More advanced codes

The Shor code was the first general-purpose quantum error-correcting code, but since then many others have been discovered. An important example, discovered independently of the Shor code, is the seven-bit Steane code:

\begin{align*}
|0\rangle & \rightarrow \frac{1}{\sqrt{8}} \left( |0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\
& \quad + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle \right), \\
|1\rangle & \rightarrow \frac{1}{\sqrt{8}} \left( |1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\
& \quad + |1100000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle \right).
\end{align*}
Stabilizers of the Steane Code

The basis vectors for the Steane Code are rather long and unwieldy. It is actually easier to specify the code in terms of its *stabilizer generators*:

\[
\begin{align*}
\hat{g}_1 &= \hat{I}\hat{I}\hat{I}\hat{X}\hat{X}\hat{X}\hat{X}, & \hat{g}_4 &= \hat{I}\hat{I}\hat{I}\hat{Z}\hat{Z}\hat{Z}\hat{Z}, \\
\hat{g}_2 &= \hat{I}\hat{X}\hat{X}\hat{I}\hat{I}\hat{X}\hat{X}, & \hat{g}_5 &= \hat{I}\hat{Z}\hat{Z}\hat{I}\hat{I}\hat{Z}\hat{Z}, \\
\hat{g}_3 &= \hat{X}\hat{I}\hat{X}\hat{I}\hat{X}\hat{I}\hat{X}, & \hat{g}_6 &= \hat{Z}\hat{I}\hat{Z}\hat{I}\hat{Z}\hat{I}\hat{Z},
\end{align*}
\]

where these are to be read as tensor products.

This gives a compact representation for a code, just as the parity-check matrix is generally more compact than listing all the codewords of a classical linear code.

The generators are in two groups with the same form, one involving \(\hat{I}\)s and \(\hat{X}\)s, the other involving \(\hat{I}\)s and \(\hat{Z}\)s.
The patterns of these operators matches the parity-check matrix $H$ of a classical linear code:

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

This matrix defines the classical Hamming code. This same code is used to correct both bit-flip errors and phase-flip errors. The Steane code is, in a certain sense, the intersection of two Hamming codes.

A quantum code of this type—using a set of generators with only $\hat{X}$s and another set with only $\hat{Z}$s—is called a CSS (Calderbank-Shor-Steane) code. The 9-bit Shor code is also a CSS code.
More on stabilizer codes

- We can’t combine just any set of $\hat{X}$ generators with any set of $\hat{Z}$ generators. The reason is that the generators of a stabilizer code must all commute. If they don’t, there will not be a simultaneous +1 eigenspace of all the generators. No code space!

- Also, if the generators do not all commute, they cannot all be measured simultaneously. Measuring a generator of one type could generate an error of the other type. Notice, in the Steane code the generators all commute.

- If we use the same classical linear code $H$ for both the $\hat{X}$ and $\hat{Z}$ errors, the generators will only commute if the matrix $H$ is self-orthogonal:

$$HH^T = 0.$$
Code parameters

- Classically, suppose $t$ bits have been flipped from a valid codeword. It is then possible to correct the error if the valid codewords have a minimum Hamming distance of at least $d = 2t + 1$ from each other. This distance $d$ is one measure of how good a code is.

- Another important measure is the rate of a code. If we encode $k$ bits into $n$ bits, the rate is $k/n$.

- If a code encodes $k$ bits in $n$ bits with a minimum distance $d$, we call it a $[n, k, d]$ code.

- A quantum code also has a rate and a minimum distance, just like a classical linear code. We write $[[n, k, d]]$ for a quantum code.
Classically, the Hamming code encodes 4 bits and can correct up to one error. It is a \([7, 4, 3]\) code.

The Steane code, by contrast, only encodes 1 q-bit. The cost of correcting both bit-flips and phase-flips reduces the number of q-bits that can be stored. It inherits the distance \(d = 3\) from the Hamming code used to construct it. The Steane code is a \([[7, 1, 3]]\) quantum code.

If we construct a CSS code from an \([n, k, d]\) classical linear code with self-orthogonal check matrix \(H\), the resulting quantum code has parameters \([[n, 2k - n, d]]\).

Because it must correct more kinds of errors, a quantum code generally has a lower rate than the corresponding classical code.
More general stabilizers

The theory of stabilizer codes includes more than just the CSS codes, though they are the easiest to understand. More generally, a quantum stabilizer code is defined by a commuting set of stabilizer generators drawn from the Pauli group.

Consider a system of $n$ q-bits. A Pauli group element on this system is an operator of the form

$$(i)^{\ell} \hat{O}_1 \otimes \hat{O}_2 \otimes \cdots \otimes \hat{O}_n$$

where the $\{\hat{O}_j\}$ are either Pauli operators or the identity, $\hat{O}_j \in \{\hat{I}, \hat{X}, \hat{Y}, \hat{Z}\}$, and $\ell = 0, 1, 2, 3$.

For example, on a single q-bit we include the operators $\hat{X}, i\hat{X}, -\hat{X}, -i\hat{X}$, and similarly with $\hat{Y}, \hat{Z}$, and $\hat{I}$. 
For $n$ bits, this set of operators forms the Pauli group $G_n$ of size $n$. Consider a subgroup $S$ of $G_n$ consisting of commuting operators, such that these operators have certain states which are joint eigenvectors of all of them with eigenvalue 1. We call this joint +1 eigenspace $V_S$.

The subgroup $S$ is the stabilizer of $V_S$. The stabilized space $V_S$ is the code space.

Listing all the elements in $S$ is very cumbersome. Instead, we list a minimal set of generators for $S$. These generators are all we need to find $V_S$.

To be a stabilizer for a nontrivial subspace $V_S$, a subgroup obviously cannot include $-\hat{I}$. For the subgroup $S$ to be commuting, the minimal set of generators must commute.
Error-correcting conditions

Under what conditions will a code enable one to correct a particular error? Let \( \hat{P} \) be the projector onto the codespace \( V_S \), and assume that we wish to correct any error from a given set \( \{ \hat{E}_j \} \). The code allows any linear combination of these errors to be corrected if

\[
\hat{P} \hat{E}_i \hat{E}_j^\dagger \hat{P} = \alpha_{ij} \hat{P},
\]

where \( \alpha \) is an Hermitian matrix.

This works because we can form linear combinations of the \( \{ \hat{E}_j \} \) that diagonalize \( \alpha \). Such a linear combination maps the code space \( V_S \) into an orthogonal space.

One detects the error by measuring which of the orthogonal spaces the state is in, and then performing a unitary correction.
The outcome of this measurement is the error syndrome. Knowing the syndrome tells us which error has occurred, and hence how to correct it.

In the case of a stabilizer code, the syndrome measurement just consists of measuring the generators of the stabilizer. If the results are all 1, no error has occurred; any \(-1\) result means an error occurred. The pattern of 1s and -1s tells us which error has occurred. (This pattern is the \textit{error syndrome}.)

If the error operators \(\{\hat{E}_j\}\) are also Pauli group elements, then we can correct them by just applying the same group element again.

Because \textit{any} operator is a linear combination of Pauli group elements, it suffices to consider only this case.
A reasonable question to ask is what is the minimum number of q-bits needed for a codeword to correct a general single-bit error? We answer this by a counting argument. An encoded q-bit requires a two-dimensional subspace. We must have enough distinct syndromes to determine which error occurred. That means we need \( M + 1 \) two-dimensional subspaces to distinguish \( M \) possible errors (plus no error). The dimension must be at least \( 2(M + 1) \).

To correct a general single-bit error, it suffices to correct single \( \hat{X}, \hat{Y} \) and \( \hat{Z} \) errors. If there are \( n \) bits, then there are \( 3n \) single bit errors, and the Hilbert space has dimension \( 2^n \). Putting this all together, we must have

\[
2^n \geq 2(3n + 1).
\]
The smallest value of $n$ which satisfies this is $n = 5$. Remarkably, a 5-bit general error-correcting code actually does exist:

\[
|0\rangle \rightarrow \frac{1}{\sqrt{8}} \left( |00000\rangle + |11100\rangle - |10011\rangle - |01111\rangle + |11010\rangle + |00110\rangle + |01001\rangle + |10101\rangle \right),
\]

\[
|1\rangle \rightarrow \frac{1}{\sqrt{8}} \left( |11111\rangle - |00011\rangle - |10000\rangle + |01100\rangle + |11001\rangle - |00101\rangle - |01010\rangle + |10110\rangle \right).
\]
This 5-bit code is also a stabilizer (but not a CSS) code.

Note that your textbook gives a different version of the 5-bit code. They are equivalent to each other in error-correcting power.

Notice a difference between the five-bit and seven-bit codes: the $|0\rangle$ and $|1\rangle$ states for the seven bit code are an evenly-weighted superposition of 8 binary strings with all positive coefficients; the five-bit code has both positive and negative coefficients. There are variations of the five-bit code, but they all have that property. In a sense, one is too constrained by the need to fit all the subspaces into a 32-dimensional Hilbert space; one lacks the freedom to make all the coefficients positive. (Fortunately, this is not a great difficulty in quantum mechanics!)
Fault-tolerant quantum computation

- Error-correcting codes can lessen the effects of decoherence and other forms of noise. They do not, however, reduce the noise rate to zero. An error-correcting code that can correct a single error will still be defeated by multiple errors, which are bound to happen sooner or later.

- There are also more subtle weaknesses. If q-bits are decoded in order to have gates performed on them, errors that happen during those gates will not be corrected.

- Errors during error correction steps can also lead to uncorrectable errors.

- Let us treat the requirements for fault-tolerance one at a time.
Never decode the q-bits

- If we represent our logical q-bits by codewords in a larger space, the q-bits are insulated from errors. If we ever decode and re-encode them, any errors that happen during this process, or when the q-bit is decoded, are uncorrectable.

- This means that if we want to replace a quantum circuit with a circuit involving encoded q-bits, we must replace all of the gates by \textit{circuits} which perform the gates on the logical q-bits.

- For example, if we wish to perform a Hadamard on a logical q-bit, we have to find a circuit $\hat{U}$ which acts on codewords such that

\[
\hat{U}|0_L\rangle \rightarrow (|0_L\rangle + |1_L\rangle)/\sqrt{2}, \quad \hat{U}|1_L\rangle \rightarrow (|0_L\rangle - |1_L\rangle)/\sqrt{2}.
\]
In fact, the requirement is rather stronger than that. Suppose that a single error has occurred before the circuit which performs our logical Hadamard. We don’t want the circuit to *spread* this into multiple (and hence, uncorrectable) errors. All circuits for such logical gates must be performed *fault-tolerantly*, meaning that a single error either before or during the circuit results in only a single error after the circuit.

For two-bit gates, this is not usually possible. In this case, we impose a weaker requirement: if there is a single error in one of the two logical q-bits, the most that will happen after the gate is that there will be single errors in each of the two logical bits. These will both be separately correctable.
For an example of how this works, consider the Steane code. This code has very nice properties for encoded operations. To do a Hadamard gate on a logical q-bit encoded in the Steane code, we simply do a Hadamard on each of the seven bits independently:

\[
\begin{array}{c}
H \\
H \\
H \\
H \\
H \\
H \\
H \\
\end{array}
\]

The Steane code has this property for the Pauli $\hat{X}$, $\hat{Y}$, $\hat{Z}$ gates and the phase gate $\hat{S}$ as well.
In fact, it has that property for the CNOT gate as well:

One can also build fault-tolerant circuits for the $\pi/8$ gate and the Toffoli gate, so it is possible to construct a universal set of logical quantum gates.
Constantly correct

By never decoding, and by using fault-tolerant circuits for gates, it is possible to keep single errors from multiplying. However, we must also (as much as possible) keep them from accumulating. This means that we must do an error correction step after every (encoded) gate. If there is decoherence in bits not undergoing gates as well (memory errors), we must correct them periodically as well. To keep up with errors as computations become larger, we must be able to correct all (or a large fraction) of our encoded bits simultaneously.

So we see that fault-tolerance requires either no intrinsic decoherence (which is unlikely for most systems), or the ability to do parallel operations.
Measure carefully

When we measure (for instance, in checking error syndromes), we don’t want errors in our measurement process to propagate back into the q-bits we are measuring. Consider the following parity check:

An error in the ancilla can propagate up to all three bits.
Instead, we can use three ancilla bits prepared in a very special state.

\[
|\text{CAT}\rangle = (|000\rangle + |011\rangle + |101\rangle + |110\rangle)/2.
\]
If one of the three control bits is in the $|1\rangle$ state, it will flip one of the three ancillas. This takes us from a superposition of all *even* bit strings to a superposition of all *odd* bit strings:

$$|\text{CAT}'\rangle = (|001\rangle + |010\rangle + |100\rangle + |111\rangle)/2.$$  

Another bit flip will move us back to the state $|\text{CAT}\rangle$. After all three CNOTs, the parity of the three ancillas will be even if the three control bits had even parity, and odd if the three control bits had odd parity.

When we measure the ancillas, the parity of the string we get tells us the parity of the three control bits; but an error occurring to one of the three ancillas will only propagate up to a *single* control bit.
Concatenated codes

If we have done everything described above, we have turned a (sufficiently low) bit error rate $p$ into an error rate on the encoded bits of $cp^2 < p$, where $c$ is a constant to represent the extra overhead.

What can we do to improve things still further? Simply replace all of the bits in our new, encoded circuit with encoded bits! We then replace all of the gates with encoded gates, etc. We end up with nested codes and nested error corrections steps. A nested code of this type is called a \textit{concatenated code}.

(Recall that the Shor code was a concatenated code, with a phase-flip code on top of a bit-flip code.)

We do error corrections at every level of concatenation.
The threshold theorem

- If we do $k$ steps of concatenation, we have reduced our error rate from $p$ to $(cp)^{2^k}/c$. So for only poly-log additional overhead, we can reduce our error rate as low as we like, provided that the error rate $p$ is low enough to start with.

- This is the error threshold theorem. Current best estimates require that our error rate be less than one error in about $10^3 - 10^4$ gates. We are very far from such precision at present! But it shows there is hope in the long term.

- Some theoretical constructions suggest that a threshold as high as 1 errors per 100 gates may be possible.

- Next time: Requirements for QIP, and the Ion Trap Quantum Computer.