Quantum error correction

MPRI 2025-2026

Anthony Leverrier

January 26, 2026

Contents

Lecture 0

General motivation

The goal of this course is to explain some of the main ideas behind quantum error correction, notably the stabilizer formalism, and to present recent constructions of good quantum LDPC codes.

The field of quantum error correction is rather recent. It started around 1996 when Peter Shor and others showed that quantum error correction and robust quantum computation were possible after all[1]. Before that, quantum computing was a fairly confidential field. While Feynman had famously dreamt of quantum machines capable of simulating quantum physics in the early 80s, this dream had remained fairly out-of-reach until someone figured out that quantum information was compatible with error correction, and in this sense, was completely different from (classical) analog computation which was studied and then forgotten since the 60s, precisely because there was no hope of making it error-free.

0.1 Goal of quantum error correction

The need of quantum error correction for useful quantum computing cannot be overstated. Today's hardware reach gate fidelities with 2 or 3 nines, that is between 99 and 99.9%. While these numbers already require feats of engineering, they are incredibly far from sufficient. In particular, if a circuit is made from gates with fidelity $1 - 10^{-3}$, then it means that the result of the computation will be faulty after a few thousand gates. Running interesting algorithms about quantum chemistry or Shor's factoring algorithm require many more gates than that, say $10^{12} - 10^{15}$ gates. Gate fidelity needs to improve by about 10 orders of magnitude! What's the solution? Quantum error correction!

It is well known that quantum systems like to couple to the environment, and that this leads quantum information to quickly decohere: the environment will essentially try to measure the quantum system and therefore prevents any useful quantum computation to take place. Quoting Barbara Terhal, the role of a quantum code is to make a *robust channel for quantum information to flow from past to future*. In other words, it will protect information by encoding it in such a way that it flows as in an (approximately) closed system.

It is quite amazing that it is possible to develop error correction and fault-tolerant techniques for quantum information. This is far from obvious since the quantum errors seem to belong to a continuum, much like what happens for analog computing. The key difference is that measuring part of the system will project the error onto a finite, discrete set, and it will be sufficient to be able to correct errors from that set. Thanks to this property, we will see that quantum fault tolerance is possible *in principle*: the set of techniques developed in quantum error correction

[1] Shor's algorithm for factoring also played an important role in putting the whole field of quantum computing under the spotlights!

and fault tolerance allow one to perform arbitrarily long quantum computations with reasonable (polylogarithmic in the number of gates) overhead (provided the qubits and gates are good enough, and that the noise can be modeled in a sufficiently nice way): this is the *threshold theorem*.

Theorem 0.1 (Aharonov, Ben-Or '97 [ABO97])**.** *Provided that the noise level is below some constant threshold, a logical circuit using m qubits and containing T gates can simulated by a fault-tolerant circuit using $O(m \operatorname{polylog}(mT))$ qubits.*

In practice, however, the overhead is quite large and finding better techniques that reduce this overhead is a major open problem if we want to build large-scale universal quantum computers. For instance, breaking cryptographic size RSA key requires a few thousands ideal qubits, but between 10^7 and 10^8 physical (good enough) qubits with state-of-the-art quantum fault-tolerance (theoretical) techniques. This is in part because qubits and quantum gates are much more noisy than classical bits and gates, and also because we still likely haven't discovered the optimal solutions to quantum error correction and fault tolerance.

The NISQ era: computing without error correction. In fact, today, quantum error correction has barely been experimentally implemented. Some recent experiments have shown that a protected qubit can store information longer than unprotected ones, but only for restricted noise models. Because of the lack of experimentally available quantum error correcting schemes, a lot of the current research tries to study noisy intermediate scale quantum (NISQ) systems, *i.e.*, systems of about 50 to 100 qubits, without any active error correction mechanism[2]. In this setup, one cannot implement Shor's factoring algorithm and it's interesting to understand whether this kind of machines can do anything useful better than a classical computer (say, for solving approximately optimization problems). This is related to the challenge of *quantum supremacy* (allegedly demonstrated by Google in Nov. 2019) which aims at performing a task with a quantum machine *provably* much faster than any classical computer could. Usually, the tasks for which we can prove this kind of advantage are pretty useless in practice: for instance, sampling a string from the output distribution of a random quantum circuit.

In this course, however, we are optimistic and assume that experimentalists will manage to improve qubits as well as quantum gates, and will be able to individually control a large number of qubits. If this is the case, then they will be able to implement quantum error correction and fault tolerance techniques, which will allow them to perform arbitrary long computations. The main requirements are that the noise level is below some constant threshold, which is around $0.1 - 1\%$, and that they can control a large number of qubits without increasing the noise level.

0.2 Is QEC possible at all?

When Shor's algorithm for factoring came out, computer scientists became quite excited with the idea of quantum computing. At the same time, physicists, and especially experimentalists had doubts about the possibility of quantum error correction. Without it, one would need to reach better and better gate fidelities to perform longer computations.

While classical error correction is very intuitive – simply add enough redundancy to the message to protect it from noise – the quantum setting appears much more challenging:

- Large entangled states are fragile: a single qubit decohering or leaking out destroys the whole superposition. For instance, if you start with a cat state

$$\frac{1}{\sqrt{2}}|0\rangle|\text{Alive}\rangle + \frac{1}{\sqrt{2}}|1\rangle|\text{Dead}\rangle$$

[2]In NISQ devices, error correction is replaced with error mitigation, which is a set of techniques aiming at lowering the effect of noise on the system.

and lose the first qubit (*i.e.,* corresponding mathematically to a partial trace), then the remaining state is

$$\frac{1}{2}|\text{Alive}\rangle\langle\text{Alive}| + \frac{1}{2}|\text{Dead}\rangle\langle\text{Dead}|,$$

which corresponds to a classical mixture of Alive and Dead.

- Applying noisy gates just increases the total noise: errors pile up. Even worse, any error correction mechanism that we wish to implement will also be prone to errors. This means that even if we can correct errors, we still need to correct them more quickly than they appear. A similar problem was present in the 50's for classical computing and Von Neumann developed a theory of fault-tolerant computation (that very much inspired the quantum version!). Fortunately, transistors quickly became so good that this classical theory became essentially pointless[3].
- As already mentioned, the possible errors seem to form a continuum in the quantum case: a quantum computation typically consists in applying some unitary U to an n-qubit state $|0\rangle^{\otimes n}$ to obtain $U|0\rangle^{\otimes n} \in (\mathbb{C}^2)^{\otimes n}$. Any process that prepares a state ε-far from the target state is problematic. How do you deal with that?
- The no-cloning theorem says that an operation that does $|\psi\rangle \mapsto |\psi\rangle|\psi\rangle$ for an arbitrary state $|\psi\rangle$ is not unitary and therefore forbidden. How do you protect quantum information without the ability to copy it, as you would do in the classical case (using the repetition code, say)?
- It is not possible to simply measure the output state to determine what it is since measuring the state will disturb it, and collapse it on a basis element.
- Assume that you manage to perform a quantum computation in a fault-tolerant way, or that some oracle gives you the final state $U|0\rangle^{\otimes n}$. At the very last step, you still need to measure this state to obtain the classical solution to the problem of interest, for instance some information about the factors of some large number. Given that this final measurement will inevitably be a little bit noisy, doesn't that ruin the whole computation?

0.3 Classical error correction and fault-tolerance

Before discussing quantum error correction, let's quickly discuss classical error correction first. A natural classical noise model is the *binary symmetric channel*, corresponding to a channel from Alice to Bob[4] (or Alice at time $t = 0$ to Alice at time $t = 1$) where each bit is independently flipped with probability $p \in [0, 1]$. The simplest solution to protect information is to add redundancy, for instance by repeating any single bit three times ($0 \mapsto 000$, $1 \mapsto 111$) and decoding[5] by taking a majority vote:

- encoding: $0 \to 000 =: \bar{0}$, $1 \to 111 =: \bar{1}$. Here, $\bar{0}$ and $\bar{1}$ form a basis for the *logical* (sometimes *encoded*) bit, while 000 and 111 refer to *physical* bits.
- channel: each bit is flipped independently with probability p,
- decoding: majority vote: $\{000, 100, 010, 001\} \to \bar{0}, \{111, 011, 101, 110\} \to \bar{1}$.

[3] Classical fault tolerance is making a come-back these days thanks for instance to the importance of computing over distributed systems.

[4] In the classical setting, computations are essentially error-free, and the issue of error correction is mostly relevant to communication between two distant parties who try to communicate over an imperfect channel.

[5] The word "decoding" can refer to 2 different operations. The true definition refers the map that is the inverse of the encoding operation: it maps an encoded logical state to an unencoded physical state, $000 \mapsto 0$ and $111 \mapsto 1$. However, it is very usual to use the word "decoding" to also include the error correction procedure. In that case, it refers both to the action of trying to correct the error that occurred and in a second time, to return the unencoded state, here $ijk \mapsto \ell$ if $|ijk \oplus \ell\ell\ell| \leq 1$.

This scheme produces the correct result if the channel flipped 0 or 1 bit (and detects that an error occurred if 1 or 2 errors occurred). The error probability drops from p for the physical bit to $(1-p)^3 + 3p(1-p)^2 = 3p^2 - 2p^3 \leq 3p^2$ for the logical bit. This reduces the error rate provided that $p < \frac{1}{2}$. If $p = \frac{1}{2}$, then the channel cannot transmit any information, and if $p > \frac{1}{2}$, one should simply flip the bit to recover the case $p < \frac{1}{2}$.

This coding strategy protects against bit flips, but is quite inefficient. Much better schemes exist, but this is a good starting point to discuss quantum coding.

Classical fault-tolerance theorem. In the scheme above, we assumed that the encoding and decoding could be implemented perfectly. In other words, that they are noiseless. While this is a reasonable assumption today, it wasn't the case in the early days of computing with vacuum tubes, which were faulty. Skeptics back then would argue that any sufficiently long computation was doomed to fail since errors would necessarily occur in encoding/decoding circuits. John von Neumann proved that this reasoning was actually incorrect. He showed that if logic gates (AND, OR, NOT) fail with some independent probability ε, then provided that ε is small enough, it is possible to build a reliable circuit for any Boolean function f. Moreover, the circuit is only reasonably larger than the circuit for f built of perfect gates.
One question though: what happens if the final gate is faulty with probability ε? It seems that you can never get a correct answer with probability greater than $1 - \varepsilon$! A solution is to encode the final result into a long string or to repeat the computation several times and take a majority vote (assumed to be error-free, because simple to implement[6]).
To prove the *classical* threshold theorem, one can use the 3-bit repetition code recursively, where each physical bit is actually replaced by the logical bit of another layer of repetition code. Even if each operation is noisy, there exists a threshold below which each level of encoding leads to a decrease of the logical error rate.
It turns out that the classical fault-tolerance theorem ended up being useless when vacuum tubes were replaced by transistors because the error rate for logic gates became ridiculously small (much smaller than the inverse of the number of gates in any useful computation).
The same thing might occur one day with quantum computing and topological quantum computing[7] might be an idea to get there, but we are currently *very, very* far from that situation. And it is therefore crucial to develop a theory of quantum error correction and quantum fault-tolerance.

The main goal of the course will be to explain the main ideas behind quantum error correction and to present some notable quantum error correcting codes. We will not focus much on fault tolerance as this would likely require another entire course. We will only quickly mention the quantum threshold theorem that provides a proof that robust quantum computation is possible. Much better schemes have been designed since then but the question of how to best achieve quantum fault tolerance remains a very active area of research at them moment. Quantum codes are easier to study, because they are only concerned with protecting quantum information from noise, while quantum fault tolerance also require to perform some computation on the system. Quite surprisingly, there remains much to discover about quantum error correcting codes. For instance, we will see at the end of the course the construction of a specific family of good quantum codes [PK22, LZ22], and these codes were not even known to exist a couple of years ago.

[6]The idea that something simple to implement can be done error-free is actually crucial. In the quantum computing case, we will assume that *classical operations* can be done without any error, and this is indeed a very reasonable hypothesis. In particular, it will be sufficient to encode the final result with a classical error correcting code to make sure that the final quantum step doesn't ruin the whole computation.

[7]For instance, with topological qubits such as Majorana qubits.

0.4 Outline of the course

This mini-course focuses on a small but very active part of quantum error correction: *quantum LDPC codes*. We assume that the basic mathematical formalism of quantum information (density matrices, quantum channels, partial trace, measurements, etc.) is already familiar.

The remaining lectures cover the following material:

- Lecture 1: simple quantum codes (3-qubit repetition, Shor's $[[9, 1, 3]]$ code), criteria for error correction, and a first look at fault tolerance and the threshold theorem.
- Lecture 2: stabilizer codes and CSS codes (the linear-algebraic framework that underlies most constructions).
- Lecture 3: quantum LDPC codes through the toric code and hypergraph product codes.
- Lecture 4: modern constructions of *good* quantum LDPC codes, with an emphasis on quantum Tanner codes and current research directions.

We will only touch on decoding (e.g. MWPM for the toric code) and will not cover fault-tolerant logical gate sets in detail; both are major topics in their own right.

0.5 Resources

One can find a lot of great resource about quantum error correction online. For instance,

- Gilles Zémor's lecture notes: introduction to quantum error correction
- John Preskill's course at Caltech
- Dan Browne's course at UCL on "Topological Codes and Computation"
- Daniel Gottesman's PhD thesis introducing stabilizer codes
- Error correction zoo: categorizes and to organizes known classical and quantum error-correction schemes
- Opportunities and Challenges in Fault-Tolerant Quantum Computation [Got22]: very recent review of the field by Daniel Gottesman, for the 28th Solvay conference on physics

Lecture 1

Shor's code, criteria for error correction, fault tolerance and the threshold theorem

In the first lecture, we have claimed that some error correction mechanism was necessary in order to perform useful quantum computation, and provided a long list of hurdles that need to be addressed for quantum error correction to be possible at all.

Let us quickly mention possible solutions to these challenges:

- *One cannot copy quantum information.* While it is true that the operation $|\psi\rangle \mapsto |\psi\rangle|\psi\rangle$ is not unitary, and therefore forbidden by the laws of quantum physics, one can still encode information redundantly by adding some ancilla qubits to the state and applying a global encoding map $\mathcal{E}$, so as to prepare a state $\mathcal{E}(|\psi\rangle|0\rangle^{\otimes m})$. This doesn't copy the information in additional registers, but still delocalizes the initial information from k initial physical qubits (if $|\psi\rangle$ is a k-qubit state) to $n := k + m$ qubits. In particular, if the encoding map is well chosen then one will be able to reverse the effect of noise provided that it doesn't affect too many of the n qubits.
- *There is a continuum of possible errors.* True. The key here will be to perform specific measurements on the n physical qubits. This will collapse the overall state onto a finite set of possible (orthogonal) states, and also provide some classical information (the classical outcome of the measurement is called the *syndrome* of the error). One then simply needs to be able to recover the error that gave rise to a specific syndrome. This is a purely classical problem called the *decoding problem*.
- *What about errors occurring at the very last step of the computation?* This is easy provided that you are only interested in a classical answer: simply encode the solution in a classical error correcting code and run a classical decoding algorithm to correct for errors.

The goals of this lecture are

- to introduce Shor's 9-qubit code which protects a qubit by encoding it into nine physical qubits, in such a way that an arbitrary error on a single qubit can be corrected,
- to give criteria for error correction: what set of errors can be handled by a given code,
- and to state the *threshold theorem*.

1.1 How to protect a single qubit from noise: Shor's code

1.1.1 A first example: the 3-qubit code Shor's code

We want to generalize the 3-bit repetition code to the quantum setting. The goal is as follows: Alice has a qubit in a pure state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and wants to send it to Bob (or to keep it in a noisy memory for some time). Here, we suppose that Alice doesn't know α and β. If she knew the qubit exactly, she could simply send the classical description to Bob by classical means who would reconstruct it. But this wouldn't be efficient when sending n-qubit states (since this would require 2^n amplitudes in general...) and in practical situations, Alice doesn't know the values of α and β.

For concreteness, let us first consider a restricted noise model similar to the binary symmetric channel where a qubit is flipped independently with probability p. In other words, with probability $1-p$, the channel acts as the identity, and with probability p, it applies a bit-flip error represented by the Pauli matrix $X = \sigma_X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. This quantum channel is called the *bit-flip channel* and is described by the *quantum channel* (or *admissible operation*)

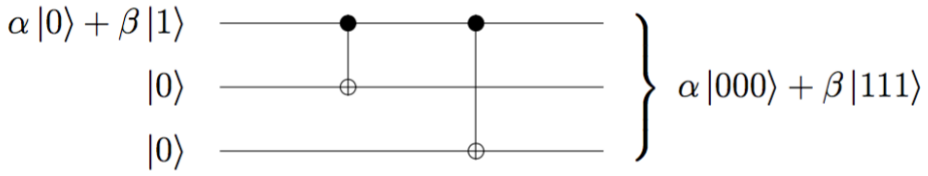
$$E_{\text{bit}}(\rho) = (1-p)\rho + pX\rho X.$$

First, the no-cloning theorem prevents us to apply the map $|\psi\rangle \to |\psi\rangle|\psi\rangle|\psi\rangle$ since it is not unitary. A better approach is to perform the following encoding isometry

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \mapsto \alpha|000\rangle + \beta|111\rangle =: \alpha|\bar{0}\rangle + \beta|\bar{1}\rangle = |\bar{\psi}\rangle,$$

where we denote by $|\bar{k}\rangle$ the logical (or encoded) qubit state corresponding to $|k\rangle$.

This encoding can be realized with a simple circuit involving two additional (ancillary) qubits and 2 CNOT gates:



The encoding map is therefore an isometry from $\mathbb{C}^2$ to $(\mathbb{C}^2)^{\otimes 3}$. In general, if we want to encode k logical qubits within $n > k$ physical qubits, the encoding map will be a isometry from $(\mathbb{C}^2)^{\otimes k}$ to $(\mathbb{C}^2)^{\otimes n}$.

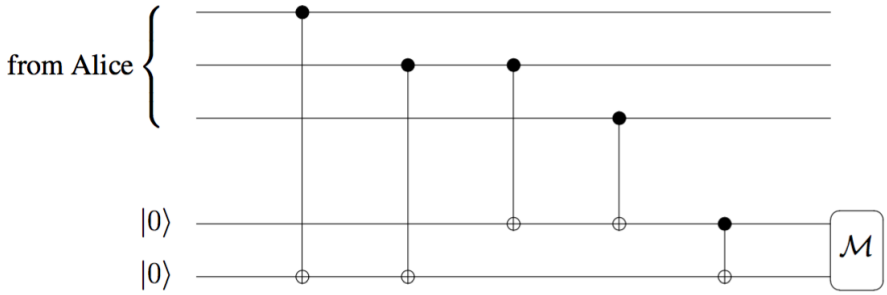
The 3-qubit state after the quantum channel is described by a density matrix on $(\mathbb{C}^2)^{\otimes 3}$:

$$\begin{aligned} E_{\text{bit}}^{\otimes 3}(|\bar{\psi}\rangle\langle\bar{\psi}|) =& (1-p)^3\rho + p(1-p)^2(X_1\rho X_1 + X_2\rho X_2 + X_3\rho X_3) \\ &+ p^2(1-p)(X_1X_2\rho X_1X_2 + X_1X_3\rho X_1X_3 + X_2X_3\rho X_2X_3) + p^3 X_1X_2X_3\rho X_1X_2X_3, \end{aligned}$$

where we wrote $\rho = |\bar{\psi}\rangle\langle\bar{\psi}|$ for conciseness.

How should we decode the state and perform the correction?? Measuring the whole state in the computational basis is a bad idea since one will get a basis state and the information about $|\psi\rangle$ will be destroyed. Rather, we want to copy what we did in the classical case to notice that an error had occurred: for this, we compare the values of the 3 bits pairwise, *i.e.*, we measure parities. If they all agree, then we conclude that no error occurred (or maybe that three errors occurred,if we were unlucky). If some values disagree, we perform a correction.

A circuit for measuring these parities is drawn below:



Let us consider what this circuit does on an example, for instance if the second qubit was flipped (the error is X_2). The 5-qubit state (3 qubits for the computation and 2 additional qubits for the parity measurements) is given by

$$\begin{aligned}(\alpha|010\rangle + \beta|101\rangle)|00\rangle &= \alpha|010\rangle|00\rangle + \beta|101\rangle|00\rangle && \text{at the beginning of the circuit}\\ &\mapsto \alpha|010\rangle|10\rangle + \beta|101\rangle|10\rangle && \text{before the parity measurement } \mathcal{M}\\ &= (\alpha|010\rangle + \beta|101\rangle)|10\rangle.\end{aligned}$$

Measuring the last two qubits yields the outcome 10 with certainty. This output string is called the *syndrome* and it tells us that a bit-flip occurred on the second qubit (qubit 10 in binary). Bob can therefore correct the error by applying the correction $X = \sigma_X$ to qubit 2:

$$\alpha|010\rangle + \beta|101\rangle \mapsto \alpha|000\rangle + \beta|111\rangle = |\overline{\psi}\rangle.$$

Finally, he can apply the inverse of the encoding circuit to recover the initial qubit

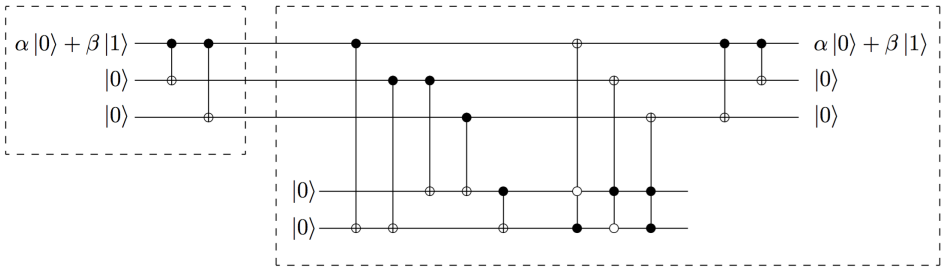
$$\alpha|000\rangle + \beta|111\rangle \mapsto \alpha|0\rangle + \beta|1\rangle.$$

This procedure works if at most one bit flip occurs:

- no error, classical states $|000\rangle, |111\rangle \longrightarrow$ syndrome 00,
- bit flip on the 1st qubit, classical states $|100\rangle, |011\rangle \longrightarrow$ syndrome 01,
- bit flip on the 2nd qubit, classical states $|010\rangle, |101\rangle \longrightarrow$ syndrome 10,
- bit flip on the 3rd qubit, classical states $|001\rangle, |110\rangle \longrightarrow$ syndrome 11.

We will see later that measuring the syndrome is equivalent to measuring the value of the observables Z_1Z_2 and Z_2Z_3, which check the parity of the first and second qubits, and of the second and third qubits. (With the convention above, it is really the observables $\frac{1}{2}(\mathbb{1} + Z_2Z_3)$ and $\frac{1}{2}(\mathbb{1} + Z_1Z_3)$.)

The overall circuit, including encoding on Alice's side and syndrome measurement, error correction and decoding on Bob's side is summarized here:



Similarly as in the classical case, the error probability goes from p for unencoded physical qubits to $3p^2 - 2p^3$ for the encoded logical qubit. Here, we have assumed that all the gates inside Alice and Bob's labs, namely the gates involved in the encoding, the syndrome measurement and the decoding operation, are perfect.

1.2 Going beyond bit-flips: the 9-qubit code

Of course, we know that qubits are prone to more general errors than simply bit-flips. Another very natural error is the phase-flip, corresponding to an application of the matrix $Z = \sigma_Z = \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right)$. It should be clear that our 3-qubit code does not protect against this type of errors. For instance, a phase-flip on any of the three physical qubits (i.e. Z_1, Z_2 or Z_3) yields

$$\alpha|000\rangle + \beta|111\rangle \mapsto \alpha|000\rangle - \beta|111\rangle,$$

which is the encoded version of $\alpha|0\rangle - \beta|1\rangle$. Measuring the syndrome would give 00 with certainty and the error cannot be corrected. As before, if we suppose that the probability for an individual phase-flip is $0 < p < \frac{1}{2}$, then the probability that the logical qubit is corrupted (corresponding to one or three corrupted physical qubits) is

$$3p(1-p)^2 + p^3 > p.$$

If we simply wanted to protect against phase-flip errors only, then the following encoding would work:

$$\alpha|0\rangle + \beta|1\rangle \mapsto \alpha|{+}{+}{+}\rangle + \beta|{-}{-}{-}\rangle.$$

To do that, one can simply perform the same encoding circuit as before, followed by a Hadamard transformation on the output qubits. This is because $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$. Then, exactly the same analysis as before goes through, and you get a coding scheme that can correctly deal with 0 or 1 phase-fip error ... but unfortunately not anymore with a single bit-flip error.

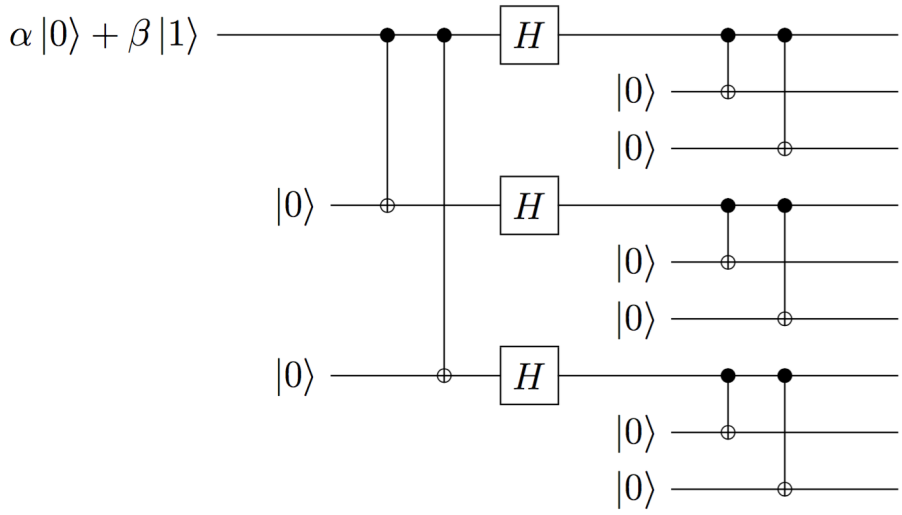
Is there a way to protect against both bit-flip and phase-flip errors at the same time? Not with a 3-qubit code. But one can with a simple 9-qubit code: Shor's 9-qubit code. The idea relies on *concatenation*: one first applies the code that protects against phase-flips then one encodes the resulting physical qubits with the code that protects against bit-flips. The encoding of a single-qubit $\alpha|0\rangle + \beta|1\rangle$ is given by

$$\begin{aligned}
\alpha|0\rangle + \beta|1\rangle &\mapsto \alpha|{+}{+}{+}\rangle + \beta|{-}{-}{-}\rangle && \text{(first level of encoding)} \\
&= \alpha\frac{1}{2\sqrt{2}}(|0\rangle + |1\rangle)^{\otimes 3} + \beta\frac{1}{2\sqrt{2}}(|0\rangle - |1\rangle)^{\otimes 3} \\
&\mapsto \alpha\frac{1}{2\sqrt{2}}(|000\rangle + |111\rangle)^{\otimes 3} + \beta\frac{1}{2\sqrt{2}}(|000\rangle - |111\rangle)^{\otimes 3} && \text{(second level of encoding)}
\end{aligned}$$

The logical qubit is spread over 9 physical qubits:

$$\begin{aligned}
|\bar{0}\rangle &= \left(\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)\right)^{\otimes 3}, \\
|\bar{1}\rangle &= \left(\frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)\right)^{\otimes 3}.
\end{aligned}$$

The encoding circuit is obtained by concatenating the two encoding circuits of the two 3-qubit codes.



To decode, Bob applies successively the decoding procedures of the two 3-qubit codes:

(*i*) he first considers the three blocks of three qubits and each such block is an encoding of one of the 3 qubits of the state $\alpha|+\rangle^{\otimes 3} + \beta|-\rangle^{\otimes 3}$. He can correct bit flips as before, and obtains 3 qubits (keeping only the first one in each corrected block). Provided that at most one bit-flip occurred in each block, then the 6 other qubits are all in the state $|0\rangle$ and can be discarded.

(*ii*) Bob now has the same three qubits as he would when encoding with the code that protects against at most a single phase-flip. He can decode it as before, and provided there is at most a single phase-flip, he recovers the correct state $\alpha|0\rangle + \beta|1\rangle$.

Overall, the qubit is recovered if there was at most a single bit-flip or a single phase-flip. Given that a Y-Pauli matrix is the product $Y = iZX$, and that the code above clearly protect against a single Y-error (since the X part of the error will be corrected by the second level of encoding and the Z part will be corrected by the first level), we have proven the following result:

Theorem 1.1. *The 9-qubit Shor code protects against any of the four Pauli errors* $\mathbb{1}, X, Y = iZX, Z$ *occurring on a single qubit.*

Example 1.2. Suppose the error $Y = XZ$ (we can forget about global phases here) occurs on qubit 6. The encoded state becomes (also forgetting about normalization)

$$\alpha(|000\rangle + |111\rangle)(|00\mathbf{1}\rangle - |11\mathbf{0}\rangle)(|000\rangle + |111\rangle) + \beta(|000\rangle - |111\rangle)(|00\mathbf{1}\rangle + |11\mathbf{0}\rangle)(|000\rangle - |111\rangle).$$

The syndrome measurement for the code protecting against bit-flips yields 00, 11 and 00 for the 3 blocks of 3 qubits. Bob infers that there is no bit-flip error in the first block, a bit-flip on the third qubit of the second block and no bit-flip error in the third block. Applying the correction X_6 yields

$$\alpha(|000\rangle + |111\rangle)(|000\rangle - |111\rangle)(|000\rangle + |111\rangle) + \beta(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)(|000\rangle - |111\rangle).$$

Finishing the decoding of the first code (by applying the inverse of the encoding circuit) gives

$$\alpha(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)(|0\rangle + |1\rangle) + \beta(|0\rangle - |1\rangle)(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) = \alpha|+-+\rangle + \beta|-+-\rangle.$$

Measuring the syndrome of the code protecting against phase-flips gives 10 indicating that a phase-flip occurred on the second qubit. Correcting for it by applying a Z correction and inverting the encoding circuit finally yields $\alpha|0\rangle + \beta|1\rangle$, as expected.

As before, we can understand the measurement of the syndrome as measuring some observables:

- one can detect a single bit flip in each subblock of size 3 as before, by measuring the Pauli operators $Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9$,
- in order to detect phase errors, we proceed similarly by testing the parities of successive subblocks. Now the equivalent of Z_1 is X applied to the first logical qubit, which is simply $X_1X_2X_3$. In other words, one simply measures the operators $X_1X_2X_3X_4X_5X_6$ and $X_4X_5X_6X_7X_8X_9$. If they both give $+1$, then there wasn't a phase error, otherwise one can deduce where the phase-flip occurred as before.

Let us consider the case of the depolarizing channel with parameter p applied independently to each of the 9 qubits. For a single qubit, this channel is described by the map

$$\rho \quad \mapsto \quad (1-p)\rho + \frac{p}{3}X\rho X + \frac{p}{3}Y\rho Y + \frac{p}{3}Z\rho Z.$$

For each qubit, the probability that a bit flip occurs is $2p/3$ (corresponding to errors X and Y), the probability that a phase flip occurs is also $2p/3$ (corresponding to errors Y and Z) and the probability that both errors occur is $p/3$ (corresponding to a Y error).

The analysis above shows that

- a bit flip on the logical qubit can only occur if there are two bit flip errors in a single cluster of 3 qubits: this occurs with probability at most $3 \times 3\left(\frac{2p}{3}\right)^2 = 4p^2$. The first factor 3 counts the number of clusters, the second corresponds to the choice of 2 positions in error for the faulty cluster.
- a phase flip on the logical qubit can only occur if there is a bit flip in two distinct clusters of 3 qubits. This happens with probabililty at most $3 \times \left(3 \times \frac{2p}{3}\right)^2 = 12p^2$, where the first factor counts the possible choices of two faulty errors, and the second factor counts the probability of a single phase-flip inside a cluster.

Overall, we see that the logical error probability is at most $16p^2$, which is an improvement compared to the unencoded qubit which is erroneous with probability p, provided that $p \leq \frac{1}{16}$.

1.3 Criteria for error correction

Once we have defined a quantum code, a natural question is which set of errors can be corrected. Consider a set of errors $\mathcal{E} = \{E_a\}_a$ that we wish to be able to correct. The idea is to perform a collective measurement (to obtain the *syndrome* of the error) and try to determine which E_a occurred and reverse it by applying $E_a^\dagger$.

A typical choice for the set $\mathcal{E}$ is the set of all Pauli errors of weight up to t. If we can correct any such error, then we say that the code can correct t errors.

Given some set of errors $\mathcal{E} = \{E_a\}$ to be corrected, let us infer some necessary and sufficient conditions to be satisfied by the code. Let us denote by $|\bar{i}\rangle$ an orthonormal basis of the codespace.

First, we note that it is *necessary* that for all $E_a, E_b \in \mathcal{E}$, we have

$$\langle \bar{i}|E_a^\dagger E_b|\bar{j}\rangle = 0, \quad \text{if} \quad i \neq j,$$

since otherwise the error would map perfectly distinguishable states $|\bar{i}\rangle$ and $|\bar{j}\rangle$ to non orthogonal states $E_a|\bar{i}\rangle$ and $E_b|\bar{j}\rangle$ that cannot be perfectly distinguished.

Considering the orthogonal states $\frac{1}{\sqrt{2}}(|\bar{i}\rangle \pm |\bar{j}\rangle)$, we obtain that

$$(\langle \bar{i}| + \langle \bar{j}|)E_a^\dagger E_b|(|\bar{i}\rangle - |\bar{j}\rangle) = 0 \quad \Longrightarrow \quad \langle \bar{i}|E_a^\dagger E_b|\bar{i}\rangle = \langle \bar{j}|E_a^\dagger E_b|\bar{j}\rangle =: c_{ab}.$$

If we denote by $\Pi = \sum_i |\bar{i}\rangle\langle\bar{i}|$ the projector onto the codespace, we see that

$$\begin{aligned}\Pi E_a^\dagger E_b \Pi &= \sum_{i,j} |\bar{i}\rangle\langle\bar{i}|\, E_a^\dagger E_b\, |\bar{j}\rangle\langle\bar{j}| \\ &= \sum_i c_{ab} |\bar{i}\rangle\langle\bar{i}| \\ &= c_{ab}\Pi.\end{aligned}$$

This is in fact a sufficient and necessary condition, known as the *Knill-Laflamme condition*:

$$\Pi E_a^\dagger E_b \Pi = c_{ab}\Pi, \quad \forall E_a, E_b \in \mathcal{E}. \tag{1.1}$$

Before proving in detail that this condition is sufficient, let us note that it is intuitively *sufficient* to have the stronger condition:

$$\langle\bar{i}|E_a^\dagger E_b|\bar{j}\rangle = \delta_{a,b}\delta_{i,j}, \qquad \forall i,j, \forall E_a E_b \in \mathcal{E}. \tag{1.2}$$

Indeed, this would imply that each error E_a maps the code space $\mathcal{H}_{\text{code}} \subset (\mathbb{C}^2)^{\otimes n}$ onto a distinct "error subspace" $E_a\mathcal{H}_{\text{code}}$, with all such subspaces being mutually orthogonal. In that case, one could simply perform a measurement to decide in which of these subspaces the state lies and then perform the appropriate correction.

If a code satisfies (1.2), it is called a *nondegenerate* code and there exists a measurement that can unambiguously identify the error E_a that occurred. We will see later in the course that many interesting quantum codes (and in particular the class of LDPC codes) are degenerate. This means that several (and in general many) distinct errors give the same syndrome, but can be corrected nevertheless.

Theorem 1.3 (Knill-Laflamme conditions)**.** *A set $\mathcal{E}$ of errors can be corrected by a code* $\text{Span}(|\bar{0}\rangle, \ldots, |\bar{i}\rangle, \ldots)$ *if and only if*

$$\langle\bar{i}|E_a^\dagger E_b|\bar{j}\rangle = c_{ab}\delta_{i,j}, \qquad \forall i,j, \quad \forall E_a, E_b \in \mathcal{E}, \tag{1.3}$$

where $c_{ab} = \langle\bar{i}|E_a^\dagger E_b|\bar{i}\rangle$ is some constant independent of i, or alternatively if

$$\Pi E_a^\dagger E_b \Pi = c_{ab}\Pi, \quad \forall E_a, E_b \in \mathcal{E}, \tag{1.4}$$

where Π is the projector onto the codespace.

Proof. The proof that this weaker condition compared to (1.2) is indeed sufficient involves constructing an explicit recovery procedure (a quantum channel) R that corrects an arbitrary error from $\mathcal{E}$. It involves two steps. Starting with a set $\mathcal{E} = \{E_a\}$ satisfying $\Pi E_a^\dagger E_b \Pi = c_{ab}\Pi$, we show the existence of another generating set $\mathcal{F} = \{F_i\}$ such that $\Pi F_i^\dagger F_j \Pi = d_i\delta_{i,j}\Pi$, and such that $\text{Span}(\mathcal{E}) = \text{Span}(\mathcal{F})$. The second step shows that one can correct the errors in $\mathcal{F}$.

- First step. Let us define the matrix $C = \sum_{a,b} c_{a,b}|a\rangle\langle b|$, where $\{|a\rangle\}$ forms an orthonormal basis. We note that C is a hermitian matrix, which means that there exists a unitary matrix U and a diagonal matrix D such that $C = UDU^\dagger$. Let us define

  $$F_i := \sum_a U_{ai} E_a.$$

  We observe that

  $$\begin{aligned}\Pi F_i^\dagger F_j \Pi &= \sum_{a,b} U_{ai}^* U_{bj} \Pi E_a^\dagger E_b \Pi \\ &= \sum_{a,b} (U^\dagger)_{ia} U_{bj} c_{ab} \Pi \\ &= (U^\dagger C U)_{ij}\Pi \\ &= d_j \delta_{ij}\Pi\end{aligned}$$

  which is what we wanted.

- Second step. The main idea of the correction procedure is to first design a measurement to identify which error has occurred. Let us define

$$M_\ell = \frac{1}{\sqrt{d_\ell}}(F_\ell \Pi F_\ell^\dagger)^{1/2} \otimes |\ell\rangle.$$

We first observe that the set $\{M_\ell\}_\ell$ defines a valid measurement. To see this, it is sufficient to show that $M := \sum_\ell M_\ell^\dagger M_\ell$ is a projector, that is, to show that $M^2 = M$. We have

$$M = \sum_\ell \frac{1}{d_\ell} F_\ell \Pi F_\ell^\dagger,$$

and therefore

$$\begin{aligned} M^2 &= \sum_{k,\ell} \frac{1}{d_k}\frac{1}{d_\ell} F_k \Pi F_k^\dagger F_\ell \Pi F_\ell^\dagger \\ &= \sum_{k,\ell} \frac{1}{d_k}\frac{1}{d_\ell} F_k (d_k \delta_{k,\ell} \Pi) F_\ell^\dagger \\ &= \sum_k \frac{1}{d_k} F_k \Pi F_\ell^\dagger \\ &= M \end{aligned}$$

Let us assume that a state $\rho = \Pi\rho\Pi$ has been through a noisy channel $\mathcal{N}$ and that each of the Kraus operators can be decomposed in terms of elements of $\mathcal{F}$,

$$\mathcal{N}(\rho) = \sum_{ij} \lambda_{ij} F_i \rho F_j^\dagger.$$

We note that the normalisation of the output state implies that

$$\sum_i \lambda_{ii} d_i = 1. \tag{1.5}$$

One can then apply a measurement map with Kraus operators $M_\ell \otimes |\ell\rangle$ and obtain $\mathcal{M}(\mathcal{N}(\rho))$ given by

$$\begin{aligned} \mathcal{M}(\mathcal{N}(\rho)) &= \sum_\ell M_\ell \Big(\sum_{ij} \lambda_{ij} F_i \rho F_j^\dagger \Big) M_\ell^\dagger \\ &= \sum_\ell \sum_{ij} \lambda_{ij} \frac{1}{d_\ell} (F_\ell \Pi F_\ell^\dagger)^{1/2} F_i \rho F_j^\dagger (F_\ell \Pi F_\ell^\dagger)^{1/2} \otimes |\ell\rangle\langle\ell|. \end{aligned}$$

We can finally define a recovery map $\mathcal{R}$ with Kraus operators

$$R_\ell := \Pi F_\ell^\dagger (F_\ell \Pi F_\ell^\dagger)^{-1/2} \otimes \langle \ell|.$$

These are well normalised since

$$\begin{aligned} \sum_\ell R_\ell^\dagger R_\ell &= \sum_\ell (F_\ell \Pi F_\ell^\dagger)^{-1/2} F_\ell \Pi F_\ell^\dagger (F_\ell \Pi F_\ell^\dagger)^{-1/2} \otimes |\ell\rangle\langle\ell| \\ &= \sum_\ell \mathbb{1}_{\mathrm{Supp}(F_\ell \Pi F_\ell^\dagger)} \otimes |\ell\rangle\langle\ell|. \end{aligned}$$

Applying this map after the measurement yields

$$\begin{aligned}
\mathcal{R}(\mathcal{M}(\mathcal{N}(\rho))) &= \sum_k R_k \Big(\sum_\ell \sum_{ij} \lambda_{ij} \frac{1}{d_\ell} (F_\ell \Pi F_\ell^\dagger)^{1/2} F_i \rho F_j^\dagger (F_\ell \Pi F_\ell^\dagger)^{1/2} \otimes |\ell\rangle\langle\ell| \Big) R_k^\dagger \\
&= \sum_k \Pi F_k^\dagger (F_k \Pi F_k^\dagger)^{-1/2} \sum_\ell \sum_{ij} \lambda_{ij} \frac{1}{d_\ell} (F_\ell \Pi F_\ell^\dagger)^{1/2} F_i \rho F_j^\dagger (F_\ell \Pi F_\ell^\dagger)^{1/2} (F_k \Pi F_k^\dagger)^{-1/2} F_k \Pi \otimes \langle k|\ell\rangle\langle \\
&= \sum_\ell \sum_{ij} \lambda_{ij} \frac{1}{d_\ell} \Pi F_\ell^\dagger (F_\ell \Pi F_\ell^\dagger)^{-1/2} (F_\ell \Pi F_\ell^\dagger)^{1/2} F_i \rho F_j^\dagger (F_\ell \Pi F_\ell^\dagger)^{1/2} (F_\ell \Pi F_\ell^\dagger)^{-1/2} F_\ell \Pi \\
&= \sum_\ell \sum_{ij} \lambda_{ij} \frac{1}{d_\ell} \Pi F_\ell^\dagger F_i \rho F_j^\dagger F_\ell \Pi \\
&= \sum_\ell \sum_{ij} \lambda_{ij} \frac{1}{d_\ell} \Pi F_\ell^\dagger F_i \Pi \rho \Pi F_j^\dagger F_\ell \Pi \\
&= \sum_\ell \sum_{ij} \lambda_{ij} \frac{1}{d_\ell} \delta_{\ell,i} d_\ell \Pi \rho \Pi \delta_{j,\ell} d_\ell \\
&= \sum_\ell \lambda_{\ell\ell} d_\ell \Pi \rho \Pi \\
&= \rho
\end{aligned}$$

where the last equality follows from the normalization condition of (1.5), and we used that ρ is an encoded state, *i.e.* $\rho = \Pi\rho\Pi$.

□

We have just proven that the effect of the noisy channel could be cancelled provided that the noise operators satisfy the *Knill-Laflamme conditions*. In general, implementing the appropriate syndrome measurement and error recovery might be quite challenging from an experimental point of view. In the next chapter devoted to the stabilizer formalism, we will focus on a class of codes that are well suited to correct Pauli errors.

1.4 Quantum Fault Tolerance: the threshold theorem

Once we have a quantum code that can correct an arbitrary error on a single qubit, it is possible to turn to the question of protecting a quantum computation. The main idea here is to encode information with this code, and perform the desired gates at the logical level, similarly to what was sketched in the classical case. When doing this, the physical error rate p is replaced by some effective noise $\leq Cp^2$ for some constant C (that depends on the number of gates necessary to implement a logical gate). The dependence is quadratic in p because two errors now need to occur to pose a problem, since single errors are corrected. In general, great care must be taken so as to avoid errors to propagate and spread through the circuit. In particular, note that a single error occurring before an ideal CNOT gate typically gives rise to 2 errors after the gate, even if this one is performed without error. For this reason, one needs to find gadgets to explain how to implement each possible gate in a fault-tolerant way, that is avoiding errors to spread.

Of course, for $p < 1/C$, the effective noise level Cp^2 is smaller than the physical error level p, but is still too large for our purpose in general. Say that one aims to perform an ideal circuit with T gates and targets a final error rate of $\varepsilon \ll p$, the idea is to repeat the procedure and to encode each of the physical qubits into another quantum code. Doing this decreases the error rate from Cp^2 to $C(Cp^2)^2 = (Cp)^4/C$. The number of qubits has increased, however. For instance, if we rely on Shor's 9-qubit code, then each level of encoding adds a factor 9 to the qubit count. Repeating this strategy over ℓ levels of encoding, we get a qubit count of $9^\ell n$, if

the initial circuit comprised n qubits, and an effective error rate of $(Cp)^{2^\ell}/C$. It is sufficient to take ℓ so that $(Cp)^{2^\ell}/C = \frac{\varepsilon}{T}$, that is

$$\ell = O(\log\log(T/\varepsilon)).$$

The number of qubits is then $n \times \mathrm{poly}(\log(T/\varepsilon))$.

It is possible to make this argument rigorous. This yields the Threshold theorem:

Theorem 1.4. *There exists a threshold value p_t with the following property: If the error rate p per physical gate or time step is below p_t, then for any $\varepsilon > 0$, there exists a fault-tolerant protocol such that any logical circuit of size T is mapped to a circuit with* $\mathrm{poly}(T/\varepsilon)$ *times as many qubits, gates and time steps, and the output of the fault-tolerance circuit is correct, except with probability ε.*

It should be noted that while the polylogarithmic factor seems rather small asymptotically, it leads to significant overhead in practice. For instance, applying Shor's algorithm to break cryptographic-size RSA keys requires a few thousands ideal qubits, but billions of physical qubits using the approach outlined above. More efficient approaches have been designed in the past few years, and in particular, it is useful to exploit other families of quantum codes, such as the surface code (see Chapter 3), instead of code concatenation. The advantage is that the code will have better performance, but one then needs to find a fault-tolerant encoding for any circuit that we aim to implement. This typically involves new techniques such as magic state distillation, that we will not cover in this course.

Finally, one important question is whether the space and time overheads of the threshold theorem are necessary. It turns out that using better quantum codes, such as constant-rate quantum LDPC codes can reduce the space overhead to a constant factor [Got14, FGL18]. One drawback of this approach, however, is that the final circuit cannot be parallelized and this typically incurs a linear blowup on the circuit depth. We will discuss these codes in Chapter 3. It is also possible to obtain similar performance while relying on code concatenation, if one uses different codes at each level of concatenation, for instance by taking a family of larger and larger codes that all correct a single error, but such that the overall encoding rate of the code tends to a nonzero constant [YK22]. The advantage is that one achieves constant space overhead and only quasi-polylogarithmic time overhead simultaneously.

Lecture 2

The stabilizer formalism and CSS codes

To go beyond the 9-qubit code of Shor, it is useful to take inspiration from classical coding theory and to try to import as much as possible from the classical setting. In particular, one very convenient class of classical codes is that of linear codes, which correspond to linear subspaces of $\mathbb{F}_2^n$. The class of quantum stabilizer codes devised by Daniel Gottesman in his PhD generalize these codes to the quantum realm.

This lecture will first recall some basic facts about classical coding theory before introducing the stabilizer formalism and defining quantum CSS (Calderbank, Shor, Steane) codes, which are a convient approach to turn two special classical codes into a quantum code.

2.1 Classical linear codes

A linear code C encoding k logical bits in n bits is a subspace of $\mathbb{Z}_2^n = \{0,1\}^n$ of dimension k, i.e. of size 2^k. We usually summarize this information with the notation $[n,k]$.

There are two convenient ways of describing such a code:

(i) as the image of a generator matrix G of size $k \times n$

$$C = \operatorname{Im} G = \{G^T x \ : \ x \in \{0,1\}^k\},$$

(ii) as the kernel of a parity-check matrix H of size $(n-k) \times n$:

$$C = \ker H = \{x \in \{0,1\}^n \ : \ Hx = 0\}.$$

The two matrices satisfy $HG^T = 0$ since any codeword of the form $G^T x$ is in the kernel of H, by definition. The generator matrix is convenient to describe the encoding circuit, the parity-check matrix is convenient to describe the error correction process. Let us immediately note that there are several possible choices of generator matrices and parity-check matrices for a given code. One can also consider a parity-check matrix with redundant equations, so that H has size $m \times n$ with $m \geq n-k$. This provides some redundancy to the syndrome, which is useful in applications to quantum computing since one expects the syndrome measurement procedure to be noisy.

Let $c \in C = \ker H$ be a codeword. If $e \in \{0,1\}^n$ is an error (the ones in e correspond to the bits which have been flipped by the noise), then the *syndrome* of e is defined as

$$H(c+e) = Hc + He = He.$$

The *minimum distance* d of the code C is the minimum weight of a nonzero codeword. It is the minimum number of bits that need to be flipped to map a codeword to another codeword:

$$d = \min_{x \in C \setminus \{0\}} |x|,$$

where $|x|$ is the *Hamming weight* of x, that is the number of 1s in the bitstring x. We say that $\mathcal{C}$ is an $[n, k, d]$ code. Such a code can detect any error of weight $\leq d-1$ and correct any error of weight $\lfloor \frac{d-1}{2} \rfloor$.

We say that a code family $(\mathcal{C}_n)$ with parameters $[n, k, d]$ is *asymptotically good* if $k = \Theta(n), d = \Theta(n)$. While it is fairly easy to show that such families exist for instance by picking a parity-check matrix at random. In fact, one can also obtain good codes with high probability by choosing a sparse parity-check matrix at random. This shows that asymptotically good classical LDPC codes also exist. Constructing explicit families of such codes that can also be decoded efficiently has remained an open question for a long time, until the discovery of *expander codes* by Sipser and Spielman [SS96].

The main application of classical error correcting codes is to protect information in (noisy) communication tasks. The setup is that one has access to a noisy classical channel and the aim is to transmit information faithfully over that channel. A *classical channel* $\mathcal{N} : X \to Y$ maps an input random variable X to an output random variable Y and is defined by a collection of conditional probabilities: $p_{Y|X}(y|x)$, that satisfy

$$\sum_y p_{Y|X}(y|x) = 1, \quad \forall x.$$

Claude Shannon devised a general coding scheme to transmit classical information over such a channel. It consists of

- an initial message $m \in \{0,1\}^k$, a k-bit long string,
- an encoding map $E : \{0,1\}^k \to \{0,1\}^n, m \mapsto x$, which maps the message to a codeword of some code C, usually described by its generator matrix, so that $E(m) = x := G^T m$,
- the encoded message X is sent through the channel $\mathcal{N}$, which is described as a conditional probability distribution: $p(Y = y|X = x)$,
- the output y is fed into a decoder which reads the syndrome Hy and outputs a guess $\hat{m}$ for the message.

The *encoding rate* is defined as the ratio k/n corresponding to the number of *logical* bits sent per *physical bit*.

Define the capacity $\mathcal{C}(\mathcal{N})$ of the channel as

$$\mathcal{C}(\mathcal{N}) = \max_{p_X(x)} I(X;Y)$$

where $I(X;Y) = H(X) + H(Y) - H(XY)$ is the mutual information between the random variables X and $Y = \mathcal{N}(X)$, and the optimization is over probability distributions for variable X.

A simple example is the binary symmetric channel: $X, Y \in \{0,1\}$ and $p_{Y|X}(y|x) = p$ if $y \neq x$, which has capacity $\mathcal{C}(\text{BSC}) = 1 - h(p)$.

Shannon established the following surprising result.

Theorem 2.1 (Channel coding theorem)**.** *One can reliably send information at any rate $k/n < \mathcal{C}(\mathcal{N})$ by exploiting error correcting codes over sufficiently many uses of the channel. The maximal error probability ($\max_m Pr[\hat{m} \neq m]$) goes to 0 when the code size goes to infinity.*

The proof of this theorem is *via* a random coding argument. A major issue is that decoding a random (linear) code is believed to be hard, and to take exponential time. For this reason, one of the main goals of channel coding in the past 70 years has been to devise explicit coding schemes that approach the rate promised by the theorem but such that both the encoding and decoding operations can be performed efficiently.

Rather surprisingly, it took a very long time until these questions were answered in a satisfying way, and one main family of codes that lead to good performance with efficient decoding algorithms is that of Low-Density Parity-Check (LDPC) codes invented by Gallager in his PhD thesis (in the 70s), and mostly forgotten until the mid-90s. We will be extremely interested in their quantum generalization in Chapters 3 and 4.

Before moving to quantum codes, let us define the *dual* of a classical code. The *dual code* of $\mathcal{C}$, denoted $\mathcal{C}^\perp$, is defined as

$$\mathcal{C}^\perp = \{y \in \mathbb{Z}_2^n \ : \ x \cdot y = 0, \forall x \in \mathcal{C}\}.$$

This is an $[n, n-k]$ linear code. Moreover the generator and parity-check matrices of $\mathcal{C}$ and $\mathcal{C}^\perp$ are swapped (up to transposition):

$$G^\perp = H^T, \quad H^\perp = G^T.$$

2.2 Stabilizer codes

We now move to the quantum generalization of classical linear codes. This formalism – *stabilizer codes* – was developed by Daniel Gottesman in his PhD thesis [Got97] in the late 90s.

Let us recall first what the Pauli and Clifford groups are. The single-qubit Pauli group $\mathcal{P}_1$ is the group $\langle i\mathbb{1}, X, Z\rangle$ generated by the Pauli matrices. Its n-qubit generalisation is the n-fold tensor product of $\mathcal{P}_1$, that is $\mathcal{P}_n = \mathcal{P}_1^{\otimes n}$:

$$\mathcal{P}_n = \langle E_1 \otimes E_2 \otimes \ldots \otimes E_n \ : \ E_i \in \mathcal{P}_1\rangle,$$

and has cardinality 4^{n+1}. An important property of Pauli operators is that any two of them either commute or anticommute.

Lemma 2.2. *Let $P, Q \in \mathcal{P}_n$. Then either $PQ = QP$ or $PQ = -QP$.*

Proof. For single-qubit matrices, this is immediate since

$$XY = -YX, \quad XZ = -ZX, \quad YZ = -ZY.$$

Let us write the Pauli operators as products of n single-qubit matrices: $P = P_1 \ldots P_n$ and $Q = Q_1 \ldots Q_n$. Then P and Q commute if and only if they anticommute in an even number of positions. Otherwise, they anticommute. □

The *Clifford group* $\mathcal{C}_n$ is the automorphism group of the Pauli group:

$$\mathcal{C}_n = \{U \in \mathcal{U}(2^n) \ : \ U\mathcal{P}_n U^\dagger = \mathcal{P}_n\},$$

where $\mathcal{U}(2^n)$ is the set of unitary matrices (such that $UU^\dagger = \mathbb{1}$) of size $2^n \times 2^n$. In words, any Pauli operator P is mapped to a Pauli operator *via* conjugation by a Clifford unitary.

Interestingly, the Clifford group can be obtained from a small set of single and two-qubit gates.

Theorem 2.3. *The Clifford group $\mathcal{C}_n$ is generated by H, P and $CNOT$:*

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad P = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Similarly to linear classical codes which are subspaces of $\mathbb{F}_2^n$, a quantum code $\mathcal{Q}$ is a subspace $(\mathbb{C}^2)^{\otimes n}$ of dimension 2^k. It allows one to encode k logical qubits within $n \geq k$ physical qubits. The stabilizer construction is very much inspired by the classical construction of linear codes, and there are two main ways to define a stabilizer code:

(*i*) via its *encoding circuit*: this is a *Clifford unitary* $U \in \mathcal{C}_n$ applied on $|\psi\rangle \otimes |0\rangle^{n-k}$ where $|\psi\rangle \in (\mathbb{C}^2)^{\otimes k}$ is a logical state of k qubits and $|0\rangle^{n-k}$ is an $(n-k)$-qubit ancilla. This gives

$$\mathcal{Q} = \{U|\psi\rangle \otimes |0\rangle^{\otimes(n-k)} \ : \ |\psi\rangle \in (\mathbb{C}^2)^{\otimes k}\}.$$

The encoded state corresponding to $|\psi\rangle$ is often denoted by $|\overline{\psi}\rangle$.

(*ii*) via its *stabilizer* group, *i.e.* a group $\mathcal{S} = \langle g_1, \ldots, g_{n-k}\rangle$ generated by a set of $n-k$ Pauli operators that all commute and that doesn't contain $-\mathbb{1}$. The code is then defined as the elements of the Hilbert space that are *stabilized* by $\mathcal{G}$:

$$\mathcal{Q} = \{|\psi\rangle \in (\mathbb{C}^2)^{\otimes n} \ : \ g_i|\psi\rangle = |\psi\rangle, \forall i \in [n-k]\}.$$

In other words, the code is defined as the $+1$ common eigenspace of all the generators. This space is well defined since the commutation condition ensures that the generators are all codiagonalizable. Moreover, since each generator is a Pauli operator, it has eigenvalues equal to ± 1.

To go from one definition to the other, observe that the states of the form $|\psi\rangle|0\rangle^{\otimes n-k}$ for $|\psi\rangle \in (\mathcal{C}^2)^{\otimes n}$ are stabilized by the generators $Z_{k+1}, \ldots, Z_n$. Indeed, each ancilla qubit in the $|0\rangle$ state is exactly stabilized by the operator Z on that qubit, and it is clear that these operators commute. To deduce the stabilizers of $U|\psi\rangle|0\rangle^{\otimes n-k}$, one simply conjugates the Z_{k+i}'s with the encoding circuit, giving $g_i = UZ_{k+i}U^\dagger$. These are Pauli operators since U is a Clifford operator, and they commute since $g_i g_j = UZ_{k+i}U^\dagger UZ_{k+j}U^\dagger = UZ_{k+i}Z_{k+j}U^\dagger = UZ_{k+j}Z_{k+i}U^\dagger = g_j g_i$.

We see that a stabilizer code is defined by its stabilizer group $\mathcal{S}$, and it is useful to consider the *normalizer* $\mathcal{N}(\mathcal{S})$ of this group.

Definition 2.4 (Normalizer)**.** The normalizer of a subgroup $S \subset \mathcal{P}_n$ in the Pauli group $\mathcal{P}_n$ is the set

$$N(S) = \{g \in \mathcal{P}_n \ : \ gS = Sg\} = \{g \in \mathcal{P}_n \ : \ gSg^{-1} = S\}.$$

Consider some Pauli operator g in the normalizer of the stabilizer group $\mathcal{S}$ of some code. By definition, we have that for any $s \in \mathcal{S}$, there exists $s' \in \mathcal{S}$ such that $sg = gs'$. For any codeword $|\psi\rangle \in \mathcal{Q}$, we have

$$\begin{aligned} sg|\psi\rangle &= gs'|\psi\rangle && \text{since } g \in \mathcal{N}(\mathcal{S}) \\ &= g|\psi\rangle && \text{since } s'|\psi\rangle = |\psi\rangle \text{ for any codeword } |\psi\rangle \text{ and } s' \in \mathcal{S}, \end{aligned}$$

which shows that $g|\psi\rangle$ is also stabilized by all the generators of $\mathcal{S}$ and therefore $g|\psi\rangle \in \mathcal{Q}$. In other words, the operator g maps a codeword to a codeword. We will define a nontrivial logical operator as an operator that acts non identically on the codespace.

Definition 2.5 (Logical Pauli operator)**.** A nontrivial logical Pauli operator of the stabilizer code with stabilizer $\mathcal{S}$ is a Pauli operator that leaves the code globally invariant, but that acts nontrivially on codewords. It is given by the set $N(\mathcal{S}) \setminus \mathcal{S}$, and corresponds to a Pauli operator that commutes with all the generators of $\mathcal{S}$, but that doesn't belong to $\mathcal{S}$.

Note that the elements of the stabilizer group $\mathcal{S}$ act trivially on the codespace since they leave all the codewords invariant, by definition. These operators do not count as nontrivial logical operators.

The fact that the encoding circuit of a stabilizer code is a Clifford operation is particularly useful because it implies that logical Pauli operators correspond to Pauli physical operators. Indeed, suppose we have the encoding $|\overline{\psi}\rangle$ of some state $|\psi\rangle$ and that we wish to obtain the encoding $\overline{P}|\overline{\psi}\rangle$ of $P|\psi\rangle$, for some Pauli operator $|\psi\rangle$. Then we have:

$$\begin{aligned}\overline{P}|\overline{\psi}\rangle &= U(P|\psi\rangle) && \text{by definition of the encoding circuit } U\\ &= (UPU^\dagger)U|\psi\rangle\\ &= (UPU^\dagger)|\overline{\psi}\rangle.\end{aligned}$$

This means that the logical operator $\overline{P}$ is given by $UPU^\dagger$, which is itself a Pauli operator since U is a Clifford operator. This is convenient in practice because it means that one can perform logical Pauli operators very easily on encoded codeword, simply by performing single-gate Pauli operators on the physical qubits.

This also implies that a nontrivial logical Pauli operator will map a word in the quantum code to an orthogonal codeword. This allows us to define the *distance* d of a quantum stabilizer code: it is the minimum weight of a nontrivial logical Pauli operator, that is the number of factors on which the operator doesn't act as the identity:

$$d := \min_{P\in\mathcal{N}(\mathcal{S})\setminus\mathcal{S}} w(P),$$

where the weight $w(P)$ of the Pauli operator $P = P_1 \otimes P_2 \ldots \otimes P_n$ is given by

$$w(P) = |\{i \text{ s.t. } P_i \neq \mathbb{1}_2\}|.$$

We summarize the parameters of a quantum code with the notation $[\![n, k, d]\!]$.

Example 2.6 (Shor's codes)**.** Both the 3-qubit code and Shor's 9-qubit code are stabilizer codes: this is because their encoding circuit is a Clifford unitary. As a consequence, these codes can also be described by their stabilizer group. We have seen that the stabilizer of the 3-qubit code is $\langle Z_1Z_2, Z_2Z_3\rangle$ and that the stabilizer of Shor's 9-qubit code is

$$\langle Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9, X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9\rangle.$$

It is straightforward to check that these groups are Abelian. As expected, these stabilizers admit respectively $2 = 3 - 1$ (three physical qubits and one logical qubit) and $8 = 9 - 1$ (9 physical qubits for a single logical qubit) generators. The Pauli operators $\overline{Z} = Z_1$ and $\overline{X} = X_1X_2X_3$ are logical operators for the 3-qubit code and for Shor's 9-qubit code, respectively. One can immediately check that they commute with all the stabilizers, hence belong to $\mathcal{N}(\mathcal{S})$, but are not in the stabilizer group. This shows that the minimum distance of the 3-qubit code is 1, and that the distance of the 9-qubit code is at most 3. Since we know that the 9-qubit code can correct an arbitrary error on a single qubit, we infer that its distance is at least 3[1], and is therefore equal to 3.

The stabilizer description is particularly useful to correct the errors. The idea is to measure the *syndrome* of the error, that is to measure the eigenvalues of the stabilizer generators for the

[1] since a code with distance 2 would have errors of weight 1 that could not be corrected with certainty

quantum state. The syndrome associated with a Pauli error $E \in \mathcal{P}_n$ is the $(n-k)$-bitstring $s = (s_1, \ldots, s_{n-k})$ defined by

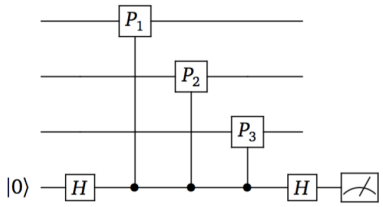
$$s_i := \begin{cases} 0 & \text{if} \quad Eg_i = g_i E, \\ 1 & \text{if} \quad Eg_i = -g_i E. \end{cases}$$

If $s_i = 1$, meaning that the error anticommutes with a stabilizer, then $E|\psi\rangle$ is a -1 eigenvalue of g_i (if $|\psi\rangle$ is a valid codeword):

$$g_i E|\psi\rangle = -Eg_i|\psi\rangle = -E|\psi\rangle.$$

Crucially, the syndrome doesn't depend on the specific codeword, only on the Pauli error. This is similar to the classical case.

Note that it is easy to devise a quantum circuit to measure any Pauli operator, and therefore to measure the syndrome. The following picture for instance depicts a circuit to measure $P_1P_2P_3$ with Pauli operator P_i acting on qubit i:



The circuit above maps the input $|\psi\rangle|0\rangle$ to

$$\to \frac{1}{\sqrt{2}}(|\psi\rangle|0\rangle + |\psi\rangle|1\rangle) \to \frac{1}{\sqrt{2}}(|\psi\rangle|0\rangle + P_1P_2P_3|\psi\rangle|1\rangle) \to \frac{1}{\sqrt{2}}(|\psi\rangle|+\rangle + P_1P_2P_3|\psi\rangle|-\rangle),$$

that is

$$\frac{1}{2}(\ |\psi\rangle + P_1P_2P_3|\psi\rangle)|0\rangle + \frac{1}{2}(\ |\psi\rangle - P_1P_2P_3|\psi\rangle)|1\rangle.$$

Measuring the last register in the computational basis projects the state onto the eigenspace of $P_1P_2P_3$ corresponding to eigenvalue $+1$ if the measurement outcome is 0, and eigenvalue -1 if the measurement outcome is 1.

One should note that this circuit is not the one that will be preferred in practice. The reason is that it is not fault tolerant: indeed, if a single error occurs on the ancilla system at the beginning of this circuit, then this error may propagate to all the data qubits, which is something we wish to avoid at all cost. One possible approach to solve this issue consists in replacing the ancilla qubit by a multiqubit cat state $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ and use each wire to perform a single controlled-P_i gate. In that way, a single error on an ancilla qubit cannot propagate to more than a single error in the data qubits.

We saw previously how to describe a coding scenario to transmit information over a noisy channel in the classical case. The same can be done in the quantum case, with the difference that the noisy channel is now described by a quantum channel, and that it can sometimes correspond to an imperfect quantum memory, in the sense that the goal is not necessarily to send quantum information from some party A to another distant party B, but more modestly to protect quantum information from decoherence, for instance if one wants to use it to perform some quantum computation.

The coding scheme is then described as follows:

- the initial message is a k-qubit quantum state $|\psi\rangle \in (\mathbb{C}^2)^{\otimes k}$,

- it is mapped to an n-qubit state by applying an encoding operator $\mathcal{U}$ applied to $|\psi\rangle \otimes |0\rangle^{n-k}$ (i.e., state and ancilla), where $\mathcal{U}$ is the Clifford operation encoding the state into the quantum code $\mathcal{Q}$,
- the n-qubit state goes through a noisy channel (communication channel, storage device, interaction with environment, etc) described by a *completely positive trace preserving (CPTP) map* $\mathcal{N}$. Alternatively, the channel is a unitary interaction between the codeword and the environment,
- a decoder with access to the output of the channel measures the syndrome s of the error, thus projecting the error onto a Pauli error E, outputs a guess $\hat{E}$ for the error, and applies the correction $\hat{E}^\dagger$ to the state.

An important difference with classical coding is that in the quantum case, the "error" is not uniquely defined, and errors differing by an element of the stabilizer group are *equivalent*, since they act exactly in the same way on all codewords. This phenomenon is called *degeneracy*. For this reason, a quantum error correction procedure will generally not aim at recovering the "true" error that occurred, but rather the equivalence class of this error. While this problem looks simpler (since we do not need to find the exact error), it turns out that this subtle difference with the classical scenario leads in fact to complications when devising efficient decoding algorithms.

2.3 CSS codes

Now that we know the definition of a stabilizer code, we want to find examples of such codes. The difficulty is to find an interesting stabilizer group. Recall that such a group is defined *via* a set of generators, which are Pauli operators that commute. This commutation requirement is not so easy to fulfill in general, especially if we are interested in quantum codes with sparse generators, each acting nontrivially only on a constant number of qubits.

The CSS codes (for the name of their inventors, Calderbank, Shor, Steane [CS96, Ste96]) form a convenient subclass of all stabilizer codes where the generators of the stabilizer group are either products of Pauli-X or products of Pauli-Z. Said otherwise, they do not involve Pauli-Y operators, nor can they consist of products of X and Z operators. This is an appealing restriction because the commutativity condition between the generators now needs to be checked only between X-type and Z-type generators, since both X-type generators and Z-type generators obviously commute among themselves. In that case, both types of generators are described by binary words (with 1s at the coordinates corresponding to an X or Z-type operator). The two sets of generators are therefore associated to two classical codes and the commutation requirement on the generators immediately translates to some simple orthogonality conditions between these two codes.

The CSS construction works as follows. One picks two classical codes $\mathcal{C}_0 = \ker H_X$ and $\mathcal{C}_1 = \ker H_Z$ of length n with the property that

$$H_X H_Z^T = 0.$$

From this, one can define two types of generators: X-type generators are products of X and $\mathbb{1}$ corresponding to the rows of H_X, and Z-type generators are products of Z and $\mathbb{1}$ corresponding to rows of H_Z:

$$g_i^X := \bigotimes_{j\,:\,(H_X)_{ij}=1} X_j, \qquad g_i^Z := \bigotimes_{j\,:\,(H_Z)_{ij}=1} Z_j.$$

It is easy to check that $H_X H_Z^T = 0$ implies that the two types of generators commute. Note that $\mathcal{C}_1^\perp$, the dual code of $\mathcal{C}_1$, is the code spanned by the rows of H_Z, and $H_X H_Z^T$ is equivalent

to $\mathcal{C}_1^\perp \subseteq \mathcal{C}_0$, and similarly $\mathcal{C}_0^\perp \subseteq \mathcal{C}_1$.

The subspace stabilized by all these generators is a stabilizer code, denoted $\mathrm{CSS}(\mathcal{C}_0, \mathcal{C}_1)$. It has n physical qubits and encodes k logical qubits, with

$$k = n - \operatorname{rk} H_X - \operatorname{rk} H_Z.$$

One can define two distances d_0 and d_1 as follows, which are the minimum weights of Z-type Pauli operators (or X-type Pauli operators) that act nontrivially on the code, and that are not corrected by $\mathcal{C}_0$ and $\mathcal{C}_1$:

$$d_0 = \min\{\ |x| \ :\ x \in \mathcal{C}_0 \setminus \mathcal{C}_1^\perp\}, \qquad d_1 = \min\{\ |x| \ :\ x \in \mathcal{C}_1 \setminus \mathcal{C}_0^\perp\}.$$

Indeed, an element of $\mathcal{C}_1^\perp$ is simply a sum of rows of H_Z and is an element of the stabilizer group of the code. The *minimum distance* of $\mathrm{CSS}(\mathcal{C}_0, \mathcal{C}_1)$ is given by

$$d = \min(d_0, d_1).$$

It is important to note that d_0 can be much larger than the distance of $\mathcal{C}_0$! This can be the case if $\mathcal{C}_1^\perp$ contains words of small weight, *i.e.*, if the CSS code admits low-weight generators. In particular, we say that a quantum code is a *low-density parity-check* (LDPC) code if $\mathcal{C}_0$ and $\mathcal{C}_1$ are both classical LDPC codes, that is, if H_X and H_Z are sparse matrices. In that case, since $\mathcal{C}_1^\perp \subseteq \mathcal{C}_0$ contains words of constant weight, namely the rows of H_Z, we have

$$\operatorname{dist}(\mathcal{C}_0) = O(1),$$

but we wish to find quantum LDPC codes with an unbounded minimum distance. In fact, we will see in Chapter 4 that there exist quantum LDPC codes with $d = \Omega(n)$!

Given two classical codes $\mathcal{C}_0$ and $\mathcal{C}_1$ as above, it is possible to find a basis of the logical operators of the corresponding CSS code.

Lemma 2.7. *A* CSS code $\mathrm{CSS}(\mathcal{C}_0, \mathcal{C}_1)$ *is defined from two classical linear codes* $\mathcal{C}_0$, $\mathcal{C}_1$ *of parameters* $[n, k_0, d_0]$ *and* $[n, k_1, d_1]$, *such that* $\mathcal{C}_1^\perp \subseteq \mathcal{C}_0$. *The quantum code has parameters* $[[n, k_0 + k_1 - n]]$ *and is spanned by the vectors*

$$|x_j + \mathcal{C}_0^\perp\rangle := \frac{1}{|\mathcal{C}_0^\perp|} \sum_{y \in \mathcal{C}_0^\perp} |x_j + y\rangle,$$

where x_j *is an element of the quotient space* $\mathcal{C}_1/\mathcal{C}_0^\perp$. *In other words, they satisfy* $x_i + x_j \notin \mathcal{C}_0^\perp$, *for any pair* $x_i \neq x_j$.

The proof will be treated in exercise.

Codewords of the CSS code have the form $|x + \mathcal{C}_1^\perp\rangle$ where $x \in \mathcal{C}_0$ and two codewords $|x + \mathcal{C}_1^\perp\rangle$ and $|x' + \mathcal{C}_1^\perp\rangle$ differ if and only if x and x' belong to different cosets of $\mathcal{C}_1^\perp$ in $\mathcal{C}_0$.

An example of CSS code is Steane's 7-qubit code, where we take $\mathcal{C}_0 = \mathcal{C}_1$ to be the $[7, 4, 3]$ Hamming code with generator and parity-check matrices

$$G^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

This is a valid CSS code since $HH^T = 0$. It encodes a single logical qubit. One can construct general Hamming codes by taking all possible words (except the all-zero word) for the columns of the parity-check matrix. These generalized codes have parameters $[2^m - 1, 2^m - m - 1, 3]$ and can tolerate one bit-flip since they all have distance 3.

Taking $x_0 = 0000000$ and $x_1 = 1111111$ as representatives of $\mathcal{C}_0/\mathcal{C}_1^\perp$, and enumerating the elements of $\mathcal{C}_1^\perp = \mathrm{Im}(H)$ as

$$\mathcal{C}_1^\perp = \{0000000, 0001111, 0110011, 0111100, 1010101, 1011010, 1100110, 1101001\},$$

we obtain the logical qubits as

$$\begin{aligned}|\bar{0}\rangle = \frac{1}{2\sqrt{2}}(&|0000000\rangle + |0001111\rangle + |0110011\rangle + |0111100\rangle + |1010101\rangle \\ &+ |1011010\rangle + |1100110\rangle + |1101001\rangle), \\ |\bar{1}\rangle = \frac{1}{2\sqrt{2}}(&|1111111\rangle + |1110000\rangle + |1001100\rangle + |1000011\rangle + |0101010\rangle \\ &+ |0100101\rangle + |0011001\rangle + |00101101\rangle).\end{aligned}$$

This illustrates one of the main strengths of the stabilizer formalism: in general, the logical qubits are given by very long expressions (a superposition over an exponential number of basis states), and the generators of the stabilizer yield a much more efficient description of the code.

The generators of the stabilizer can be chosen to be the rows of H, for both the X and Z-type generators:

$$\begin{aligned} X_4X_5X_6X_7, \quad X_2X_3X_6X_7, \quad X_1X_3X_5X_7, \\ Z_4Z_5Z_6Z_7, \quad Z_2Z_3Z_6Z_7, \quad Z_1Z_3Z_5Z_7. \end{aligned}$$

One can find logical operators of weight 3 for the Steane code, which implies that Steane's 7-qubit code is a $[[7, 1, 3]]$ quantum code.

We now describe an explicit error correction strategy for a CSS code.

Error correction for a CSS code.

The general strategy is as usual: measure the syndrome and apply a correction. For CSS codes, the syndrome is naturally divided into two parts: X-type errors are corrected with the syndrome of the Z-type generators, and Z-type errors are corrected with the syndrome of the X-type generators. This suggests a two-part procedure:

- **X-type errors:** (i) compute the syndrome for code $\mathcal{C}_1$ (i.e. for generators of the form Z^e for e a row of H_Z): this is the reversible operation
$$|y\rangle|0\ldots0\rangle \mapsto |y\rangle|s_0(y)\rangle,$$
with $s_0(y) = H_Z y$, the syndrome of y with respect to code $\mathcal{C}_1$; (ii) measure the syndrome, and (iii) correct for bit-flips by applying Pauli-X corrections;
- **swapping between codes:** apply a Hadamard transform to every qubit;
- **Z-type errors:** same procedure as before but for the code $\mathcal{C}_0$, i.e. generators of the form X^f for f a row of H_X;
- **returning to the initial code:** apply again a Hadamard transform to every qubit.

Let us verify that this decoding procedure correctly recovers the codeword provided that the weight of the error is less than t, where t is the maximum number such that both $\mathcal{C}_0$ and $\mathcal{C}_1$ can correct t errors[2].

[2]The parameter t of a classical code is related to the minimum distance via $d = 2t + 1$.

Consider a Pauli error $X^v Z^w$ with bit strings $v, w \in \{0,1\}^n$ applied to a codeword $\sum_j \alpha_j |x_j + \mathcal{C}_1^\perp\rangle$. The state becomes

$$X^v Z^w \sum_j \alpha_j |x_j + \mathcal{C}_1^\perp\rangle = \sum_j \alpha_j (-1)^{v\cdot w} Z^w |x_j + v + \mathcal{C}_1^\perp\rangle$$

where we used that $X^v Z^w = (-1)^{v\cdot w} Z^w X^v$.

The first step of the correction procedure corrects X-type errors by computing the syndrome relative to $\mathcal{C}_1$. This will yield the syndrome of v for $\mathcal{C}_1$, and provided that $|w| \leq t$, the procedure will return v and apply the Pauli correction X^v, giving,

$$\sum_j \alpha_j (-1)^{v\cdot w} Z^w |x_j + v + \mathcal{C}_1^\perp\rangle = Z^w \sum_j \alpha_j |x_j + \mathcal{C}_1^\perp\rangle.$$

After the Hadamard transformation, the state becomes

$$\begin{aligned}
H^{\otimes n} Z^w \sum_j \alpha_j |x_j + \mathcal{C}_1^\perp\rangle &= X^w H^{\otimes n} \sum_j \alpha_j |x_j + \mathcal{C}_1^\perp\rangle && \text{(since } H^{\otimes n} Z^w = X^w H^{\otimes n}\text{)} \\
&= \sum_j \alpha_j X^w H^{\otimes n} X^{x_j} |\mathcal{C}_1^\perp\rangle \\
&= \sum_j \alpha_j X^w H^{\otimes n} X^{x_j} H^{\otimes n} |\mathcal{C}_1\rangle && \text{(proven in exercise)} \\
&= \sum_j \alpha_j X^w Z^{x_j} |\mathcal{C}_1\rangle.
\end{aligned}$$

Using the same argument as before, one can compute the syndrome relative to $\mathcal{C}_0$ and correct the error X^w. This will work correctly provided that $|w| \leq t$, where t is a lower bound on the correction capacity of $\mathcal{C}_1$. Undoing the Hadamard transform then returns the original codeword.

We note that in a real experiment, one would likely directly measure the two types of generators directly, without performing a transversal Hadamard to all the qubits to switch from X-type measurements to Z-type measurements.

2.4 Correcting arbitrary errors with a stabilizer code

In general, the error channel need not be a Pauli channel. This is often used in the literature as a simplification, but more general errors do and will occur in the lab. For instance, it may happen that the qubits are affected by a coherent noise, of the form $U = e^{i\varepsilon X}$ for all qubits, with some small value of ε. In that case, the noise per qubit is very small, but all qubits are affected by the noise. Then how can we be sure that the codes we have described will handle such errors?

Consider a very general noise model given by a noise channel $\mathcal{N}$,

$$\rho \quad \longrightarrow \quad \mathcal{N}(\rho) = \sum_k N_k \rho N_k^\dagger,$$

with $\sum_k N_k^\dagger N_k = \mathbb{1}$.

One can decompose each of the Kraus operators in the Pauli basis:

$$N_k = \sum_\ell \lambda_{k\ell} P_\ell,$$

for Pauli operators P_ℓ, where ℓ corresponds for instance to a bit string of length $2n$ describing the Pauli operator for each of the n qubits. This gives

$$\mathcal{N}(\rho) = \sum_{k,\ell,m} \lambda_{k,\ell} \lambda_{k,m}^* P_\ell \rho P_m.$$

Now, each Pauli error P_ℓ gives a specific syndrome $s(P_\ell)$, and we wish to model how the syndrome measurement takes place. Let us describe the stabilizer group by a set of generators $g_1, \ldots, g_m$, which are commuting Pauli operators. Recall that the syndrome $s(P) = s_1(P), \ldots, s_m(P)$ associated to the Pauli operator P is an m-bit string defined as

$$s_i(P) := \begin{cases} 0 & \text{if} \quad Pg_i = g_i P, \\ 1 & \text{if} \quad Pg_i = -g_i P. \end{cases}$$

The circuit that performs the measurement of s_i involves an ancilla qubit:

$$|\psi\rangle|0\rangle_i \quad \mapsto \quad \frac{1}{2}(\mathbb{1} + g_i)|\psi\rangle|0\rangle_i + \frac{1}{2}(\mathbb{1} - g_i)|\psi\rangle|1\rangle_i.$$

For $s \in \{0,1\}^m$, let us define the operator Π_s as follows:

$$\Pi_s = \frac{1}{2^m} \prod_{i=1}^{m} (\mathbb{1} + (-1)^{s_i} g_i).$$

The previous circuit applied to $\mathcal{N}(\rho)$ prepares the state

$$\sum_{s,t} \sum_{k,\ell,m} \lambda_{k,\ell} \lambda^*_{k,m} \Pi_s P_\ell \rho P_m \Pi_t \otimes |s\rangle\langle t|.$$

Let us see how Π_s and P_ℓ commute:

$$\begin{aligned} \Pi_s P_\ell &= \frac{1}{2^m} \prod_{i=1}^{m} (\mathbb{1} + (-1)^{s_i} g_i) P_\ell \\ &= P_\ell \frac{1}{2^m} \prod_{i=1}^{m} (\mathbb{1} + (-1)^{s_i + s_i(P_\ell)} g_i) \end{aligned}$$

by definition of $s_i(P_\ell)$. This gives

$$\Pi_s P_\ell = P_\ell \Pi_{s + s(P_\ell)}.$$

Now, since the initial state ρ belongs to the code space by assumption, and thus lies in the support of the projector Π_0, we have that

$$\Pi_s P_\ell \rho P_m \Pi_t = P_\ell \Pi_{s+s(P_\ell)} \rho \Pi_{t+s(P_m)} P_m = P_\ell \rho P_m \delta_{s,s(P_\ell)} \delta_{t,s(P_m)}.$$

This means that the state has evolved to

$$\sum_{k,\ell,m} \lambda_{k,\ell} \lambda^*_{k,m} P_\ell \rho P_m \otimes |s(P_\ell)\rangle\langle s(P_m)|.$$

Measuring the register containing the classical description syndrome of the error, one obtains some value s. This projects the state onto

$$\sum_s \sum_{k,\ell,m \text{ s.t. } s(P_\ell)=s(P_m)=s} \lambda_{k,\ell} \lambda^*_{k,m} P_\ell \rho P_m \otimes |s\rangle\langle s|.$$

Given some syndrome s, the decoder will propose a candidate Pauli correction P_e compatible with this syndrome, that is, such that $s(P_e) = s$. After applying this correction, the final state ρ' reads

$$\rho' = \sum_s \sum_{k,\ell,m \text{ s.t. } s(P_\ell)=s(P_m)=s} \lambda_{k,\ell} \lambda^*_{k,m} P_{\ell+e} \rho P_{m+e}.$$

Now, suppose that the decoder can correct all Pauli errors from a set $\mathcal{E}$. If the decoder is optimal, it can for instance correct all the Pauli errors of weight $\leq (d-1)/2$. This means that only errors outside $\mathcal{E}$ are problematic:

$$\rho' = \Big(\sum_s \sum_{k,\ell,m \text{ s.t. } s(P_\ell)=s(P_m)=s,(\ell,m)\in\mathcal{E}^2} \lambda_{k,\ell}\lambda_{k,m}^*\Big)\rho + \sum_s \sum_{k,\ell,m \text{ s.t. } s(P_\ell)=sP_(m)=s,(\ell,m)\notin\mathcal{E}^2} \lambda_{k,\ell}\lambda_{k,m}^* P_{\ell+e}\rho P_{m+e}.$$

For a general noise model, the second term in the right-hand side is nonzero, so we cannot expect perfect recovery of the initial state. However, in the realistic case where $\lambda_{k,\ell}$ rapidly goes to zero for large-weight Pauli error, the final state will be very close to the original one, and the standard assumptions of quantum fault tolerance are usually of this form.

And of course, if all the errors from the set $\mathcal{E}$ only involve Pauli terms with weight $\leq (d-1)/2$, then an ideal decoder will correct such errors perfectly. This is for instance the case of Shor's 9-qubit code if all the errors (possibly not Pauli errors) involve at most a single qubit.

It should be emphasized that the analysis above makes at least one unrealistic assumption, namely that the circuit that performs the syndrome extraction is noiseless. In the lab, this circuit will also consist of imperfect gates which may introduce additional errors that prevent the exact recovery of the initial state. The theory of fault-tolerant quantum computing aims at addressing this specific difficulty, and indeed there are known solutions to deal with the fact that all the physical operations are noisy in the lab. One possible approach is to consider quantum LDPC codes. In that case, because the generators have constant weight, the circuit that extracts the syndrome has a constant depth, which prevents too many errors to occur. Developing a full scheme that is fault tolerant is possible with quantum LDPC codes [Got14], but is beyond the scope of these lecture notes.

Lecture 3

Quantum LDPC codes: the toric code and hypergraph product codes

One can recap the requirements of a CSS code by the need to find two matrices H_X and H_Z such that every row of H_X is orthogonal to every row of H_Z, that is such that

$$H_X H_Z^T = 0.$$

For practicality issues, one also would like both matrices to be sparse: each row and column only contains a small (constant, independent of n) number of 1s. This for instance allows experimentalists to easily measure the syndrome while relying on a constant-depth circuit, an important feature if one assumes that such syndrome extraction circuits are noisy. Combining both requirements is not completely obvious, and one needs to consider specific structured constructions in order to satisfy both conditions. An area of mathematics where such objects appear rather naturally is topology, and it was a brilliant insight of Alexei Kitaev to notice this.

3.1 The toric code

At the end of the 90s, Kitaev showed that cellullations of surfaces (and of higher-dimensional manifolds) gave a very general method to derive CSS codes, with parameters depending on the properties of the surface. The most famous example is the *toric code*, which can be realized by taking a square cellullation of a torus.

Consider an $N \times N$ square grid on a torus, and put a qubit on each of the $2N^2$ edges[1]. We define a CSS code by choosing the following generators of weight 4:

- *plaquette* operators: for each plaquette p on the grid, define $g_p^X := \bigotimes_{e \in \partial p} X_e$, where $e \in \partial p$ means that edge e belongs to the boundary of plaquette p,
- *star* operators: for each vertex v in the grid, define $g_v^Z := \bigotimes_{e \sim v} Z_e$, where $e \sim v$ means that edge e is incident to vertex v.

Let us immediately verify that these generators commute: for this, it is enough to notice that a vertex and a plaquette operator either do not overlap, or else overlap in exactly 2 positions.

There are N^2 vertices on the grid and N^2 plaquettes, so we have defined $2N^2$ generators. Note, however, that these generators are not independent since the product of all vertex operators

[1] In practice, the qubits need not lie on a true torus. Rather, the picture of the torus is helpful to see how to connect the different qubits, or how to measure the syndrome. Nevertheless, it is nice if one can put the qubits on a 2-dimensional surface, as one can rely on short-range gates. To do this, an interesting variant of the toric code called the *surface code* puts the qubits on a 2d grid, with boundary. Two types of boundaries are required, rough and smooth. The only drawback compared to the original toric code is that such a surface code only protects a single logical qubit instead of 2.

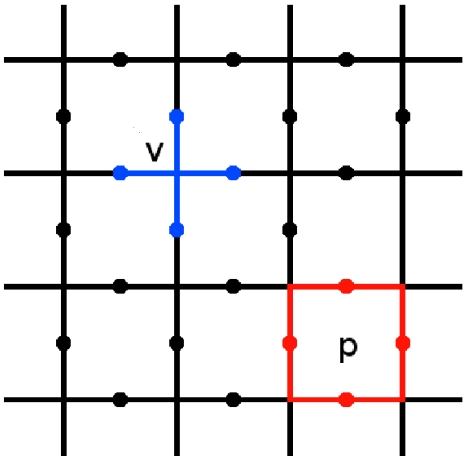


Figure 3.1: Local structure of the toric code: qubits are placed on edges, vertex operators are the product of X operators applied to the 4 neighboring qubits of a vertex, plaquette operators are the product of Z operators applied to the 4 qubits on the boundary of a plaquette (By James Wootton, https://commons.wikimedia.org/w/index.php?curid=11823316)

is the identity, and the product of all plaquette operators is also the identity (this is because every qubit, *i.e.* every edge, belongs to two plaquettes and to two stars):

$$\bigotimes_p g_p^X = \mathbb{1}, \quad \bigotimes_v g_v^Z = \mathbb{1}.$$

These are the only nontrivial relations, meaning that there are $2N^2 - 2$ independent generators for $2N^2$ qubits, which yields 2 logical qubits.

From our earlier definition of CSS codes, we need two classical codes $\mathcal{C}_0 = \ker H_X$ and $\mathcal{C}_1 = \ker H_Z$, with $\mathcal{C}_1^\perp \subseteq \mathcal{C}_0$:

- the code $\mathcal{C}_0$ is the *cycle code* of the grid: the support of codewords corresponds to a cycle, *i.e.* its boundary is zero;
- the code $\mathcal{C}_1^\perp$ is generated by words whose support is the boundary of a set of plaquettes.

The inclusion $\mathcal{C}_1^\perp \subset \mathcal{C}_0$ follows from the fact that the boundary of a boundary is always zero:

$$\partial\partial = 0.$$

This relation is at the heart of all topological/homological quantum error correcting code constructions. Importantly, to get a nontrivial quantum code, it is crucial that there exist elements in $\mathcal{C}_0 \setminus \mathcal{C}_1^\perp$, that is cycles which are not a boundary. Loops around the torus are exactly of this form, and will give the support of logical operators on the code space.

CSS codes from algebraic topology. More formally, one can define the following *chain complex*:

$$C_2 = \mathbb{F}_2^{N^2} \xrightarrow{\partial_1} C_1 = \mathbb{F}_2^{2N^2} \xrightarrow{\partial_0} C_0 = \mathbb{F}_2^{N^2}$$

where C_2, C_1, C_0 are vector spaces corresponding respectively to the spaces of plaquettes (Z-generators), edges (qubits) and vertices (or star, X-generators), and such that

$$\partial_0 \circ \partial_1 = 0.$$

A *chain complex* is a sequence of linear subspaces $(C_i)_i$ together with linear maps $\partial_i : C_{i+1} \to C_i$, called *boundary operators*, such that $\partial_{i-1}\partial_i = 0$, for all i.

The classical codes of the CSS construction are given by

$$\mathcal{C}_0 = \ker \partial_0, \quad \mathcal{C}_1^\perp = \operatorname{Im} \partial_1.$$

The parity-check matrices associated with these two codes can be chosen to be

$$H_X = \operatorname{Mat}(\partial_0), \qquad H_Z = \operatorname{Mat}(\partial_1^T),$$

corresponding to the matrices of the operators ∂_0 and ∂_1^T in some chosen basis of the spaces C_2, C_1, C_0 and expresses the maps ∂_0 and ∂_1 with respect to these bases. We immediately observe that the relation $\partial_0 \partial_1 = 0$ is equivalent to $H_X H_Z^T = 0$.

In this language, the space of Z-logical operators is $(\ker \partial_0)/(\operatorname{Im} \partial_1)$, which is, by definition, the *first homology group of the complex*. In order to study the X-type logical operators, one can consider the co-complex, where the boundary operators are the transposed operators of the boundary operators of the complex:

$$C_0 = \mathbb{F}_2^{N^2} \xrightarrow{\partial_0^T} C_1 = \mathbb{F}_2^{2N^2} \xrightarrow{\partial_1^T} C_2 = \mathbb{F}_2^{N^2}.$$

This is again a valid complex since $\partial_0^T \circ \partial_1^T = 0$.

Every such chain complex of length 2 gives rise to a CSS code.

Logical operators of the toric code. In order to describe the logical qubits of the toric code, we need to understand the equivalence classes of $\mathcal{C}_0/\mathcal{C}_1^\perp$, that is, the cycles that are not a boundary. There are indeed two inequivalent families of such cycles, corresponding to the two types of loop around the torus. These cycles are *homologically nontrivial* meaning that they cannot be deformed (by addition of boundary) to yield the zero cycle. For this reason, the toric code is an example of *topological code*: properties of the quantum code result from the topology of the underlying manifold. In fact, the toric code is given by a specific cellullation of the torus, that is, a decomposition of the torus in plaquettes. The standard toric code uses square plaquettes but one could choose other types of plaquettes, for instance triangles.

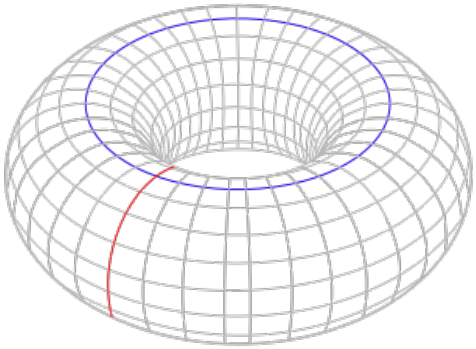


Figure 3.2: Local structure of the toric code: qubits are placed on edges, vertex operators are the product of X operators applied to the 4 neighboring qubits of a vertex, plaquette operators are the product of Z operators applied to the 4 qubits on the boundary of a plaquette (Image from Wikimedia Commons)

In particular, since the minimum size of a nontrivial cycle is N, we deduce that the minimum distance of the code is also N, and the parameters of the toric code read:

$$[[2N^2, 2, N]].$$

Another particularly interesting feature of the toric code is that it is an example of *low-density parity-check (LDPC)* code, meaning that each generator only involves a constant number of qubits (4 for the toric code) and that each qubit is only involved in a constant number of generators (4 again for the toric code). This LDPC condition is particularly important when it comes to experimental implementation since measuring the syndrome for the toric code will only require small circuits involving at most 4 physical (data) qubits for each bit of the syndrome. In fact, the leading approaches to build a quantum computer are based on the toric code (or its cousin the surface code).

Despite an intensive study of quantum LDPC codes, it turned out to be extremely difficult to find better LDPC codes than the toric codes. For about 20 years, the best bound for the minimum distance was $n^{1/2}\log^{1/4} n$. In 2020, a series of papers showed that much better distances are in fact possible. First, Hastings, Haah and O'Donnell showed how to get $d_{\min} \approx n^{3/5}$, and then Panteleev and Kalachev achieved a distance $n/\log n$. Finally, in November 2021, Panteleev and Kalachev proved the existence of asymptotically good quantum LDPC codes with dimension $k = \Theta(n)$ and distance $d_{\min} = \Theta(n)$. We will discuss some examples of codes better than the toric code in Chapter 4.

Decoding the toric code.

Let us first consider X-type errors. Their associated syndrome is obtained via the Z-type generators, corresponding to vertices. Let X^E be an X-type error with support on the set E. Then the syndrome of X^E is given by the set of vertices in the *boundary of* E:

$$s(X^E) = \partial E.$$

In order to decode, one must therefore find an error with the appropriate syndrome, that is to find a pattern of small weight with a given boundary, such that this pattern differs from the true error by a sum of generators: they should differ by a boundary. A particularly popular decoder is the *minimum weight perfect matching* (MWPM) algorithm. This is not optimal, however, since the most probable error isn't necessarily the error of minimum weight, but rather the error whose equivalence class is the most probable. However, the minimum weight perfect matching algorithm performs quite well against random errors. A drawback is that its complexity scales like the cube of the number of qubits, *i.e.* like N^6. While this is efficient in the computer science sense, that is polynomial time, this is far from fast, and alternative decoders have been devised which have a complexity scaling linearly with the number of qubits, but slightly worse performance, meaning that they will not handle physical error rates as large as MWPM does.

In order to address Z-type errors, it is convenient to exploit *Poincaré duality*: the dual of the cellulation looks exactly like the initial cellulation, but with the roles of vertices and plaquettes exchanged. In other words, one can correct Z-type errors with the same approach as explained for the X-type errors, but working in the dual cellulation.

The *threshold* of the toric code is about $10-11\%$ for the depolarizing channel, corresponding to an error model where X, Y, Z errors occur independently with probability $p/3$ and no error occurs with probability $1-p$. This means that for error rates below the threshold, increasing the code size will result in better (lower) logical error rate (after optimal decoding). In other words, if the noise level is below the threshold, one can in principle achieve arbitrary good protection simply by sufficiently increasing the code size.

Topological codes beyond the toric code. This approach isn't limited to tessellations of the torus, but can be generalized in a straightforward way to cellullations of arbitrary closed manifolds in arbitrary dimensions. For 2-dimensional constructions, which are interesting from an experimental point of view, Bravyi, Poulin and Terhal [BPT10] have showed that the parameters always satisfy:

$$kd^2 = O(n).$$

This is quite far from the ideal upper bound that would scale like n^3.

One can also change the geometry and work with hyperbolic geometry rather than Euclidean space. In particular, hyperbolic geometry is very useful to get codes with a large dimension. It is then possible to define quantum codes with a linear dimension $k = \Theta(n)$ and with a polynomial distance $d_{\min} = \Omega(n^{1/5})$, which is not possible in Euclidean geometry.

3.2 Hypergraph product codes

Finding quantum LDPC codes for which d grows significantly faster than $\sqrt{n}$ has remained completely opened for more than 20 years. A simpler task was to improve the rate k/n of the code without decreasing the distance below $\sqrt{n}$. The main new idea in this direction was the hypergraph product code construction due to Jean-Pierre Tillich and Gilles Zémor [TZ14].

CSS codes from algebraic topology. Following the idea of the topological quantum code construction described above, we can see that a CSS code is in general given by a chain complex of length 3:

$$C_2 = \mathbb{F}_2^{m_Z} \quad \xrightarrow{\partial_1} \quad C_1 = \mathbb{F}_2^{n} \quad \xrightarrow{\partial_0} \quad C_0 = \mathbb{F}_2^{m_X}$$

where C_2, C_1, C_0 are vector spaces corresponding respectively Z-generators, physical qubits and X-generators, and such that

$$\partial_0 \circ \partial_1 = 0.$$

The classical codes of the CSS construction are given as before by

$$\mathcal{C}_0 = \ker \partial_0, \quad \mathcal{C}_2 = \operatorname{Im} \partial_1.$$

In this language, the space of Z-logical operators is again $H_1 = (\ker \partial_0)/(\operatorname{Im} \partial_1)$.

Tensor product of chain complexes. A natural way to obtain a chain complex of length 3 is to take the tensor product of two chains complexes of length 2:

$$(C_1 \xrightarrow{\partial_C} C_0) \quad \otimes \quad (D_1 \xrightarrow{\partial_D} D_0).$$

This gives a new chain complex

$$(C_1 \otimes D_1) \quad \xrightarrow{\partial_1} \quad (C_0 \otimes D_1 + C_1 \otimes D_0) \quad \xrightarrow{\partial_0} \quad (C_0 \otimes D_0),$$

with

$$\begin{aligned} \partial_1 &:= (\partial_C \otimes \mathrm{id}_D, \mathrm{id}_C \otimes \partial_D) \\ \partial_0 &:= \mathrm{id}_C \otimes \partial_D + \partial_C \otimes \mathrm{id}_D. \end{aligned}$$

It is immediate to check that

$$\begin{aligned} \partial_0 \circ \partial_1 &= (\mathrm{id}_C \otimes \partial_D + \partial_C \otimes \mathrm{id}_D)(\partial_C \otimes \mathrm{id}_D, \mathrm{id}_C \otimes \partial_D) \\ &= \partial_C \otimes \partial_D + \partial_C \otimes \partial_D \\ &= 0 \end{aligned}$$

since we work over $\mathbb{F}_2$.

Note that a chain complex of length 2 is nothing but a classical linear code, where the boundary operator plays the role of the parity check matrix. This means that starting with two arbitrary classical codes (specified by their parity-check matrices), one obtains a quantum CSS code: this is the hypergraph product construction. Note that the terminology of hypergraph product code construction comes from the initial intuition behind the construction: the idea was to take the product of the Tanner graphs (which can themselves be represented as hypergraphs) of two classical codes.

For the construction to be well defined, it is important to describe the two classical codes using their parity-check matrices. Consider two codes $\ker H$ and $\ker H'$ with parameters $[n, k, d]$ and $[n', k', d']$, and let us also define their transpose codes $\ker H^T$ and $\ker H'^T$ with parameters $[n-k, k^T, d^T]$ and $[n'-k', k'^T, d'^T]$.

Theorem 3.1. *The hypergraph product code obtained from the parity-check matrices H and H' has the following parameters*

$$[\![nn' + (n-k)(n'-k'), kk' + k^T k'^T, \min(d, d', d^T, d'^T)]\!].$$

The toric code as an hypergraph product code. A first example of hypergraph product code is obtained by taking the constituant codes to be a repetition code with an $n \times n$ cyclic parity-check matrix. In this case, we get $n = n'$, $k = k' = k^T = k'^T = 1$ and $d = d' = d^T = d'^T = n$. One exactly recovers the parameters of the toric code $[\![2n^2, 2, n]\!]$.

Hypergraph product code of two good LDPC codes. Consider two good classical codes, that is $k, k', d, d' = \Theta(n)$. If the parity-check matrices are chosen to be full rank, then $k^T = k'^T = 0$ and we get the corresponding minimum distances $d^T, d'^T = \infty$. Then the resulting CSS code has parameters:

$$[\![\Theta(n^2), \Theta(n^2), \Theta(n)]\!].$$

In other words, one keeps the same distance than the toric code, but the number of logical qubits is now linear in the length instead of being constant!

Decoding hypergraph product codes. Given the remarkable parameters of the construction and its versatility, it is natural to ask whether it is possible to decode hypergraph product codes efficiently. Several solutions exist and try to mimic the decoding of classical LDPC codes. One solution is particularly efficient and comes with a decoder that can be parallelized to work in logarithmic depth: the *small-set-flip decoder* of quantum expander codes [LTZ15]. Such codes even offer the possibility of quantum fault-tolerance with only a *constant space overhead* instead of a polylogarithmic overhead as promised by concatenation [FGL18].

Lecture 4

Good quantum LDPC codes

As we have seen in the previous lecture, hypergraph product codes provide a very general construction of quantum LDPC codes with fairly good code parameters. The main remaining open question is whether the rather poor distance, scaling as $\sqrt{n}$, can be improved at all. It turned out to be extremely nontrivial to significantly beat this bound. A topological construction due to Freedman, Luo and Meyer [FML02], gave a code with a constant number of logical qubits and a distance $\Theta(\sqrt{n}\log^{1/4} n)$. This bound remained the best known one for more than 15 years until ideas from high-dimensional expansion could improve the logarithmic factor, to $\sqrt{n}\log^c n$, for an arbitrary constant c [KT21]. In 2020, Hastings, Haah and O'Donnell proposed a *fiber bundle* code construction achieving parameters $[\![n, n^{4/5}, n^{3/5}]\!]$, which finally broke the square-root bound for the distance [HHO21]. Soon after that, Breuckmann and Eberhardt introduced a generalisation of the hypergraph product code known as the *balanced product* construction [BE21], and conjectured that this construction could be compatible with good quantum LDPC codes. This conjecture was established by Panteleev and Kalachev [PK22]. While this construction is rather technical, we will focus here on a closely related construction called *quantum Tanner codes* [LZ22], that is conceptually simpler and also provides a family of good quantum LDPC codes. Before delving in the construction, it is useful to make a detour *via* the classical setting and to describe classical Tanner codes.

4.1 Classical LDPC codes: Tanner codes and expander codes

Before diving in the quantum case, let us first consider classical LDPC codes. These codes were defined by Gallager in his thesis in the 60s then largely forgotten until the mid-90s. They are defined by a sparse parity-check matrix H of size $(n-k)\times n$ as

$$\mathcal{C} = \ker H = \{x \in \{0,1\}^n \ : \ Hx = 0\},$$

where sparsity means that each column and row of the binary matrix H only contains a constant (independent of n) number of 1s. One reason to care about the LDPC condition is that one can design efficient decoding algorithms for these codes. Classical LDPC codes have found a lot of applications, and are one of the two main codes families used in 5G for instance (together with polar codes).

In the classical setting, it is fairly easy to prove the existence of good LDPC codes. This can be done by a probabilistic argument: simply pick a sparse parity-check matrix H of size $tn \times n$ at random, for some fixed value $t \in (0,1)$. With high probability, the matrix will have full rank, and the code will have constant rate $r = 1-t$, that is, dimension $k = tn$. To show that the code has a linear minimum distance, one can simply count the number of parity-check matrices leading to a sublinear distance and show that they only form a negligible fraction of all possible sparse random matrices. See the textbook by Richardson and Urbanke for details [RU08].

Finding explicit constructions of good LDPC codes, that can in addition be efficiently decoded, required more work. The first step was a general construction due to Tanner to turn a large graph and a small code into a large code [Tan81]. The second step occurred much later when Sipser and Spielman noticed that applying this recipe to a special class of graphs known as expander graphs would yield good LDPC codes [SS96].

4.1.1 Tanner codes

Consider some Δ-regular graph $\mathcal{G} = (V, E)$ with vertex set V and edge set E. Here, Δ-regular means that each vertex is incident to Δ edges. The main idea of Tanner is to construct a code by putting bits on the edges and linear constraints on the vertices. This can be achieved if each vertex enforces some linear constraint corresponding to a "small" code of length Δ. Let $\mathcal{C}_0 = [\Delta, \rho\Delta, \delta\Delta]$ be such a linear code. One can index the Δ edges incident to a vertex $v \in V$ and, for any word $x \in \mathbb{F}_2^E$, consider the restriction x_v of the word to the coordinate corresponding to the neighborhood of v in the graph.

The Tanner code $\mathrm{Tan}(\mathcal{G}, \mathcal{C}_0)$ associated to the graph $\mathcal{G} = (V, E)$ and the *local code* $\mathcal{C}_0$ (together with a choice of indexing of each edge-neighborhood) is defined as

$$\mathrm{Tan}(\mathcal{G}, \mathcal{C}_0) = \{x \in \mathbb{F}_2^E \ : \ x_v \in \mathcal{C}_0, \forall v \in V\}.$$

In order to simplify things later on, it will be convenient to consider the double cover $\mathcal{G}_2$ of the graph $\mathcal{G}$, which is the bipartite graph with vertex set $V_0 \cup V_1$, corresponding to two copies of V, and edges given by $(v, 0) \sim (w, 1)$ and $(v, 1) \sim (w, 0)$ for any edge $v \sim w$ in the original graph.

It is easy to lower bound the dimension of the code by simply counting the number of constraints. There are $2|V| \times (1-\rho)\Delta$ constraints (which are not necessarily linearly independent) and $n = |V|\Delta$ bits (corresponding to edges in the double cover). The number of constraints is $2(1 - \rho)n$, and therefore the dimension of the code satisfies

$$k \geq (2\rho - 1)n. \tag{4.1}$$

We see that we need the rate of each small code to be at least $1/2$ to guarantee a positive rate for the Tanner code.

4.1.2 Product codes and their dual

The lower bound on the rate is not tight in general, and a good example of this is given by product codes. These can be interpreted as Tanner codes defined on the complete bipartite graph $K_{n,n}$. Note that each edge is defined by a couple (i, j) with $i, j \in [n]$. Recall that defining a Tanner code requires to choose a labeling of the edges around each vertex. The idea is to choose the natural indexing such that the n edges incident to vertex $i \in V_0$ are labeled by (i, j) for $j \in V_1$ and the n edges incidence to vertex $j \in V_1$ are labeled by (i, j) for $i \in V_0$.

Now, pick two classical codes C_0 and C_1 of length n, rates ρ_0 and ρ_1, and distances $\delta_0 n, \delta_1 n$. The *product code* $C_0 \otimes C_1$ is defined as the set of words represented as $n \times n$ binary matrices $X \in \mathbb{F}_2^{n \times n}$ such that

$$X_{i\cdot} \in C_1, \forall i, \quad X_{\cdot j} \in C_0, \forall j.$$

In other words, each column of X is a codeword of C_0 and each row of X is a codeword of C_1.

It is a standard result that the parameters of $C_0 \otimes C_1$ are $[n^2, \rho_0\rho_1 n^2, \delta_0\delta_1 n^2]$, and in particular the dimension is much larger than the one promised by the lower bound of (4.1). In particular, the rate is positive provided both ρ_0 and ρ_1 are positive. The dimension is easy to see by noticing that the generating matrix of $C_0 \otimes C_1$ is simply $G_0 \otimes G_1$, if G_i is the generating matrix of C_i.

It will also be useful to introduce the dual codes of product codes. Recall that the *dual code* $C^\perp$ of a code C is the set of words orthogonal to every codeword in C:

$$C^\perp := \{\ x \in \mathbb{F}_2^n\ :\ \langle x, y\rangle = 0 \forall y \in C\},$$

where the inner product is defined over $\mathbb{F}_2$ as $\langle x, y\rangle = \sum_{i=1}^n x_i y_i \mod 2$. The dimension of the dual code is given by

$$\dim C^\perp = n - \dim C.$$

Let us call *dual product code* the dual of the product code $C_0 \otimes C_1$. It is given by

$$(C_0 \otimes C_1)^\perp = C_0^\perp \otimes \mathbb{F}_2^n + \mathbb{F}_2^n \otimes C_1^\perp.$$

Codewords of the dual product code correspond to sums of column codewords of $C_0^\perp$ and row codewords of $C_1^\perp$. Since product codes have a larger dimension than naively expected (by counting constraints), we deduce that dual product codes will typically have a smaller dimension than expected. In particular, if C_0 and C_1 are nontrivial then the dual product code will not coincide with the whole space. A notable consequence is that any codeword $x \in (C_0 \otimes C_1)^\perp$ can be written as a sum of a column word $c \in C_0^\perp \otimes \mathbb{F}_2^n$ and a row codeword $r \in \mathbb{F}_2^n \otimes C_1^\perp$ in a non-unique fashion. Indeed, any such decomposition $c + r$ can be replaced by $c' + r'$ with

$$c' = c + u, \quad r' = r + u,$$

with $u \in C_0^\perp \otimes C_1^\perp$. In particular, for any such x, one can ask for a decomposition of smallest weight, where we define a notion of weight for row codewords (resp. column codewords counting the number of rows (resp. columns) that are non zero:

$$\|c\| = \text{number of nonzero columns in } c, \quad \|r\| = \text{number of non zero rows in } r.$$

If dual product codes are significantly worse than product codes in terms of distance (since their distance is the minimum of the distances of $C_0^\perp$ and $C_1^\perp$, and scales at best like the square-root of the codelength), they can make up for it in terms of *robustness*, which basically says that if a codeword x has a low Hamming weight, then it can be decomposed as $x = c + r$, where both $\|c\|$ and $\|r\|$ are small. Very recently, [DHLV22, KP22] showed that if C_0 and C_1 are chosen randomly with some fixed rate, then their tensor product is κ-robust for some constant κ independent of the code length Δ. This means that for any *dual tensor codeword* $x \in (C_0 \otimes C_1)^\perp$, there exist $c \in C_0^\perp \otimes \mathbb{F}_2^n, r \in \mathbb{F}_2^n \otimes C_1^\perp$ such that $x = c + r$ and

$$|x| \geq \kappa n(\|c\| + \|r\|). \tag{4.2}$$

This property will be central in the analysis of the good quantum LDPC constructions.

4.1.3 Expander codes

See the excellent lecture notes of Guruswami's course for a great exposition [Gur10].

We recall that we are interested in constructing explicit LDPC codes. It makes sense to apply the Tanner construction to a graph with small degree rather than $K_{n,n}$, so that we can use local codes of small size. Let us consider a bipartite graph $\mathcal{G}$, say the double cover of some Cayley graph $\mathrm{Cay}(G, A)$ defined for a group G and a generator set A (symmetric). The vertices of this graph are two copies of the elements of G and we put edges as follows:

$$(g, 0) \sim (ag, 1), \qquad (g, 1) \sim (ag, 0).$$

We will consider a group of size n with $|A| = \Delta$, so that the graph $\mathcal{G}$ admits $2n$ vertices and Δn edges.

While we are looking for a graph of small degree, we do not want to lose the good connectivity properties of the complete bipartite graph, and a possible approach is to take an expander graph. This amounts to say that the nontrivial eigenvalues of the adjacency matrix of the graph are bounded by some $\lambda \ll \Delta$. In particular, if one chooses a Ramanujan graph, which is possible by picking the right group G and generators A, then $\lambda \leq 2\sqrt{\Delta}$.

An important consequence of the expansion property is the *expander mixing lemma*, which states that small subgraphs of $\mathcal{G}$ cannot have a large average degree.

Lemma 4.1 (Expander mixing lemma)**.** *Let $\mathcal{G} = (V_0 \cup V_1, E)$ be an $n \times n$ Δ-regular λ-expander graph. Then for any $S \subseteq V_0, T \subseteq V_1$, the number of edges $E(S,T)$ between S and T satisfies*

$$|E(S,T)| \ \leq \lambda\sqrt{|S| \cdot |T|} + \frac{\Delta}{n}|S| \cdot |T|. \tag{4.3}$$

We will be interested in the regime where S and T both have linear size, but are sufficiently small so that the second term of (4.3) is smaller than the first. In that case, we see that

$$|E(S,T)| = O(\sqrt{\Delta |S||T|}).$$

If we denote by d_S and d_T the average degree of vertices in S and T of the subgraph induced by S and T, we also have that

$$|E(S,T)| = d_S|S| = d_T|T| = \sqrt{d_S d_T}\sqrt{|S||T|}.$$

Together with the previous equation, this shows that

$$d_S d_T = O(\Delta),$$

which means that the subgraph induced by S and T cannot be too dense, unless S and T are large.

We can exploit this property to show that the distance of expander codes is large. An *expander code* is a Tanner code defined on an expander graph $\mathcal{G}$ with local codes $C_0 = [\Delta, \rho\Delta, \delta\Delta]$.

Consider a nonzero codeword x of this expander code. It induces a subgraph of $\mathcal{G}$, with vertex set $S \cup T$. If $|x|$ is not too large, then the expander mixing lemma implies that the average degrees satisfy $d_S d_T = O(\Delta)$. However, x must satisfy the local constraints at each vertex, which means that each vertex of $S \cup T$ must be incident to at least $\delta\Delta$ edges, which implies that $d_S, d_S \geq \delta\Delta$. This is a contradiction.

In more detail, (4.3) gives

$$\sqrt{|S||T|} \geq (\delta - \frac{\lambda}{\Delta})n,$$

and $|x| \geq \delta\Delta\sqrt{|S||T|}$ finally implies that

$$|x| \geq \delta(\delta\Delta - \lambda)n.$$

The same expansion properties can also be exploited to give a very simple parallel decoding algorithm that can correct any errors of weight $\leq \alpha n$ in logarithmic time. The idea is very simple and one alternates between decoding steps on V_0 and V_1. For each step, each vertex v flips the bits on incident edges so as to obtain the closest codeword in $\mathcal{C}_0$. Again, the lecture notes of [Gur10] give a very clear exposition of the classical case.

4.2 Quantum LDPC codes, Codes from Left-Right Cayley complexes

Here, we would like to exploit the Tanner code formalism to define quantum LDPC codes. It should be clear that one cannot directly apply the same recipe as in the classical case and associate qubits with edges of a graph and constraints (generators) with vertices. The reason is that two generators would either not share any qubit, or share only one (if the qubit corresponds to the edge between the two qubits). But then these two generators would not commute if they are of different type. The solution is to upgrade the graph to a complex of larger dimension, for instance a complex with vertices, edges and faces of dimension 2.

4.2.1 The (rotated) toric code

The first construction of quantum LDPC codes is the toric code proposed by Kitaev, and follows this strategy. It generalizes the Tanner code construction of the repetition code. In this chapter, we focus on a slightly different version of the toric code compared to the previous chapter. This version is sometimes called the *rotated toric code* and will be easier to generalise to obtain good quantum LDPC codes.

One considers a grid of $2L \times 2L$ squares, which correspond to qubits, with periodic boundary conditions (to get the topology of a torus). Then one considers a 2-coloring of the vertices. Red vertices are associated with X-type generators, and blue vertices with Z-type generators. Each generator has support on the 4 incident squares. It is clear that an X and a Z generator have supports that either don't intersect or that intersect on two squares, which implies their commutation. The parameters of this code are

$$n = 4L^2, k = 2, d = 2L = \sqrt{n}.$$

This construction really is a generalization of the repetition code on a circle, which explains that it has a small dimension but a rather good distance. It is surprising that one cannot get a better distance immediately. In fact, it turns out that the toric code has optimal parameters among all the quantum codes that can be defined on a 2D surface. Getting much better parameters requires to work with an expander complex.

4.2.2 The left-right Cayley complex

Such an expander complex is a square complex, that is an combinatorial object with vertices, edges and *squares*, that generalizes the complex of the toric code. We will rely on a specific complex called *left-right Cayley complex* that appears implicitly in the work of Panteleev and Kalachev [PK22] and can also be thought of as a quadripartite version of the left-right Cayley complex of Dinur *et al.* [DEL+22]. It is an incidence structure between a set V of vertices, two sets of edges E_A and E_B, that we will refer to as A-edges and B-edges, and a set Q of squares. The vertex-set V is partitioned into four subsets $V = V_{00} \cup V_{01} \cup V_{10} \cup V_{11}$, corresponding to four copies[1] of a fixed group G, that is, $V_{ij} = G \times \{i, j\}$. We also have two self-inverse subsets $A = A^{-1}$ and $B = B^{-1}$ of the group G and assume for simplicity that A and B are of the same cardinality Δ. For $i \in \{0, 1\}$, two vertices $v = (g, i0) \in V_{i0}$ and $v' = (g', i1) \in V_{i1}$ are related by an A-edge if $g' = ag$ for some $a \in A$. Similarly, for $j \in \{0, 1\}$, vertices $v = (g, 0j)$ and $v' = (g', 1j)$ are related by a B-edge if $g' = gb$ for some $b \in B$. The sets E_A and E_B make up the set of A-edges and B-edges respectively and define two graphs $\mathcal{G}_A = (V, E_A), \mathcal{G}_B = (V, E_B)$,

[1] This quadripartite version of the complex is the generalization of the double cover of a graph.

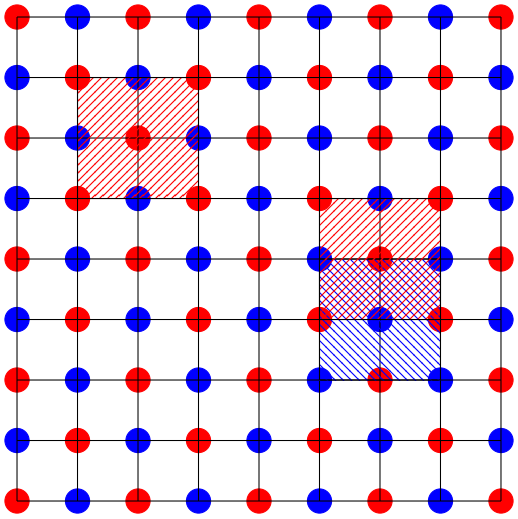


Figure 4.1: Rotated toric code. Qubits are on the squares. Red and blue vertices correspond to the two types of generators. An X and a Z-generator can overlap, but always on 2 squares.

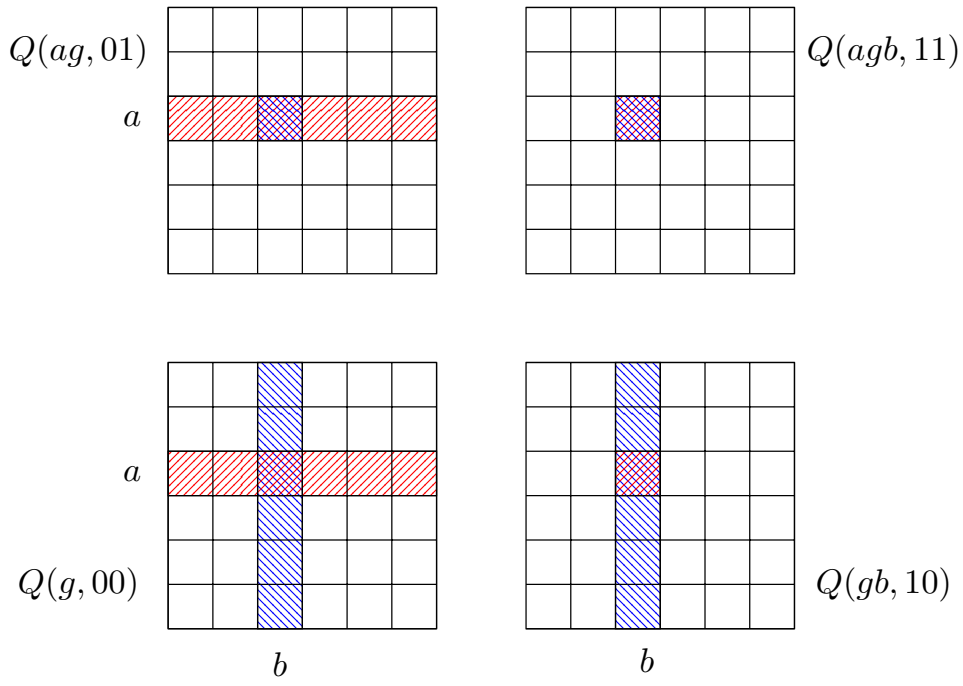


Figure 4.2: The four Q-neighbourhoods $Q(v)$ that contain the square $\{(g, 00), (ag, 01), (agb, 11), (gb, 10)\}$ depicted in red and blue. The Q-neighbourhoods of two vertices connected by an A-edge (resp. a B-edge) share a row depicted in red (resp. a column in blue). The labeling is chosen to ensure that a given square, such as the one in red and blue, is indexed similarly, by (a, b) here, in the four Q-neighbourhoods. The X-type generators are codewords of $C_A \otimes C_B$ in the Q-neighbourhoods of $V_{00} \cup V_{11}$; the Z-type generators are codewords of $C_A^\perp \otimes C_B^\perp$ in the Q-neighbourhoods of $V_{01} \cup V_{10}$. They automatically commute since their support can only intersect on a shared row or column (as depicted), and the orthogonality of the local codes ensure that they commute on this row or column.

each of which consists of two disjoint copies of the double cover of a Cayley graph over the group

G (with generator set A for $\mathcal{G}_A$, and B for $\mathcal{G}_B$). Next, the set Q of squares is defined as the set of 4-subsets of vertices of the form $\{(g, 00), (ag, 01), (agb, 11), (gb, 10)\}$, with the four vertices belonging to distinct copies of G.

If we restrict the vertex set to $V_0 := V_{00} \cup V_{11}$, every square is now incident to only two vertices: one in V_{00} and one in V_{11}. The set of squares can then be seen as a set of edges on V_0, and it therefore defines a bipartite graph that we denote by $\mathcal{G}_0^{\square} = (V_0, Q)$. Similarly, the restriction to the vertices of $V_1 := V_{01} \cup V_{10}$ defines the graph $\mathcal{G}_1^{\square}$, which is an exact replica of $\mathcal{G}_0^{\square}$: both graphs are defined over two copies of the group G, with $g, g,' \in G$ being related by an edge whenever $g' = agb$ for some $a \in A, b \in B$. For any vertex v, we denote by $Q(v)$ the Q-neighbourhood of v defined as the set of squares incident to v. The Q-neighbourhood $Q(v)$ has cardinality Δ^2 and is isomorphic to the product set $A \times B$: the situation is illustrated on Fig. 4.2.

One of the features of this complex is that the Q-neighborhoods of two vertices in $V_{00} \cup V_{11}$ and $V_{10} \cup V_{01}$ either don't overlap or overlap on a single column or row. In particular, we can exploit that to associate X-type generators to vertices of $V_{00} \cup V_{11}$ and Z-type generators to vertices of $V_{10} \cup V_{10}$. Then to ensure that two such generators commute, it is sufficient to show that their support are orthogonal on each row or column. This can be done very conveniently with the help of product codes.

4.3 Quantum Tanner codes are asymptotically good

4.3.1 Quantum Tanner codes

Quantum Tanner codes are quantum CSS codes formed by two classical Tanner codes $\mathcal{C}_0$ and $\mathcal{C}_1$ with support on the set Q of squares of a square complex as above. The CSS construction [Ste96, CS96] requires both codes to satisfy the orthogonality condition $\mathcal{C}_0^{\perp} \subset \mathcal{C}_1$. To this end, we define local codes on the space $\mathbb{F}_2^{A \times B}$ that we may think of as the space of matrices whose rows (columns) are indexed by A (by B). If $C_A \subset \mathbb{F}_2^A$ and $C_B \subset \mathbb{F}_2^B$ are two linear codes, we define the *tensor* (or product) code $C_A \otimes C_B$ as the space of matrices x such that for every $b \in B$ the column vector $(x_{ab})_{a \in A}$ belongs to C_A and for every $a \in A$ the row vector $(x_{ab})_{b \in B}$ belongs to C_B. Recalling that the dual $C^{\perp}$ of a code $C \subset \mathbb{F}_2^{\Delta}$ is the set of words orthogonal to all words in C, we define $\mathcal{C}_0$ and $\mathcal{C}_1$ to be the classical Tanner codes $\mathcal{C}_0 := \mathrm{Tan}(\mathcal{G}_0^{\square}, (C_A \otimes C_B)^{\perp})$ and $\mathcal{C}_1 := \mathrm{Tan}(\mathcal{G}_1^{\square}, (C_A^{\perp} \otimes C_B^{\perp})^{\perp})$, with bits associated to each square of Q and local constraints enforced at the vertices of V_0 and V_1, respectively. To check the orthogonality condition between the two codes, it is convenient to look at their generators (or parity-checks). We define a X-generator for $\mathcal{C}_0$ (resp. a Z-generator for $\mathcal{C}_1$) as a vector of $\mathbb{F}_2^Q$ whose support lies entirely in the Q-neighbourhood $Q(v)$ of V_0 (resp. V_1), and which is equal to a codeword of $C_A \otimes C_B$ (resp. $C_A^{\perp} \otimes C_B^{\perp}$) on $Q(v)$. The Tanner code $\mathcal{C}_0$ (resp. $\mathcal{C}_1$) is defined as the set of vectors orthogonal to all X-generators (resp. Z-generators). The commutation between both types of generators follows from the fact that if a X-generator on $v_0 \in V_0$ and a Z-generator on $v_1 \in V_1$ have intersecting supports, then v_0 and v_1 must be neighbours in the left-right Cayley complex and their Q-neighbourhoods must intersect on either a column or a row, on which the two generators equal codewords of C_A and $C_A^{\perp}$, or of C_B and $C_B^{\perp}$ (see Fig. 4.2). Since Δ is chosen constant with respect to n, we see that the generators of the code have constant weight and that each qubit only appears in a constant number of generators: the resulting quantum Tanner code is therefore a quantum LDPC code by definition.

4.3.2 Bound on the dimension

Choosing C_A and C_B of rates ρ and $1-\rho$, so that both $C_A \otimes C_B$ and $C_A^\perp \otimes C_B^\perp$ have rate $\rho(1-\rho)$, yields a quantum code with encoding rate $k/n \geq (1-2\rho)^2$, as can be seen by counting the number of generators of the code. Indeed, we have

- $\Delta^2|G|$ qubits,
- $\rho(1-\rho)\Delta^2$ generators for every vertex, so $4\rho(1-\rho)\Delta^2|G|$ generators in total.

The rate of the code is

$$k/n \geq 1 - \frac{4\rho(1-\rho)\Delta^2|G|}{\Delta^2|G|} = (1-2\rho)^2.$$

In particular, it is always positive unless $\rho = 1/2$.

The toric code is a specific example where $\Delta = 2$, G is the product of two cyclic groups,

$$G = \mathbb{Z}/(2L\mathbb{Z}) \times \mathbb{Z}(2L\mathbb{Z}), \qquad A = \{(1,0),(-1,0)\}, \qquad B = \{(0,1),(0,-1)\},$$

and the codes C_A and C_B are simply the repetition codes of length 2, $C_A = C_B = \{00, 11\}$. In particular, their rate is $\rho = 1/2$, so the lower bound above doesn't guarantee a positive rate.

4.3.3 Bound on the distance

We will sketch the proof that quantum Tanner codes obtained from an Ramanujan left-right Cayley complex with robust local codes are asymptotically good. More precisely, their distance satisfies

$$d_{\min} \geq \frac{\delta^2\kappa^2}{256\Delta} n,$$

and this bound is tight with respect to Δ since there exist logical errors of weight $\leq n/\Delta$. A full proof of this fact is given in [LZ22].

The proof is somewhat similar to the classical case, but needs to be adapted. Note indeed that the local codes (dual product codes) are not good, and only have distance $\Theta(\Delta)$ while the graph has degree Δ^2. This implies that the classical argument doesn't work directly since the expansion parameter is $\Theta(\Delta)$ and in fact smaller than the distance of the local codes. It is not surprising that the argument cannot work since the codes $\mathcal{C}_0$ and $\mathcal{C}_1$ do in fact contain words of constant weight, namely the generators of the quantum code. The proof of the distance of the quantum code must therefore exploit something beyond the distance of the local codes. This role is played by the robustness of the local codes, which says that the codewords have a specific structure.

The sketch of the proof is as follows:

- We pick the support of a nontrivial logical operator $x \in \mathcal{C}_1 \setminus \mathcal{C}_0^\perp$. We want to show that $|x| = \Omega(n)$, which implies that $d_1 = \Omega(n)$. By symmetry, the same argument will show that $d_0 = \Omega(n)$. We choose this case as it allows us to work with C_A and C_B instead of $C_A^\perp$ and $C_B^\perp$, which lightens the notations a little bit. We do not work directly with the Hamming weight $|x|$, but with its norm $\|x\|$ that we define now.
- We show that x can be decomposed as $x = C_0 + C_1 + R_0 + R_1$ (don't confuse C_0, C_1 which are specific vectors with the codes $\mathcal{C}_0, \mathcal{C}_1$ of the CSS construction). We pick a minimal representation of x, which locally minimizes the norm

$$\|x\| := \|C_0\| + \|C_1\| + \|R_0\| + \|R_1\|.$$

- We define the sets of vertices $S_{ij} \subseteq V_{ij}$ appearing in the subcomplex induced by x, as well as the sets of exceptional vertices S^e_{ij}, for which it holds that $\|c_v\| + \|r_v\| \geq \delta^2\Delta/256$. Provided that the error weight is not too large, we show that

$$|S^e_{ii}| = O\Big(\frac{|S_0|}{\Delta^2}\Big), \qquad |S^e_{i\bar{i}}| = O\Big(\frac{|S_1|}{\Delta^2}\Big).$$

  This uses expansion (the expander mixing lemma) in the graph $\mathcal{G}^\square$ and robustness to say that each exceptional vertex has weight $\Theta(\Delta^2)$.
  For instance, if we want to prove the bound for S^e_{00}, one should consider the subgraph induced by $R_0 + C_0$ in $\mathcal{G}^\square_0$. The minimality of the decomposition and the robustness show that

$$|E(S^e_{00}, S_{11})| \geq \alpha\kappa\Delta^2|S^e_{00}|,$$

  with $\alpha := \delta^2/256$. The expander mixing lemma then implies

$$\alpha\kappa\Delta^2|S^e_{00}| \leq |E(S^e_{00}, S_{11})| \leq 4\Delta\sqrt{|S^e_{00}||S_{11}|} + \frac{\Delta^2}{|V_{00}|}|S^e_{00}||S_{11}|.$$

- The crucial part of the argument is to remark that ordinary columns/rows (i.e., those that do not belong to exceptional vertices) agglutinate in a small number of vertices $|T| = O\Big(\frac{S_{\bar{i}j}}{\Delta}\Big)$. This only uses expansion in $\mathrm{Cay}(G, A)$ and $\mathrm{Cay}(G, B)$. For concreteness, we consider the set T_{00}, defined as the vertices of V_{00} such that the Q-neighborhood share a non-zero column vector (of x) with an ordinary vertex of V_{10}. This is a *heavy* edge. The key idea is to look at a graph isomorphic to $\mathcal{G}_A = Cay(g, A)$ induced by x over the vertices of $V_{00} \cup V_{01}$. This is done by collapsing every row onto a single square which becomes an edge in this graph. One can consider the subgraph induced by x and apply the expander mixing lemma on T and its neighborhood:

$$|T|\frac{\delta\Delta}{2} \leq |E(T, S_{01})| \leq 2\sqrt{\Delta}\sqrt{|T||S_{01}|} + \frac{\Delta}{|V_{01}|}|T||S_{01}|.$$

- We conclude as follows: assuming that there are indeed few exceptional vertices, it means that almost all vertices are ordinary, and therefore that many rows or columns are ordinary. They cluster on a set of vertices of size $|S|/\Delta$, meaning that the average weight of x in these vertices is $\Theta(\Delta^2)$. This shows that there are a constant fraction of vertices in T which are exceptional. But T cannot be too small since a single vertex cannot contain more than Δ ordinary rows or columns. We get that $\Omega(|S|/\Delta)$ exceptional vertices, a contradiction.

To summarize,

- the classical proof needs two ingredients: expansion implies that a small subgraph has an average degree $O(\sqrt{\Delta})$, but the local constraints impose that the degree is $\Omega(\Delta)$, contradiction.
- the quantum proof is similar: expansion in $\mathcal{G}^\square$ implies that a small subgraph has only very few vertices of large degree, but the local constraints together with expansion in $\mathcal{G}^{A/B}$ imply the existence of many vertices of large degree. Contradiction!

4.3.4 Follow up and open questions

Similarly to the classical case, a little more work gives us an efficient decoding algorithm – either a sequential linear-time algorithm, or a logarithmic-time parallel decoder – that can correct arbitrary errors of weight $\leq \alpha n$, for some $\alpha > 0$. This decoder is very likely single-shot, meaning that it can still work well even when the syndrome is noisy. This is crucial for application to

quantum computing since real hardware is always noisy and syndrome extraction is not different.

There are many open questions remaining:

- Can we find good quantum LDPC of reasonable size? The construction outlined here, and that of Panteleev-Kalachev are both very asymptotic.
- Can we perform logical gates in a fault-tolerant fashion with such good LDPC codes?
- The bound of the minimum distance of quantum Tanner codes also proves that the codes defined by Dinur et al [DEL$^+$22] are locally testable. It is interesting to ask whether quantum locally testable codes exist and whether ideas similar to those outlined here can help to construct such objects.

Bibliography

[ABO97] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 176–188, 1997.

[BE21] Nikolas P Breuckmann and Jens N Eberhardt. Balanced product quantum codes. *IEEE Transactions on Information Theory*, 67(10):6653–6674, 2021.

[BPT10] Sergey Bravyi, David Poulin, and Barbara Terhal. Tradeoffs for reliable quantum information storage in 2d systems. *Phys. Rev. Lett.*, 104:050503, Feb 2010.

[CS96] Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54:1098–1105, Aug 1996.

[DEL+22] Irit Dinur, Shai Evra, Ron Livne, Alexander Lubotzky, and Shahar Mozes. Locally testable codes with constant rate, distance, and locality. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 357–374, New York, NY, USA, 2022. Association for Computing Machinery.

[DHLV22] Irit Dinur, Min-Hsiu Hsieh, Ting-Chun Lin, and Thomas Vidick. Good quantum LDPC codes with linear time decoders. *arXiv preprint arXiv:2206.07750*, 2022.

[FGL18] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. Constant overhead quantum fault-tolerance with quantum expander codes. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 743–754. IEEE Computer Society, 2018.

[FML02] Michael H Freedman, David A Meyer, and Feng Luo. Z_2-systolic freedom and quantum codes. In *Mathematics of quantum computation*, pages 303–338. Chapman and Hall/CRC, 2002.

[Got97] Daniel Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, California Institute of Technology, 1997.

[Got14] Daniel Gottesman. Fault-tolerant quantum computation with constant overhead. *Quantum Information & Computation*, 14(15-16):1338–1372, 2014.

[Got22] Daniel Gottesman. Opportunities and challenges in fault-tolerant quantum computation. *arXiv preprint arXiv:2210.15844*, 2022.

[Gur10] Venkatesan Guruswami. Expander codes and their decoding. https://www.cs.cmu.edu/~venkatg/teaching/codingtheory/notes/notes8.pdf, 2010.

[HHO21] Matthew B Hastings, Jeongwan Haah, and Ryan O'Donnell. Fiber bundle codes: breaking the $n^{1/2}$ polylog(n) barrier for quantum LDPC codes. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1276–1288, 2021.

[KP22] Gleb Kalachev and Pavel Panteleev. Two-sided robustly testable codes. *arXiv preprint arXiv:2206.09973*, 2022.

[KT21] Tali Kaufman and Ran J Tessler. New cosystolic expanders from tensors imply explicit Quantum LDPC codes with $\Omega(\sqrt{n}\log^c n)$ distance. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1317–1329, 2021.

[LTZ15] Anthony Leverrier, Jean-Pierre Tillich, and Gilles Zémor. Quantum expander codes. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 810–824. IEEE Computer Society, 2015.

[LZ22] Anthony Leverrier and Gilles Zémor. Quantum Tanner codes. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 872–883. IEEE, 2022.

[PK22] Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical LDPC codes. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 375–388, 2022.

[RU08] Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge university press, 2008.

[SS96] Michael Sipser and Daniel A Spielman. Expander codes. *IEEE transactions on Information Theory*, 42(6):1710–1722, 1996.

[Ste96] Andrew Steane. Multiple-particle interference and quantum error correction. *Proc. R. Soc. Lond. A*, 452(1954):2551–2577, 1996.

[Tan81] R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[TZ14] Jean-Pierre Tillich and Gilles Zémor. Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Trans. Inf. Theory*, 60(2):1193–1202, 2014.

[YK22] Hayata Yamasaki and Masato Koashi. Time-efficient constant-space-overhead fault-tolerant quantum computation. *arXiv preprint arXiv:2207.08826*, 2022.