin{aligned} \begin{smallmatrix} Z & I \\ I & I \end{smallmatrix}|0_L\rangle &\propto \quad |\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\rangle - |\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\rangle \propto |0_Z\rangle && (17) \\ \begin{smallmatrix} Z & I \\ I & I \end{smallmatrix}|1_L\rangle &\propto -|\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\rangle \propto -|1_Z\rangle && (18) \end{aligned}$$

Once again, the resulting error spaces are the same for all phase flips, but the latter qubit phase errors apply an additional sign to $|1_L\rangle$ --- a logical phase-flip. Single phase-flip errors are detectable, but not correctable, since we cannot know if there is a logical phase flip that needs undoing.

Considering single-qubit bit and phase flips requires us to consider the effect of both types of errors occurring at the same time. A joint **bit-phase flip**, represented by the Pauli matrix $\begin{smallmatrix} I & I \\ Y & I \end{smallmatrix}$, acting on the 3rd qubit yields

$$\begin{aligned} \begin{smallmatrix} I & I \\ Y & I \end{smallmatrix}|0_L\rangle &\propto |\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\rangle - |\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\rangle \propto |0_Y\rangle && (19) \\ \begin{smallmatrix} I & I \\ Y & I \end{smallmatrix}|1_L\rangle &\propto |\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\rangle - |\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\rangle \propto |1_Y\rangle\,. && (20) \end{aligned}$$

Bit-phase errors thus map code state to a third error space, orthogonal to the X- and Z-error spaces. A similar argument as before shows that single-qubit Y errors are also detectable, but not correctable.

As described above, the four-qubit code can detect any single-qubit Pauli errors $\{X, Y, Z\}$, as well as the no-error identity case I. Six subspaces are utilized for this detection: two for X on any qubit, two for any Y, one for any Z, and the codespace corresponding to the trivial error I. Quite surprisingly, this is sufficient for correcting all superpositions of such errors.

Consider, for example, a continuous rotation error

$$R_\theta = \begin{pmatrix} e^{-i\theta} & 0 \\ 0 & e^{i\theta} \end{pmatrix} = \cos\theta I - i\sin\theta Z\,. \tag{21}$$

This rotation can be written as a superposition of the identity and Z --- a detectable Pauli error. As a result, a rotation on the second qubit maps the codespace to both itself and the Z-error space,

$$\begin{aligned} \begin{smallmatrix} I & R_\theta \\ I & I \end{smallmatrix}|\psi_L\rangle &= \cos\theta|\psi_L\rangle - i\sin\theta\ \begin{smallmatrix} I & Z \\ I & I \end{smallmatrix}|\psi_L\rangle && (22) \\ &= \cos\theta\left(c_0|0_L\rangle + c_1|1_L\rangle\right) - i\sin\theta\left(c_0|0_Z\rangle + c_1|1_Z\rangle\right)\ . && (23) \end{aligned}$$

Measurement of the error syndromes in the code's error detection procedure collapses the above superposition into a state that is *either* in the codespace *or* in the error space. In the latter case, an error is detected, while in the former, no error occurs. To quote S. M. Girvin: "noise is continuous, but measured errors are discrete!"

The above example is straightforwardly extended to general encodings because the identity and the three Pauli matrices can be used to express any single-qubit error, and n-fold tensor products of these four operators, called **Pauli strings**, do the same for n-qubit errors.

0.4 Error-detection and correction conditions

The ability to detect bit-flip errors $N_1 = \begin{smallmatrix} X & I \\ I & I \end{smallmatrix}$ and $N_2 = \begin{smallmatrix} I & I \\ X & I \end{smallmatrix}$ hinges on the fact that each error mapped the codespace into an orthogonal error space. This implies that a malicious environment able to measure the two errors cannot distinguish between the codewords,

$$\langle 0_L|N_j|0_L\rangle = \langle 1_L|N_j|1_L\rangle\ . \tag{24}$$

Detection ability also implies that the environment cannot perform logical gates, e.g., logical bit flips, within the codespace by applying the two errors,

$$\langle 0_L|N_j|1_L\rangle = \langle 1_L|N_j|0_L\rangle = 0\,. \tag{25}$$

The above two equations can be combined using the projection onto the codespace,

$$\Pi \equiv |0_L\rangle\langle 0_L| + |1_L\rangle\langle 1_L|\,, \tag{26}$$

into the **quantum error-detection conditions**

$$\mathit{\Pi} N_j \mathit{\Pi} = c_j \mathit{\Pi} \quad \text{for} \quad N_j \in \mathcal{N} \qquad \text{(error-detection conditions)} \tag{27}$$

where the consant $c_j = 0$ for the four-qubit code. These are the error-detection conditions for the error set $\mathcal{N} = \{N_1, N_2\}$. These are satisfied for all errors in a given error set iff all error in the set are detectable.

The complex constant c_j need not be zero in general. For example, consider the operator $ZZII$ acting on the four-qubit code,

$$\begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix} |0_L\rangle = |0_L\rangle \qquad \text{and} \qquad \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix} |1_L\rangle = |1_L\rangle \,, \tag{28}$$

which keeps the logical information within the codespace without affecting it. This operator does not cause an error on the code, and its projection $\mathit{\Pi} \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix} \mathit{\Pi} = \mathit{\Pi}$ yields a constant of one.

Bit-flip errors $N_{j=1,2}$ are not correctable because they map codewords into the same error space, but with outcomes differing by a logical bit flip $X_L = \begin{smallmatrix} X & I \\ X & I \end{smallmatrix}$. Equivalently, the two errors cannot undo each other. Trying to reverse the effect of N_2 using N_1 yields the residual logical operation,

$$\langle 1_L | N_1^\dagger N_2 | 0_L \rangle = \langle 1_L | \begin{smallmatrix} X & I \\ X & I \end{smallmatrix} | 0_L \rangle = \langle 1_L | X_L | 0_L \rangle = 1 \neq 0 \,. \tag{29}$$

The above is a violation of one of the Knill-Laflamme **error-correction conditions**,

$$\mathit{\Pi} N_j^\dagger N_k \mathit{\Pi} = c_{jk} \mathit{\Pi} \quad \text{for} \quad N_j, N_k \in \mathcal{N} \qquad \text{(error-correction conditions)} \tag{30}$$

Given a code basis, the diagonal matrix elements of the above conditions are called the **non-deformation conditions**, while off-diagonals are called **orthogonality conditions**. If these conditions are satisfied for all errors in a given error set, then those errors are correctable. The conditions ensure that any overlap between error spaces (i.e., when $c_{jk} \neq 0$) corresponds to errors that induce the same effect within the error space. The conditions hold for more general errors that, unlike Pauli strings, can be non-unitary, non-Hermitian, or even non-diagonalizable.

The error-correction conditions can be alternatively obtained by looking at what the environment obtains from the logical state. Complementary channel

$$\mathcal{N}(\rho_L) = \mathrm{tr}_{\mathrm{env}} \left\{ U_{\mathcal{N}} \rho_L U_{\mathcal{N}}^\dagger \right\} = \sum_j N_j \rho_L N_j^\dagger \tag{31}$$

$$\widehat{\mathcal{N}}(\rho_L) = \mathrm{tr}_{\mathrm{sys}} \left\{ U_{\mathcal{N}} \rho_L U_{\mathcal{N}}^\dagger \right\} = \sum_{j,k} \mathrm{tr} \left\{ N_j^\dagger N_k \rho_L \right\} |k\rangle\langle j| \tag{32}$$

Action of noise channel $\mathcal{N}$ can be thought of as an attempt by environment to extract expectation values of products of error operators. Plugging in the error-correction conditions, we see that the environment has not obtained any information about the logical state.

Channels whose noise operators are Pauli strings — Pauli channels — are much easier to deal with that non-Pauli channels. One important non-Pauli channel is **qubit amplitude damping** with no-jump and jump Kraus operators

$$N_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix} \qquad\qquad N_1 = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix} \tag{33}$$

for loss rate $p \in [0,1]$. The four-qubit code can **correct** a single amplitude damping error to order $O(p^2)$.

1. The collective no-jump operator $\begin{smallmatrix} N_0 & N_0 \\ N_0 & N_0 \end{smallmatrix}$ and the four single "jumps" — $\begin{smallmatrix} N_1 & N_0 \\ N_0 & N_0 \end{smallmatrix}$ and its three permutations — are detectable because single-qubit errors are detectable by our previous analysis.

2. But there's more: the four single jumps cause transitions to distinct qubit subspaces, and so do not overlap. For example, $\mathit{\Pi} \left(\begin{smallmatrix} N_1 & N_0 \\ N_0 & N_0 \end{smallmatrix} \right)^\dagger \begin{smallmatrix} N_0 & N_1 \\ N_0 & N_0 \end{smallmatrix} \mathit{\Pi} = 0$. For products of the same jump, the deformation conditions turn out to be satisfied, e.g.,

$$\langle 0_L | \left(\begin{smallmatrix} N_1 & N_0 \\ N_0 & N_0 \end{smallmatrix} \right)^\dagger \begin{smallmatrix} N_1 & N_0 \\ N_0 & N_0 \end{smallmatrix} | 0_L \rangle = \left(\sqrt{p/2} \left\langle \begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix} \right| \right) \left(\sqrt{p/2} \left| \begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix} \right\rangle \right) = p/2 \tag{34}$$

$$\langle 1_L | \left(\begin{smallmatrix} N_1 & N_0 \\ N_0 & N_0 \end{smallmatrix} \right)^\dagger \begin{smallmatrix} N_1 & N_0 \\ N_0 & N_0 \end{smallmatrix} | 1_L \rangle = \left(\sqrt{p/2} \left\langle \begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix} \right| \right) \left(\sqrt{p/2} \left| \begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix} \right\rangle \right) = p/2 \tag{35}$$

3. The no-jump Kraus operator yields

$$\begin{smallmatrix} N_0 & N_0 \\ N_0 & N_0 \end{smallmatrix} | 0_L \rangle = \left(\left| \begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix} \right\rangle + (1-p)^2 \left| \begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix} \right\rangle \right) / \sqrt{2} \tag{36}$$

$$\begin{smallmatrix} N_0 & N_0 \\ N_0 & N_0 \end{smallmatrix} | 1_L \rangle \propto (1-p) \left(\left| \begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix} \right\rangle + \left| \begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix} \right\rangle \right) / \sqrt{2} \tag{37}$$

so the error-correction conditions are satisfied to linear order in p,

$$\Pi \left(\begin{smallmatrix} N_0 & N_0 \\ N_0 & N_0 \end{smallmatrix}\right)^{\dagger} \begin{smallmatrix} N_0 & N_0 \\ N_0 & N_0 \end{smallmatrix} \Pi = (1-2p)\,\Pi + O(p^2) \tag{38}$$

Cases when Kraus operators are correctable to some order in the noise parameter, which occurs often with bosonic codes, are examples of what is sometimes called **approximate error correction**. This naming is meant to differentiate from cases where the Kraus operators are partitioned into a set of perfectly correctable and non-correctable operators, as is the case for Pauli channels.

1. How do operators transform? Heisenberg picture: evaluating logical L operator with code state $\rho_L = \Pi\rho_L\Pi$:

$$\begin{aligned} \langle L\rangle &= \mathrm{tr}\left(L\rho_L\right) && (39)\\ &\overset{!}{=} \mathrm{tr}\left(L\mathcal{D}\mathcal{N}(\rho_L)\right) && (40)\\ &= \mathrm{tr}\left((\mathcal{D}\mathcal{N})^{\dagger}(L)\rho_L\right) && (41) \end{aligned}$$

   This yields operator QEC conditions, namely, there exists a decoder $\mathcal{D}$ such that

$$\mathcal{D}\mathcal{N}(\rho_L) = \rho_L \qquad \leftrightarrow \qquad \Pi(\mathcal{D}\mathcal{N})^{\dagger}(L)\Pi = \Pi L \Pi \tag{42}$$

   for all logical operators L.

2. HW: complete the square:

$$\begin{array}{ccc} \text{QEC conditions} & \leftrightarrow & \text{Complementary channel conditions} \\ \mathcal{N}(\rho_L) = \mathrm{tr}_{\mathrm{env}}\left\{U_{\mathcal{N}}\rho_L U_{\mathcal{N}}^{\dagger}\right\} & & \widehat{\mathcal{N}}(\rho_L) = \mathrm{tr}_{\mathrm{sys}}\left\{U_{\mathcal{N}}\rho_L U_{\mathcal{N}}^{\dagger}\right\} \\ \updownarrow & & \updownarrow \\ \text{Heisenberg picture QEC conditions} & \leftrightarrow & ??? \\ \mathcal{N}^{\dagger}(\rho_L) = \mathrm{tr}_{\mathrm{env}}\left\{U_{\mathcal{N}}^{\dagger}\rho_L U_{\mathcal{N}}\right\} & & \widehat{\mathcal{N}}^{\dagger}(\rho_L) = \mathrm{tr}_{\mathrm{sys}}\left\{U_{\mathcal{N}}^{\dagger}\rho_L U_{\mathcal{N}}\right\} \end{array} \tag{43}$$

0.5 Cooking with Pauli: four-qubit code as a $[[4,1,2]]$ stabilizer code

Stabilizer group is an abelian Pauli subgroup S s.t. $-I \notin S$. **Stabilizer code** is the joint $+1$ eigenspace of a stabilizer group.
Stabilizers for $[[4,1,2]]$ code are

$$S = \langle \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \rangle \equiv \langle M_1, M_2, M_3\rangle = \left\{ M_1^a M_2^b M_3^c \,|\, a,b,c \in \mathbb{Z}_2 \right\} \tag{44}$$

Advantages of stabilizer codes

1. Efficient presentation in terms of stabilizer generators $\{M_j\,|\,, j = 1, \cdots, r\}$ instead of codewords
2. Syndromes obtained for free: stabilizer generators are check operators
3. Detectable/undetectable errors determined simply from check operators
4. General idea: works for bosons, fermions, modular qudits, Galois qudits, molecules

Stabilizer codes partition Pauli strings P into three different subsets:

Pauli P	Description	$\Pi P \Pi =$	Rel-n to stab. gen's
Stabilizers	Act trivially within codespace	Π	$S = \prod_{j=1}^{n-k} M_j^{p_j}$
Detectable errors	Map codespace into error spaces	0	$EM_j = -M_jE$ for some j
Logical Paulis	Act nontrivially within codespace	L	$LM_j = M_jL$ for all j

1. Detectable errors
   - All errors anti-commute with at least one stabilizer generator. For example, for $E = \begin{smallmatrix} I & Z \\ I & I \end{smallmatrix}$:
     $$\begin{smallmatrix} I & Z \\ I & I \end{smallmatrix}\begin{smallmatrix} X & X \\ X & X \end{smallmatrix} = -\begin{smallmatrix} X & X \\ X & X \end{smallmatrix}\begin{smallmatrix} I & Z \\ I & I \end{smallmatrix} \tag{45}$$
     This error maps code into error space spanned by $\{|0_Z\rangle, |1_Z\rangle\}$.
   - We can associate one E for each of the 8 available error spaces:

E	p	q	r	$\lvert 0_E\rangle$	$\lvert 1_E\rangle$	Previous lecture
$\begin{smallmatrix} I & I \\ I & I \end{smallmatrix}$	$+$	$+$	$+$	$\left\lvert\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right\rangle + \left\lvert\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right\rangle$	$\left\lvert\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right\rangle + \left\lvert\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\right\rangle$	$\lvert 0_L\rangle, \lvert 1_L\rangle$
$\begin{smallmatrix} I & Z \\ I & I \end{smallmatrix}$	$-$	$+$	$+$	$\left\lvert\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right\rangle - \left\lvert\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right\rangle$	$\left\lvert\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right\rangle - \left\lvert\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\right\rangle$	$\lvert 0_Z\rangle, \lvert 1_Z\rangle$
$\begin{smallmatrix} I & I \\ X & I \end{smallmatrix}$	$+$	$-$	$+$	$\left\lvert\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\right\rangle + \left\lvert\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right\rangle$	$\left\lvert\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right\rangle + \left\lvert\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right\rangle$	$\lvert 0_X\rangle, \lvert 1_X\rangle$
$\begin{smallmatrix} I & I \\ I & X \end{smallmatrix}$	$+$	$+$	$-$	$\left\lvert\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right\rangle + \left\lvert\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right\rangle$	$\left\lvert\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right\rangle + \left\lvert\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right\rangle$	
$\begin{smallmatrix} I & Z \\ X & I \end{smallmatrix}$	$-$	$-$	$+$	$\left\lvert\begin{smallmatrix} 0 & 0 \\ 1 & 0 \end{smallmatrix}\right\rangle - \left\lvert\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right\rangle$	$\left\lvert\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right\rangle - \left\lvert\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right\rangle$	$\lvert 0_Y\rangle, \lvert 1_Y\rangle$
$\begin{smallmatrix} I & Z \\ I & X \end{smallmatrix}$	$-$	$+$	$-$	$\left\lvert\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right\rangle - \left\lvert\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right\rangle$	$\left\lvert\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right\rangle - \left\lvert\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right\rangle$	
$\begin{smallmatrix} I & I \\ X & X \end{smallmatrix}$	$+$	$-$	$-$	$\left\lvert\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right\rangle + \left\lvert\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right\rangle$	$\left\lvert\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right\rangle + \left\lvert\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right\rangle$	
$\begin{smallmatrix} I & Z \\ X & X \end{smallmatrix}$	$-$	$-$	$-$	$\left\lvert\begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix}\right\rangle - \left\lvert\begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix}\right\rangle$	$\left\lvert\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right\rangle - \left\lvert\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right\rangle$	

   - Error spaces are called the **Pauli frames** of the original code.
     - They are themselves codespaces of a $[[4,1,2]]$ stabilizer code with stabilizer set
       $$\mathsf{S}_{pqr} = \langle p \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, q \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, r \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \rangle \tag{46}$$
     - For codes that can correct errors (NOT THIS ONE), if we track fast enough via rounds of QEC, we don't need to apply corrections every time.

2. Logical Paulis
   - For $[[4,1,2]]$ code, the four different logical Paulis can be realized by
     $$L \in \{\overline{I}, \overline{X}, \overline{Z}, \overline{Y}\} = \{\begin{smallmatrix} I & I \\ I & I \end{smallmatrix}, \begin{smallmatrix} X & I \\ X & I \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix}, \begin{smallmatrix} Y & Z \\ X & I \end{smallmatrix}\} \tag{47}$$
     Notice these are the same Pauli logicals for all error spaces — Pauli logicals are **error transparent**.
   - Logicals times stabilizers are just as good sets of logicals. There are eight different ways to implement logical-Z, logical-Y, and logical-I, e.g., four of them are
     $$\overline{I} \cong \begin{smallmatrix} I & I \\ I & I \end{smallmatrix} \cong \begin{smallmatrix} X & X \\ X & X \end{smallmatrix} \cong \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix} \cong \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \tag{48}$$
     $$\overline{X} \cong \begin{smallmatrix} X & I \\ X & I \end{smallmatrix} \cong \begin{smallmatrix} I & X \\ I & X \end{smallmatrix} \cong \begin{smallmatrix} Y & I \\ Y & I \end{smallmatrix} \cong \begin{smallmatrix} X & Z \\ X & Z \end{smallmatrix} \tag{49}$$
     $$\overline{Y} \cong \begin{smallmatrix} Y & Z \\ X & I \end{smallmatrix} \cong \begin{smallmatrix} Z & Y \\ I & X \end{smallmatrix} \cong \begin{smallmatrix} X & Z \\ Y & I \end{smallmatrix} \cong \begin{smallmatrix} Y & I \\ X & Z \end{smallmatrix} \tag{50}$$
     $$\overline{Z} \cong \begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix} \cong \begin{smallmatrix} Y & Y \\ X & X \end{smallmatrix} \cong \begin{smallmatrix} I & Z \\ Z & I \end{smallmatrix} \cong \begin{smallmatrix} Z & I \\ I & Z \end{smallmatrix} \tag{51}$$
   - Given fixed set of logical Paulis L and stabilizers $S \in \mathsf{S}$, any codespace preserving Pauli can be decomposed *uniquely* as
     $$\text{codespace-preserving Pauli} = L \cdot S \tag{52}$$
   - Code-preserving Paulis make up the **normalizer**
     $$\mathsf{N}(\mathsf{S}) \equiv \mathsf{N}_{\mathsf{P}_n}(\mathsf{S}) = \text{everything in } \mathsf{P}_n \text{ that commutes with everything in } \mathsf{S} \tag{53}$$
     while logical Pauli representatives L make up the quotient group $\mathsf{N}(\mathsf{S})/\langle i, \mathsf{S}\rangle$:

	$[[4,1,2]]$ code	Size	Stabilizer	Size
stabilizer group	$\langle \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \rangle$	$2^3 = 8$	S	2^{n-k}
code-preserving Paulis $L \cdot S$	$\langle i, \begin{smallmatrix} X & I \\ X & I \end{smallmatrix}, \begin{smallmatrix} I & X \\ I & X \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ Z & Z \end{smallmatrix}, \begin{smallmatrix} Z & I \\ I & Z \end{smallmatrix} \rangle$	$4 \cdot 2^5 = 128$	$\mathsf{N}(\mathsf{S})$	$4 \cdot 2^{n+k}$
non-trivial logical Paulis	a large set		$\mathsf{N}(\mathsf{S}) - \langle i, \mathsf{S}\rangle$	
logical Paulis modulo $\langle i, \mathsf{S}\rangle$	$\{\begin{smallmatrix} I & I \\ I & I \end{smallmatrix}, \begin{smallmatrix} X & I \\ X & I \end{smallmatrix}, \begin{smallmatrix} Y & Z \\ X & I \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix}\}$	$\frac{4 \cdot 2^5}{4 \cdot 2^3} = 4$	$\mathsf{N}(\mathsf{S})/\langle i, \mathsf{S}\rangle$	4^k

0.6 Decoding stabilizer codes

Combining notion of Pauli frame + logical actions + stabilizer equivalence, we have that any error AND any recovery operator can be decomposed (up to phase) as

$$\text{any Pauli} = E \cdot L \cdot S \tag{54}$$

where E is designated error for that error space, L is a designated logical, and S is a stabilizer. For example, our Pauli-Y error from before can be expressed as a product of the E's and L's we picked above, multiplied by a compensating S:

$$\begin{smallmatrix} I & I \\ Y & I \end{smallmatrix} = \begin{smallmatrix} I & Z \\ X & I \end{smallmatrix} \cdot \begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix} \cdot \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix} \tag{55}$$

Counting the three different components of a Pauli string:

$$\text{number of representative } E\text{'s} = 2^{n-k} = 8 = 2^{4-1} \tag{56}$$

$$\text{number of representative } L\text{'s} = 4^{k} = 4 \tag{57}$$

$$\text{number of } S\text{'s} = 2^{n-k} = 8 \tag{58}$$

$$\text{total} = 4^{n} = |\mathsf{P}_n| / \langle i \rangle = 4^4 \ \checkmark \tag{59}$$

There are two types of decoding problems for quantum codes, one that is based on classical codes (which are non-degenerate) and one which takes into account degeneracy.

1. ML decoding: given an E, FIND L that corrects the most likely error:

$$(L \cdot S)_\star = \arg\max_{L,S} \Pr(L, S \,|\, E) \tag{60}$$

   (a) Given E, there are $4^k \times 2^{n-k} = 2^{n+k} = 2^5 = 32$ combinations of $L \cdot S$ to choose from.

   (b) There are 2^{n-k} error spaces, and all exact codes have $k \leq n/2$, so we will run out of space writing down a **look up table** of recoveries for large enough n. In such cases, we have to compute on the fly.

   (c) Similar to decoding classical codes, ML decoding is NP-complete. Many cryptographic protocols are based on decoding classical ECCs.

2. Degenerate ML decoding: we don't need to care about stabilizer action since it leaves codespace invariant, so instead FIND L that corrects most likely representative logical operator (i.e., **error class**):

$$L_\star = \arg\max_{L} \sum_{S} \Pr(L, S \,|\, E) \tag{61}$$

   (a) There are cases where dominant error $(L \cdot S)_\star$ is in one class, but $L_\star$ is in another due to accumulation of contributions from stabilizers. This problem is #P-complete.

   (b) **Stat-mech mapping:** for certain stabilizer codes, the sum above mapped to partition function of statistical mechanical spin glass model.

0.7 Erasure errors + cleaning lemma

All logicals L are supported on at least d qubits, and, for the four-qubit code, any logical Pauli group is supported on 3 qubits. This correlates with distance: we can correct $d - 1 = 1$ erasures, meaning that logical information has to be storable in 3 qubits only:

$$\mathrm{tr}_{\substack{\checkmark\times\\ \times\times}} \rho = \mathrm{tr}_{\substack{\times\times\\ \checkmark\times}} \rho = \mathrm{tr}_{\substack{\times\checkmark\\ \times\times}} \rho = \mathrm{tr}_{\substack{\times\times\\ \times\checkmark}} \rho = \frac{1}{2} I \tag{62}$$

In general, an $[[n, k, d]]$ code can correct (known) erasures on $d - 1$ qubits, and tracing out all but any $d - 1$ qubits yields a state that is independent of the logical information.

Since the lowest-weight logical has weight equal to the distance d, this means that logical operators cannot be supported on any qubit subset $\mathtt{M}$ containing at most $d - 1$ qubits. But there is more: logical operators can be moved or **cleaned** via application of stabilizers,

$$L \cdot S \to L \cdot S \cdot S' \,, \tag{63}$$

and the **cleaning lemma** says that, for every subset $\mathtt{M}$ of qubits,

- Either you "can clean `M`": all logicals can be chosen to act outside of `M`
- Or you "cannot clean `M`": $\exists$ a logical acting entirely within `M`

Moreover, $g(\mathtt{M}) + g(\mathtt{M}^{\perp}) = 2k$, where g is the number of logical-X, Z Paulis supported on region `M`.. For example,

1. $|\mathtt{M}| = 1$ "all clean": all logical-X, Z can be "cleaned" (via stabilizers) to act outside of `M` $(0 + 2 = 2)$
2. $|\mathtt{M}| = 2$ "a little dirty": logical-X Pauli supported on both regions $(1 + 1 = 2)$

$$\text{vertical} \left\langle \begin{smallmatrix} X & I \\ X & I \end{smallmatrix} \right\rangle \tag{64}$$
$$\text{horizontal} \left\langle \begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix} \right\rangle \tag{65}$$
$$\text{diagonal} \left\langle \begin{smallmatrix} Z & I \\ I & Z \end{smallmatrix} \right\rangle \tag{66}$$

3. $|\mathtt{M}| = 3$ "filthy": all logicals can be converted via stabilizers to act inside of `M` $(2 + 0 = 2)$

0.8 Four-qubit expansion pack: the $[[4, 2, 2]]$ code

Another logical qubit can be added to the four-qubit code to yield a code with stabilizer group

$$\mathsf{S} = \left\langle \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix} \right\rangle \tag{67}$$

logical states

$$\begin{array}{ll} |00_L\rangle \propto \left| \begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix} \right\rangle + \left| \begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix} \right\rangle & |01_L\rangle \propto \left| \begin{smallmatrix} 1 & 1 \\ 0 & 0 \end{smallmatrix} \right\rangle + \left| \begin{smallmatrix} 0 & 0 \\ 1 & 1 \end{smallmatrix} \right\rangle \\ |10_L\rangle \propto \left| \begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix} \right\rangle + \left| \begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix} \right\rangle & |11_L\rangle \propto \left| \begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix} \right\rangle + \left| \begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix} \right\rangle \end{array} . \tag{68}$$

and logical Pauli group represented by

$$\Big\{ \underbrace{\begin{smallmatrix} X & I \\ X & I \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix}}_{\text{qubit I}}, \underbrace{\begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}}_{\text{qubit II}} \Big\} \tag{69}$$

0.9 Subsystem stabilizer codes

Let us *not* use the second qubit for storage, but as a tunable knob or "gauge" degree of freedom. The two-qubit (initially fully logical) space,

$$\mathbb{C}^{2^4} = \mathbb{C}^4 \oplus \mathbb{C}^{12} = \left(\mathbb{C}^2 \otimes \mathbb{C}^2\right) \oplus \mathbb{C}^{12} , \tag{70}$$

is split into a logical qubit for the first $\mathbb{C}^2$-factor/subsystem a "gauge qubit" for the second $\mathbb{C}^2$-subsystem. This is an example of a subsystem code.

We need another (this time, non-Abelian) group G to keep track of which "gauge" qubits we have picked. A general **subsystem code** is defined by pair of groups (S, G) with **stabilizer group** S and **gauge group** G satisfying

$$\langle i, \mathsf{S} \rangle \subseteq \mathsf{G} \subseteq \mathsf{N}(\mathsf{S}) , \tag{71}$$

Only subgroups of code-preserving Paulis are allowed because those preserve the joint $+1$ eigenspace of the stabilizer group. Conversely, one can say that the **center** of G — the set of elements of G which commute with all elements of G — has to be $\mathsf{Z}(\mathsf{G}) = \langle i, S \rangle$. The two edge cases are:

1. $\mathsf{G} = \langle i, \mathsf{S} \rangle$: no qubits are gauge –> stabilizer code
2. $\mathsf{G} = \mathsf{N}(\mathsf{S})$: all qubits are gauge –> no logical subspace

Qubit subsystem stabilizer codes are denoted as $[[n, k, g, d]]$, where g is the number of gauge qubits. The similarity to gauging in electromagnetism is only conceptual. In the latter, electric and magnetic potentials can be changed via gauge transformations without affecting the physically observable fields. In the former, the gauge qubits can be manipulated and measured without affect the logical information.

1. The process of converting logical qubits of a stabilizer code into gauge qubits is called **gauging out**, where one adds some Pauli subgroup $\mathsf{F} \subseteq \mathsf{N}(\mathsf{S})$ to the initially trivial gauge group. The stabilizer code is converted into a subsystem code as

$$\mathsf{S} \to \mathsf{Z}\left(\langle i, \mathsf{S}, \mathsf{F} \rangle\right) \tag{72}$$
$$\langle i, \mathsf{S} \rangle \to \langle i, \mathsf{S}, \mathsf{F} \rangle \tag{73}$$

- For example, starting with the $[[4,2,2]]$ code and gauging out the second qubit proceeds by adding the second qubit's logical Pauli group, $\mathsf{F} = \langle \begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix} \rangle$, into the gauge group:

$$\langle \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix} \rangle \rightarrow \langle i, \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix} \rangle \tag{74}$$

$$\langle i, \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix} \rangle \rightarrow \Big\langle i, \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix}, \underbrace{\begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}}_{\text{qubit II}} \Big\rangle = \Big\langle i, \begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \Big\rangle \tag{75}$$

  This converts the $[[4,2,2]]$ code into the $[[4,1,1,2]]$ subsystem code.
- Since the gauge group contains i by convention, the same code is obtained by starting with $\mathsf{S}_{pq} = \langle p \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, q \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix} \rangle$ for any $p, q = \pm$.

2. The process of "setting and forgetting" the gauge qubits is called **gauge fixing**. This allows us to switch treat different stabilizer codes as being part of a single gauge code.
   - Gauge fixing second qubit to $|0_L\rangle$ (and forgetting it) gets back to $[[4,1,2]]$ code with stabilizer group $\langle \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \rangle$.
   - Gauge fixing second qubit to $|+_L\rangle$ (and forgetting it) means we have picked $\mathsf{F} = \langle \begin{smallmatrix} X & X \\ I & I \end{smallmatrix} \rangle$ yields "dual" $[[4,1,2]]$ code with stabilizer group $\langle \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix}, \begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ X & X \end{smallmatrix} \rangle$.

Advantages of subsystem codes:

1. Measuring lower-weight operators can yield error syndromes. For example, one can measure $\{ \begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ X & X \end{smallmatrix} \}$ and infer $\begin{smallmatrix} X & X \\ X & X \end{smallmatrix}$:

$$\boxed{\begin{smallmatrix} X & X \\ I & I \end{smallmatrix} = -1} \quad + \quad \boxed{\begin{smallmatrix} I & I \\ X & X \end{smallmatrix} = +1} \quad = \quad \boxed{\begin{smallmatrix} X & X \\ X & X \end{smallmatrix} = -1} \quad \checkmark \tag{76}$$

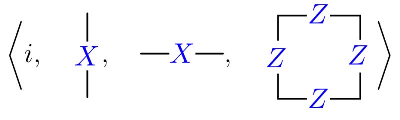
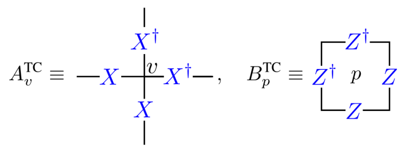
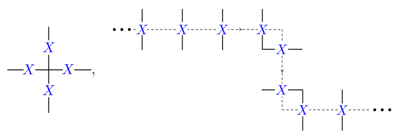
   Extending this class of codes yields the Bacon-Shor codes, e.g.,

$$\mathsf{S}_{[[9,1,4,3]]} = \Big\langle \begin{smallmatrix} X & X & I \\ X & X & I \\ X & X & I \end{smallmatrix}, \begin{smallmatrix} I & X & X \\ I & X & X \\ I & X & X \end{smallmatrix}, \begin{smallmatrix} Z & Z & Z \\ Z & Z & Z \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ Z & Z & Z \\ Z & Z & Z \end{smallmatrix} \Big\rangle \tag{77}$$

$$\mathsf{G}_{[[9,1,4,3]]} = \Big\langle i, \mathsf{S}, \begin{smallmatrix} X & X & I \\ I & I & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ X & X & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & X & X \\ I & I & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ I & X & X \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} Z & I & I \\ Z & I & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & Z & I \\ I & Z & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ Z & I & I \\ Z & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ I & Z & I \\ I & Z & I \end{smallmatrix} \Big\rangle \tag{78}$$

   Products of weight-two gauge operators can be used to obtain stabilizer generators.

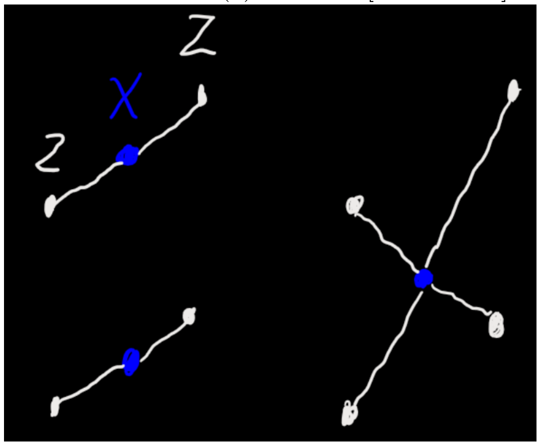
2. Subsystem codes realize more phases of matter than stabilizer codes, e.g.,

	planar surface code [2112.11394]	$\mathbb{Z}_2^{(0)}$ subsystem code [2211.03798]
G		
S		
Hamiltonian	$-\sum_{S \in \mathsf{S}} S$	$-\sum_{G \in \mathsf{G}} J_G G \quad \text{s.t.} \quad J_G \in \mathbb{R}$
Realizes	$\mathbb{Z}_2$ topological order	$\mathbb{Z}_2$ gauge theory (w/$XXXX$ constraint)

3. gauge fixing can allow you to switch between different codes, and many gadgets can be understood as subsystem codes
   (a) Recall Eastin-Knill: no cont-parameter family of gates: for distance ≥ 2, assume BWOC there is a transversal generator $E = \sum_{j=1}^{n} E_j$ that does a logical gate, then

$$PEP \neq cP \rightarrow PE_jP \neq c_jP \rightarrow \text{no logical rotation} \tag{79}$$

4. **MBQC** is a computing paradigm in which one "substitutes time for space", measuring a resource state called a **cluster state** layer by layer in order perform a computation. This resource state can be thought of as a code state of a subsystem code. CSS codes can be converted into their corresponding cluster states via (1) clusterization and (2) foliation [1607.02579].



   (a) Z checks of $[[4,1,2]]$ code with $\mathsf{S} = \langle \begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ Z & Z \end{smallmatrix}, \begin{smallmatrix} X & X \\ X & X \end{smallmatrix} \rangle$ can be **clusterized** into a larger 6-qubit cluster state by relating each Z-type generator to a new ancilla qubit and making connection between said ancilla and all data qubits which support the Z-type generator. There are two Z-type generators, $\begin{smallmatrix} Z & Z \\ I & I \end{smallmatrix}$ and $\begin{smallmatrix} I & I \\ Z & Z \end{smallmatrix}$, so we add two ancilla qubits.
   Then, we construct a cluster-state stabilizer group on the graph formed by the above connections. Each node j (either data or ancilla) admits a generator $X \prod_{j \in N(v)} Z_j$, where $N(v)$ are all neighbors of v. There are six nodes, so we have six generators:

$$\mathsf{S}_{\text{primal}} = \langle \begin{smallmatrix} X & Z & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} Z & X & Z \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & Z & X \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ X & Z & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ Z & X & Z \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ I & Z & X \end{smallmatrix} \rangle \tag{80}$$

   The clusterized code can be thought of as a subsystem code with extra qubits being the gauge qubits

$$\mathsf{G} = \langle i, \mathsf{S}_{\text{primal}}, \begin{smallmatrix} I & X & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ I & X & I \end{smallmatrix} \rangle \tag{81}$$

   After measuring the gauge qubits in X, the resulting state is in either the code or an error space of the four-qubit code.

      i. Once a state is initialized in the cluster state of $\mathsf{S}_{\text{primal}}$, outcomes of gauge-qubit measurements yield Z-type error syndromes,

$$\boxed{\begin{smallmatrix} I & X & I \\ I & I & I \end{smallmatrix} = -1 \text{ (by msmnt.)}} \quad + \quad \boxed{\begin{smallmatrix} Z & X & Z \\ I & I & I \end{smallmatrix} = +1 \text{ (by assumption)}} \quad = \quad \boxed{\begin{smallmatrix} Z & I & Z \\ I & I & I \end{smallmatrix} = -1} \tag{82}$$

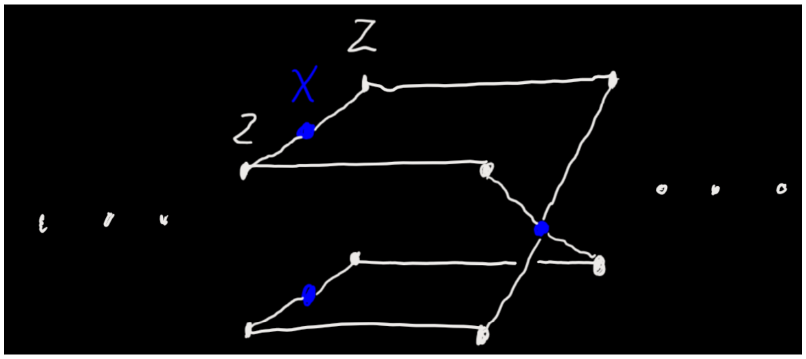
      ii. X-type generators are products of stabilizer generators which have Z on ancilla qubits; they are all $+1$ by assumption. For example,

$$\boxed{\begin{smallmatrix} X & Z & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & Z & X \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ X & Z & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ I & Z & X \end{smallmatrix} = +1 \text{ (by assumption)}} \quad \rightarrow \quad \boxed{\begin{smallmatrix} X & I & X \\ X & I & X \end{smallmatrix} = +1} \tag{83}$$

   (b) X checks clusterized in the same way, and measuring the gauge qubits in this case yields X-type error syndromes.

$$\mathsf{S}_{\text{dual}} = \left\langle \begin{smallmatrix} Z & & Z \\ & X & \\ Z & & Z \end{smallmatrix}, \begin{smallmatrix} I & & I \\ & Z & \\ I & & X \end{smallmatrix}, \begin{smallmatrix} I & & I \\ & Z & \\ X & & I \end{smallmatrix}, \begin{smallmatrix} X & & I \\ & Z & \\ I & & I \end{smallmatrix}, \begin{smallmatrix} I & & X \\ & Z & \\ I & & I \end{smallmatrix} \right\rangle \tag{84}$$

   (c) The codes can then be combined ("foliated") via links between physical qubits, yielding a cluster state that trades time dimension for a spatial one. Stabilizer checks are measured as layers are traversed, and error correction is done using both the power of the underlying CSS code + taking into account single-X msmnt errors.



Summary of subsystem code groups:

	$[[n,k,g,d]] = [[4,1,1,2]]$ subsystem	Size	Subsystem	Size
gauge group	$\langle i, \begin{smallmatrix}X&X\\I&I\end{smallmatrix}, \begin{smallmatrix}I&I\\X&X\end{smallmatrix}, \begin{smallmatrix}Z&I\\Z&I\end{smallmatrix}, \begin{smallmatrix}I&Z\\I&Z\end{smallmatrix}\rangle$	64	G	$4\cdot 2^{n-k+g}$
stabilizer group	$\langle \begin{smallmatrix}X&X\\X&X\end{smallmatrix}, \begin{smallmatrix}Z&Z\\Z&Z\end{smallmatrix}\rangle$	4	S	2^{n-k-g}
code-preserving	$\langle i, \begin{smallmatrix}X&I\\X&I\end{smallmatrix}, \begin{smallmatrix}X&X\\I&I\end{smallmatrix}, \begin{smallmatrix}X&I\\I&X\end{smallmatrix}, \begin{smallmatrix}Z&I\\Z&I\end{smallmatrix}, \begin{smallmatrix}Z&Z\\I&I\end{smallmatrix}, \begin{smallmatrix}Z&I\\I&Z\end{smallmatrix}\rangle$	256	$\mathsf{N(S)}$	$4\cdot 2^{n+k+g}$
logical Paulis	$\langle \begin{smallmatrix}X&I\\X&I\end{smallmatrix}, \begin{smallmatrix}Z&Z\\I&I\end{smallmatrix}\rangle$ (up to G)	4	$\mathsf{N(S)/G}$	4^k
gauge Paulis	$\langle \begin{smallmatrix}X&X\\I&I\end{smallmatrix}, \begin{smallmatrix}Z&I\\Z&I\end{smallmatrix}\rangle$	4	$\frac{\mathsf{N(S)}}{\mathsf{N(S)/G}\times\langle i,\mathsf{S}\rangle} = \mathsf{G}/\langle i,\mathsf{S}\rangle$	4^g
preserve gauge	$\langle i, \begin{smallmatrix}X&X\\X&X\end{smallmatrix}, \begin{smallmatrix}Z&Z\\Z&Z\end{smallmatrix}, \begin{smallmatrix}X&I\\X&I\end{smallmatrix}, \begin{smallmatrix}Z&Z\\I&I\end{smallmatrix}\rangle$	64	$\mathsf{N(G)}$	$4\cdot 2^{n+k-g}$
"bare" logical Paulis	$\langle \begin{smallmatrix}X&I\\X&I\end{smallmatrix}, \begin{smallmatrix}Z&Z\\I&I\end{smallmatrix}\rangle$ (up to S)	4	$\mathsf{N(G)}/\langle i,\mathsf{S}\rangle$	4^k

0.10 Stabilizer code switching

Code switching can be done via unitary transformations, e.g., via n-qubit Clifford group

$$\mathsf{C}_n = \left\{U \in \mathsf{U}(2^n)\,|\,UP_nU^\dagger = P_n\right\} \tag{85}$$

There are also **non-unitary** ways to switch codes while maintaining logical information. For example, as we have already seen, gauging out converts stabilizer codes into subsystem codes but converting some logical qubits to gauge qubits.

Code switching between stabilizer codes proceeds as follows: given a stabilizer group (i.e., abelian group with no -1) $\mathsf{F} \in \mathsf{P}_n$,

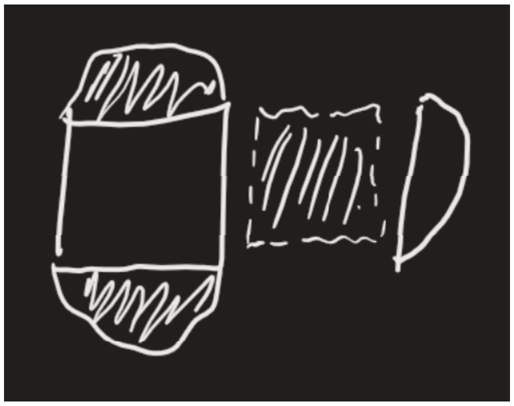
$$\boxed{\mathsf{S} \to \mathsf{N}_{\langle\mathsf{S},\mathsf{F}\rangle}(\mathsf{F})\ .} \tag{86}$$

Generators of F should generally be understood as having been measured, thereby restricting the initial stabilizer code to the subspace corresponding to the eigenvalues obtained from the measurements. If a -1 outcome is obtained for some element F, then $-F$ goes into F.

1. Starting with the four qubit code with $\mathsf{S} = \langle \begin{smallmatrix}X&X\\X&X\end{smallmatrix}, \begin{smallmatrix}Z&I\\Z&I\end{smallmatrix}, \begin{smallmatrix}I&Z\\I&Z\end{smallmatrix}\rangle$ and measuring $+1$ eigenvalue for generator in $\mathsf{F} = \langle \begin{smallmatrix}X&X\\I&I\end{smallmatrix}\rangle$ yields
$$\mathsf{S} \to \mathsf{N}_{\langle\mathsf{S},\mathsf{F}\rangle}(\mathsf{F}) = \langle \begin{smallmatrix}X&X\\I&I\end{smallmatrix}, \begin{smallmatrix}I&I\\X&X\end{smallmatrix}, \begin{smallmatrix}Z&Z\\Z&Z\end{smallmatrix}\rangle\ , \tag{87}$$
the "dual" four-qubit code from before. This is a case when two stabilizer codes that are switchable into each other belong to the same subsystem code whose gauge qubits are set to different states and then forgotten [1810.10037].

$$\begin{tikzcd}
 & \mathsf{G} = \langle i, \begin{smallmatrix}X&X\\I&I\end{smallmatrix}, \begin{smallmatrix}I&I\\X&X\end{smallmatrix}, \begin{smallmatrix}Z&I\\Z&I\end{smallmatrix}, \begin{smallmatrix}I&Z\\I&Z\end{smallmatrix}\rangle \arrow[dl, "\text{gauge fixing}"'] \arrow[dr, "\text{gauge fixing}"] & \\
\mathsf{S} = \langle \begin{smallmatrix}X&X\\X&X\end{smallmatrix}, \begin{smallmatrix}Z&I\\Z&I\end{smallmatrix}, \begin{smallmatrix}I&Z\\I&Z\end{smallmatrix}\rangle \arrow[rr, leftrightarrow, "\text{code switching}"'] & & \mathsf{S} = \langle \begin{smallmatrix}X&X\\I&I\end{smallmatrix}, \begin{smallmatrix}I&I\\X&X\end{smallmatrix}, \begin{smallmatrix}Z&Z\\Z&Z\end{smallmatrix}\rangle
\end{tikzcd} \tag{88}$$

2. An example of code switching is **lattice surgery**, which combines codes in a way that yields low-overhead logical measurements. Consider the four-qubit $[[4,1,2]]$ code and a trivial two qubit $[[2,1,1]]$ code:



Stabilizers of each code are

$$\mathsf{S}_1 = \langle \begin{smallmatrix} Z & Z & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ Z & Z & I \end{smallmatrix}, \begin{smallmatrix} X & X & I \\ X & X & I \end{smallmatrix} \rangle \qquad \mathsf{S}_2 = \langle \begin{smallmatrix} I & I & X \\ I & I & X \end{smallmatrix} \rangle \tag{89}$$

Codewords are

$$|\overline{00}\rangle \propto |\begin{smallmatrix} 0&0&0 \\ 0&0&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&0 \\ 1&1&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&1 \\ 0&0&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&1 \\ 1&1&1 \end{smallmatrix}\rangle \tag{90}$$
$$|\overline{01}\rangle \propto |\begin{smallmatrix} 0&0&0 \\ 0&0&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&0 \\ 1&1&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&1 \\ 0&0&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&1 \\ 1&1&0 \end{smallmatrix}\rangle \tag{91}$$
$$|\overline{10}\rangle \propto |\begin{smallmatrix} 1&1&0 \\ 0&0&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&0 \\ 1&1&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&1 \\ 0&0&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&1 \\ 1&1&1 \end{smallmatrix}\rangle \tag{92}$$
$$|\overline{11}\rangle \propto |\begin{smallmatrix} 1&1&0 \\ 0&0&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&0 \\ 1&1&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&1 \\ 0&0&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&1 \\ 1&1&0 \end{smallmatrix}\rangle \tag{93}$$

Let us measure $F = \begin{smallmatrix} I & Z & Z \\ I & Z & Z \end{smallmatrix}$. Assuming we get $+1$, we can merge the two codes using $\mathsf{F} = \langle \begin{smallmatrix} I & Z & Z \\ I & Z & Z \end{smallmatrix} \rangle$,

$$\mathsf{S} \to \mathsf{N}_{\langle \mathsf{S},\mathsf{F} \rangle}(\mathsf{F}) = \langle \begin{smallmatrix} Z & Z & I \\ I & I & I \end{smallmatrix}, \begin{smallmatrix} I & I & I \\ Z & Z & I \end{smallmatrix}, \begin{smallmatrix} X & X & I \\ X & X & I \end{smallmatrix}, \begin{smallmatrix} I & I & X \\ I & I & X \end{smallmatrix}, \begin{smallmatrix} I & Z & Z \\ I & Z & Z \end{smallmatrix} \rangle \tag{94}$$

with codewords

$$(I + \begin{smallmatrix} I & Z & Z \\ I & Z & Z \end{smallmatrix})\,|\overline{00}\rangle \propto |\begin{smallmatrix} 0&0&0 \\ 0&0&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&0 \\ 1&1&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&1 \\ 0&0&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&1 \\ 1&1&1 \end{smallmatrix}\rangle \tag{95}$$
$$(I + \begin{smallmatrix} I & Z & Z \\ I & Z & Z \end{smallmatrix})\,|\overline{01}\rangle = 0 \tag{96}$$
$$(I + \begin{smallmatrix} I & Z & Z \\ I & Z & Z \end{smallmatrix})\,|\overline{10}\rangle = 0 \tag{97}$$
$$(I + \begin{smallmatrix} I & Z & Z \\ I & Z & Z \end{smallmatrix})\,|\overline{11}\rangle \propto |\begin{smallmatrix} 1&1&0 \\ 0&0&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&0 \\ 1&1&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1&1 \\ 0&0&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0&1 \\ 1&1&0 \end{smallmatrix}\rangle \tag{98}$$

In other words, measuring F combined the two codes into a $[[6,1,2]]$ code while, at the same time, performing a logical $\overline{ZZ}$ measurement. Had we obtained the -1 outcome, we would have used the $I - \begin{smallmatrix} I & Z & Z \\ I & Z & Z \end{smallmatrix}$ projection and switched into a different $[[6,1,2]]$ code. This procedure is resource efficient for larger surface codes because one can perform a logical operation using only a 1D subset of a 2D qubit lattice.

3. **Floquet codes** are examples of code switching that allows one to obtain error syndromes by measuring weight-two operators, but without having to multiply many of them like in the case of Bacon-Show codes.

   (a) Start with stabilizer, $\mathsf{S}_1 = \langle \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \rangle$, meaning we are now in the span of $\{|\begin{smallmatrix} 0&0 \\ 0&0 \end{smallmatrix}\rangle, |\begin{smallmatrix} 1&0 \\ 1&0 \end{smallmatrix}\rangle, |\begin{smallmatrix} 0&1 \\ 0&1 \end{smallmatrix}\rangle, |\begin{smallmatrix} 1&1 \\ 1&1 \end{smallmatrix}\rangle\}$, i.e., trivial code. Measure

$$\mathsf{F}_1 = \langle \begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ X & X \end{smallmatrix} \rangle \tag{99}$$

   Assuming outcome to be $+1$ for both, this projects onto the protected codespace

$$(1 + \begin{smallmatrix} X & X \\ I & I \end{smallmatrix} + \begin{smallmatrix} I & I \\ X & X \end{smallmatrix} + \begin{smallmatrix} X & X \\ X & X \end{smallmatrix})\,|\begin{smallmatrix} 0&0 \\ 0&0 \end{smallmatrix}\rangle \text{ or } |\begin{smallmatrix} 1&1 \\ 1&1 \end{smallmatrix}\rangle \propto |\begin{smallmatrix} 0&0 \\ 0&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1 \\ 1&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&1 \\ 0&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&0 \\ 1&1 \end{smallmatrix}\rangle \tag{100}$$
$$(1 + \begin{smallmatrix} X & X \\ I & I \end{smallmatrix} + \begin{smallmatrix} I & I \\ X & X \end{smallmatrix} + \begin{smallmatrix} X & X \\ X & X \end{smallmatrix})\,|\begin{smallmatrix} 1&0 \\ 1&0 \end{smallmatrix}\rangle \text{ or } |\begin{smallmatrix} 0&1 \\ 0&1 \end{smallmatrix}\rangle \propto |\begin{smallmatrix} 1&0 \\ 1&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&1 \\ 0&1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0&1 \\ 1&0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1&0 \\ 0&1 \end{smallmatrix}\rangle \tag{101}$$

   with stabilizer group the subgroup of $\langle \mathsf{S}_1, \mathsf{F}_1 \rangle = \langle \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix}, \begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ X & X \end{smallmatrix} \rangle$ that commutes with F_1,

$$\mathsf{S}_1 \to \mathsf{S}_2 = \mathsf{N}_{\langle \mathsf{S}_1,\mathsf{F}_1 \rangle}(\mathsf{F}_1) = \langle \begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix} \rangle \tag{102}$$

   Other outcomes yields same stabilizer group, up to signs.

   (b) Now measure $\mathsf{F}_2 = \langle \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \rangle$. Assuming $+1$ for both outcomes, we are back to a four-qubit code:

$$\mathsf{S}_2 \to \mathsf{S}_3 = \mathsf{N}_{\langle \mathsf{S}_2,\mathsf{F}_2 \rangle}(\mathsf{F}_2) = \langle \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix}, \begin{smallmatrix} X & X \\ X & X \end{smallmatrix} \rangle \tag{103}$$

(c) Now measure $\mathsf{F}_3 = \langle \begin{smallmatrix} Y & Y \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ Y & Y \end{smallmatrix} \rangle$. Assuming $+1$ for both outcomes, this yields

$$\mathsf{S}_3 \to \mathsf{S}_4 = \mathsf{N}_{\langle \mathsf{S}_3, \mathsf{F}_3 \rangle}(\mathsf{F}_3) = \langle \begin{smallmatrix} Y & Y \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ Y & Y \end{smallmatrix}, \begin{smallmatrix} X & X \\ X & X \end{smallmatrix} \rangle \tag{104}$$

(d) A novel fact about Floquet codes is that they cannot be thought of as gauge fixing. When treated as a subsystem code with gauge group

$$\mathsf{G} = \langle \mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3 \rangle = \langle i, \begin{smallmatrix} X & X \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ X & X \end{smallmatrix}, \begin{smallmatrix} Y & Y \\ I & I \end{smallmatrix}, \begin{smallmatrix} I & I \\ Y & Y \end{smallmatrix}, \begin{smallmatrix} Z & I \\ Z & I \end{smallmatrix}, \begin{smallmatrix} I & Z \\ I & Z \end{smallmatrix} \rangle \tag{105}$$

its center is $Z(\mathsf{G}) = \langle i, \begin{smallmatrix} X & X \\ X & X \end{smallmatrix}, \begin{smallmatrix} Z & Z \\ Z & Z \end{smallmatrix} \rangle$, with the size of the stabilizer being

$$2^{n-k-g} = 2^{4-k-g} \equiv 2^2 \to k + g = 2 \tag{106}$$

Counting g and then inferring k, we see the gauge group is too large:

$$|\mathsf{G}/\langle i, \mathsf{S} \rangle| = \frac{4 \cdot 2^6}{4 \cdot 2^2} = 2^4 \equiv 4^g \to g = 2 \tag{107}$$

which means that $\boxed{k = 0}$, i.e., no logical qubits. In other words, Floquet code operations are not gauge fixing.

Summary of the dynamic procedures we learned about and their applications. References in gray were not covered in the above text.

	gauging out	gauge fixing	code switching
from/to using	stab. $\to$ gauge $\mathsf{F} \subseteq \mathsf{N}(\mathsf{S})$	gauge $\to$ gauge stab. group $\mathsf{F} \subset \mathsf{G}$	stab. $\to$ stab. stab. group $\mathsf{F} \subset \mathsf{P}_n$
S transforms as G transforms as	$\mathsf{S} \to \mathsf{Z}(\langle i, \mathsf{S}, \mathsf{F} \rangle)$ $\langle i, \mathsf{S} \rangle \to \langle i, \mathsf{S}, \mathsf{F} \rangle$	$\mathsf{S} \to \langle \mathsf{S}, \mathsf{F} \rangle$ $\mathsf{G} \to \mathsf{N}_{\mathsf{G}}(\mathsf{F})$	$\mathsf{S} \to \mathsf{N}_{\langle \mathsf{S}, \mathsf{F} \rangle}(\mathsf{F})$
MBQC		✓ [1607.02579]	✓ [2212.06775]
lattice surgery		✓ [1810.10037]	✓ [1810.10037]
Floquet codes		× [2107.02194]	✓ [2107.02194], [2307.10353]
anyon condensation			✓ [2212.00042]
chiral abelian top. phases	✓ [2211.03798]		

0.11 Fault tolerance

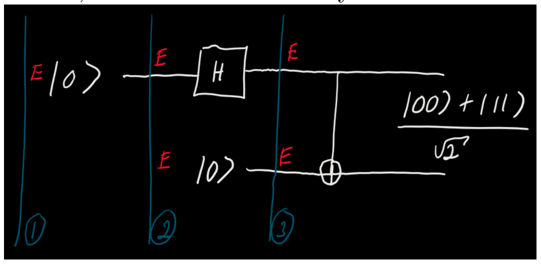
In a computational setting, one is also interested in performing operations, which include state initialization, gates, and measurements, on the stored information. The purpose of making fault-tolerant gadgets that implement said operations is to make sure errors do not spread too much so that interleaving rounds of decoding will be sufficient to quell the noise. As opposed to the cleaner assumption of noise only occuring during a noise channel, the assumption now is that there is noise everywhere during the performance of our operation.

Let us take a look at how to do fault tolerant state initization gadgets.

- For the four-qubit code, the logical-plus state is a direct product of Bell states,

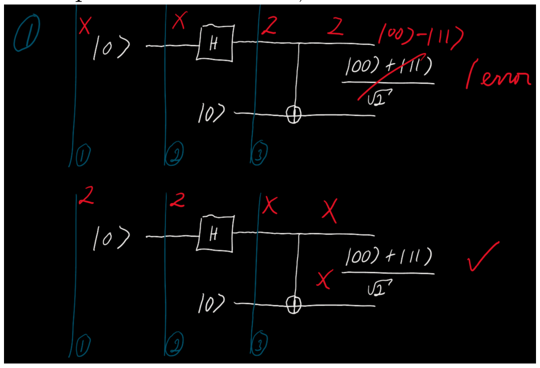
$$|+_L\rangle \propto |\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\rangle = (|\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\rangle + |\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\rangle)^{\otimes 2} \tag{108}$$

It turns out that a simple gadget making such a state is fault-tolerant if (1) we assume at most one fault occurs and (2) post-select on the no-error case. In other words, one fault causes at most one error in the circuit, which is detectable by the code.

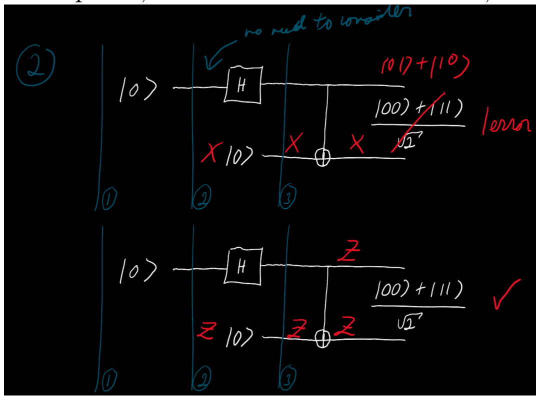


To check this, we need to identify all possible single fault locations and check whether faults cause an error in the output of the circuit.

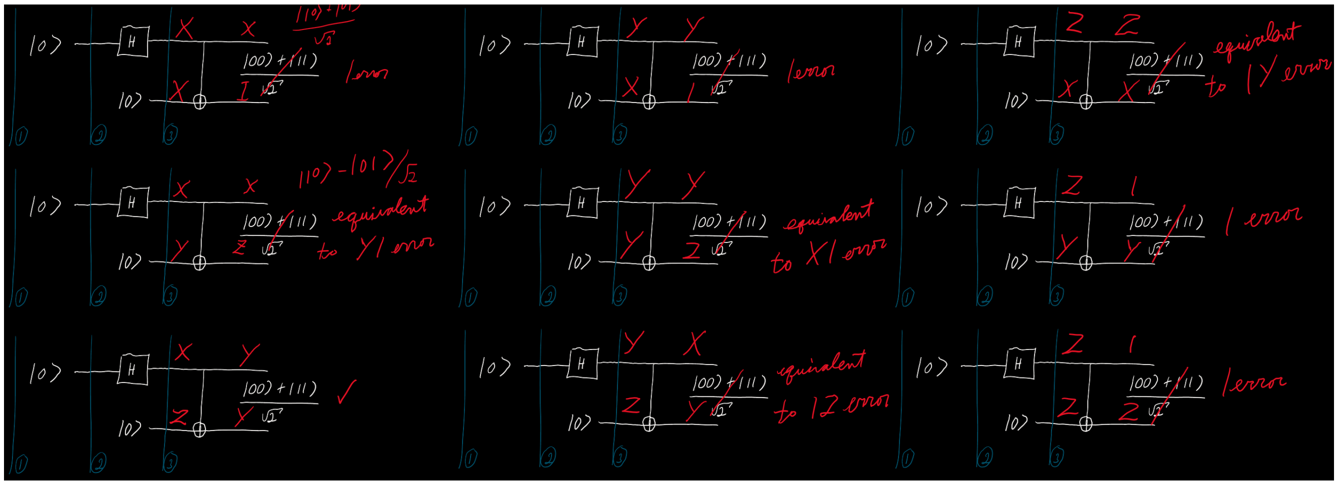
1. At step one of the circuit, there is one fault location.



2. At step two, there are two fault locations, but one of them has already been considered in step one.

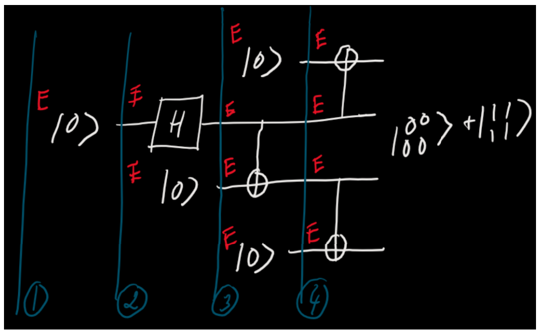


3. At step 3, all faults have already been considered.
4. Two-qubit gate errors occur on two qubits, so we have to take those into account by considering weight-two Paulis at any two-qubit gate locations. There is one two-qubit gate, and there 15 different two-qubit faults that can occur there. We have already considered all weight-one faults of the form $I\star$ and $\star I$, and the nine weight-two faults are considered below. Although several of them result in weight-two errors, all such cases are equivalent to another weight-one error and so are all detectable.

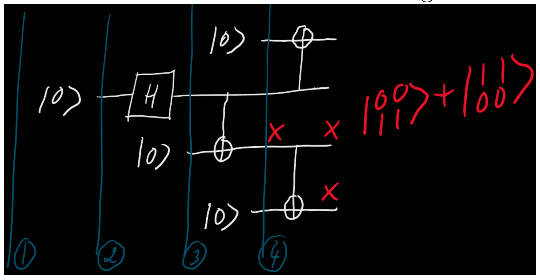


- Let us now consider making a $|0_L\rangle$ state, which is a GHZ state of four qubits.

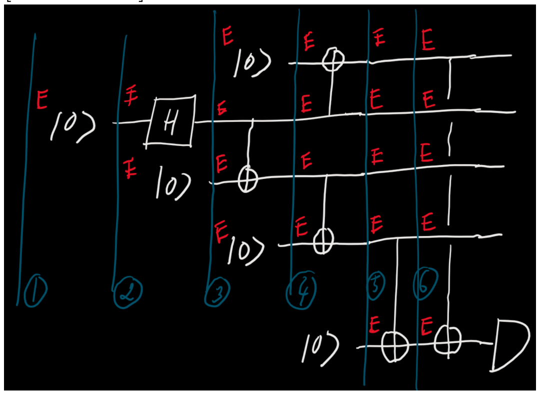
   1. We can naively consider simply extending the above circuit by adding successive CNOTs to spread the entanglement.



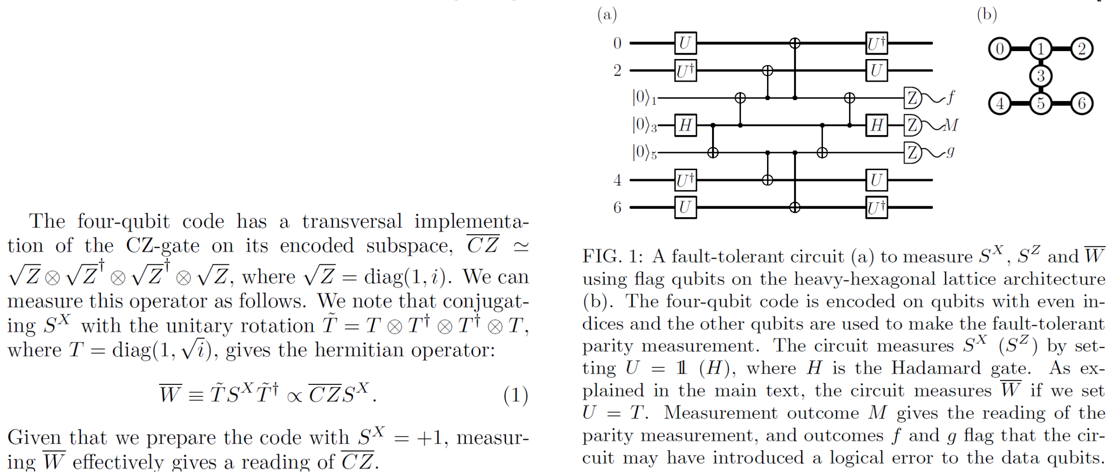
This is not fault-tolerant since a single X fault on third qubit causes two errors at output:



2. We can make the above circuit FT by adding a **flag qubit** that detects the above error pattern [1610.03507]:



- More involved FT circuits for preparing magic states are also possible for the four-qubit code [2305.13581]:



(a)
(b)
0
2
$|0\rangle_1$
$|0\rangle_3$
$|0\rangle_5$
4
6
U
$U^\dagger$
H
Z
f
M
g
0
1
2
3
4
5
6

The four-qubit code has a transversal implementation of the CZ-gate on its encoded subspace, $\overline{CZ} \simeq \sqrt{Z} \otimes \sqrt{Z}^\dagger \otimes \sqrt{Z}^\dagger \otimes \sqrt{Z}$, where $\sqrt{Z} = \text{diag}(1, i)$. We can measure this operator as follows. We note that conjugating S^X with the unitary rotation $\tilde{T} = T \otimes T^\dagger \otimes T^\dagger \otimes T$, where $T = \text{diag}(1, \sqrt{i})$, gives the hermitian operator:

$$\overline{W} \equiv \tilde{T} S^X \tilde{T}^\dagger \propto \overline{CZ} S^X. \tag{1}$$

Given that we prepare the code with $S^X = +1$, measuring $\overline{W}$ effectively gives a reading of $\overline{CZ}$.

FIG. 1: A fault-tolerant circuit (a) to measure S^X, S^Z and $\overline{W}$ using flag qubits on the heavy-hexagonal lattice architecture (b). The four-qubit code is encoded on qubits with even indices and the other qubits are used to make the fault-tolerant parity measurement. The circuit measures S^X (S^Z) by setting $U = \mathbb{1}$ (H), where H is the Hadamard gate. As explained in the main text, the circuit measures $\overline{W}$ if we set $U = T$. Measurement outcome M gives the reading of the parity measurement, and outcomes f and g flag that the circuit may have introduced a logical error to the data qubits.

0.12 Summary

1. Classical states are elements of a space X; quantum states are functions on X.
2. Error-correction paradigm works for spatio-temporal channels & classical/quantum info [Shannon].

3. Quantum codes have to protect against both bit- and phase-flip errors; there is a tradeoff. QEC requires space-time overhead, which can be "Wick-rotated" (e.g., MBQC).
4. Degeneracy makes decoding harder; yields connections to statistical mechanics.
5. Geometric locality is physically relevant, but handicaps code parameters (QLDPC).
6. Circuit-centric approach emerging that requires less overhead for same robustness (e.g., Floquet).
7. Fault tolerance is the art of using QEC to make sure errors are not amplified during performance of desired task.
8. QEC has many non-computational applications (e.g., sensing, holography, topological order).