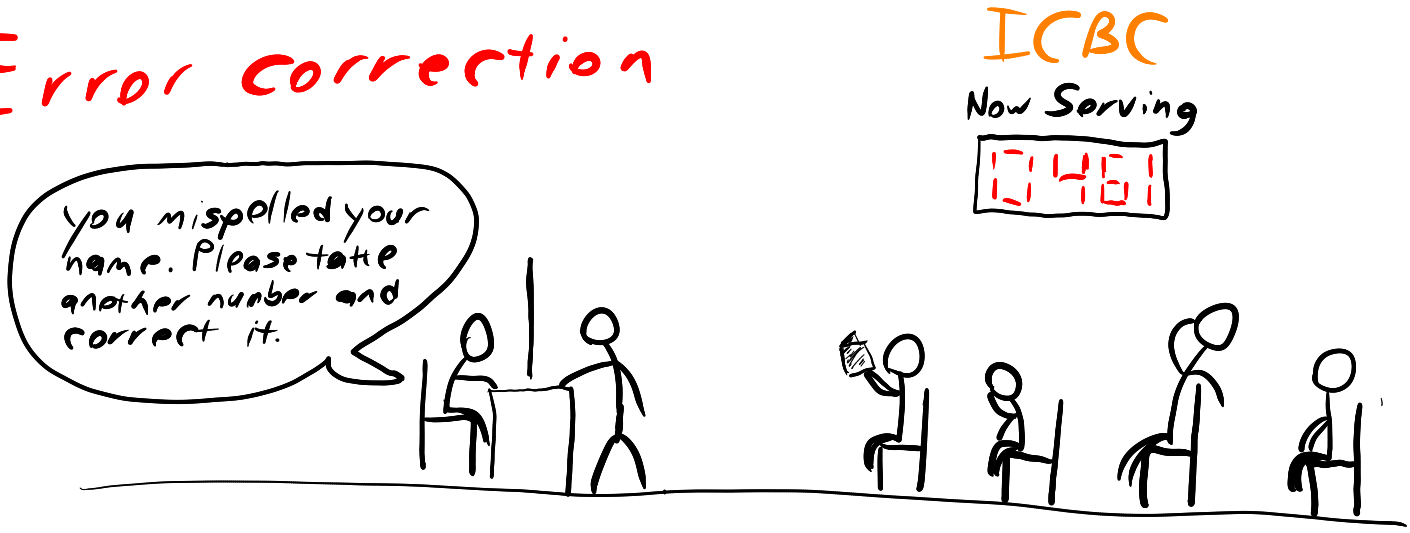
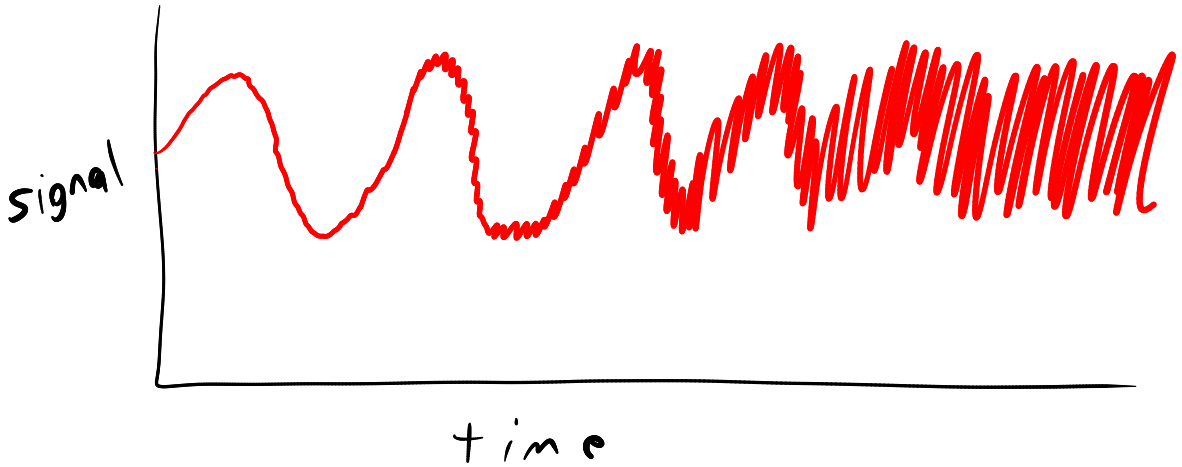
CMPT 476 Lecture 27

Error correction



We've spent a lot of time talking about quantum algorithms and the applications of quantum information processing, but there's a problem — a problem **so big** that it's led prominent researchers like **Gil Kalai** to believe quantum computing can never be realized in practice. This is the problem of **errors** and **noise**.



(Noisy computation and errors)

Consider a computer (quantum or otherwise) where every operation (e.g. arithmetic in a CPU) has a non-zero probability of error p. An algorithm with t steps hence has probability $(1-p)^t$ of succeeding without any errors. If p is very very small, maybe that's OK, but computation can quickly degrade with even small probabilities of error.

Classical computers are subject to errors (as is any necessarily imperfect physical process), but for most CPU's we can imagine it as around

10^{-17} ← probability a photon randomly flips a bit

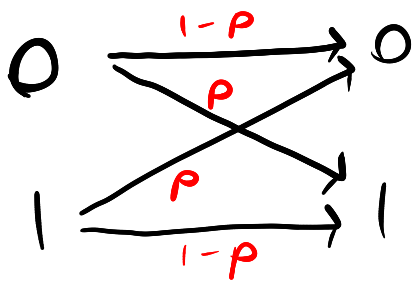
which amounts to $\sim 10^{17}$ operations to degrade.

By contrast, quantum computers have error rates around 10^{-2} and can optimistically get down to 10^{-3} in the coming years. At this rate, we get something like ~ 100 gates before our state is completely useless.

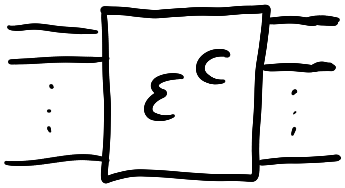
There are contexts in classical computing with high error rates, like data transmission through space, so to figure out how to deal with quantum noise, we can first look at how it's dealt with classically.

(Error models)

Before we can correct errors, we first need to know (formally) what errors can occur and how. This is called an error model. One simple classical error model is to assume that when we transmit n bits of information, each bit is flipped with independent probability p.



This is called the bit flip channel, denoted $\mathcal{E}^C$



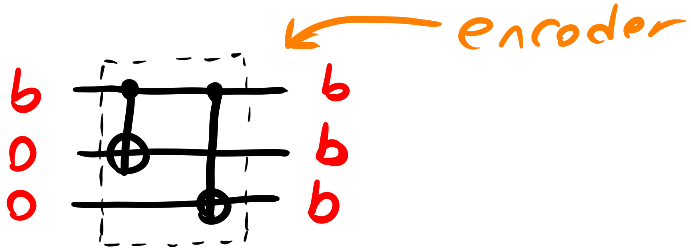
Note that for small p, a single bit flip is much more likely than two — roughly $O(p)$ vs $O(p^2)$.

(Error correction)

If we have a single bit b that we want to send over a noisy bit flip channel, the natural thing to do is make copies of b and send all of them. Intuitively we would expect that on average, assuming $p << \frac{1}{2}$, no bit flip occurs. If Alice is on the other end of the channel, she can then just take the average value, or the majority value of all received bits, which should be b with reasonably high probability.

(The three-bit code)

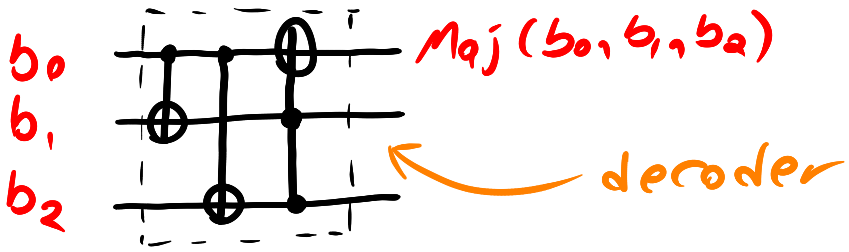
Suppose in the previous scheme we use 3 bits to send Alice the single bit b. First we prepare the 3-bit state bbb, called the encoding of b.



Then we send this to Alice through the bit flip channel, which results probabilistically in a three-bit state $b_0 b_1 b_2$. To decode $b_0 b_1 b_2$ (i.e. guess what b was), Alice takes the majority value of $b_0 b_1 b_2$. Noting that

$$\begin{aligned} Maj(b_0, b_1, b_2) &= b_0 b_1 \oplus b_0 b_2 \oplus b_1 b_2 \\ &= b_0 \oplus (b_0 \oplus b_1)(b_0 \oplus b_2) \end{aligned}$$

Alice can decode as follows



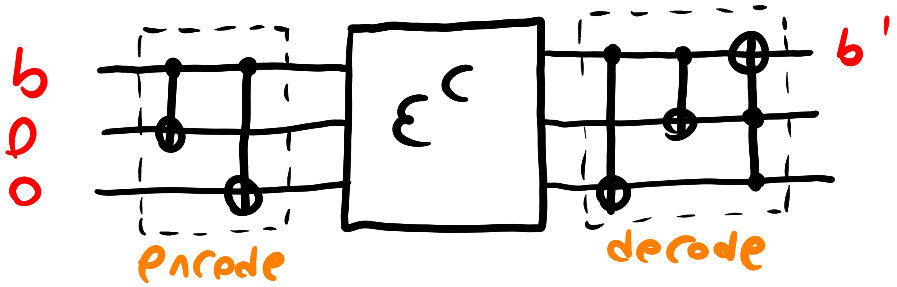
Suppose we sent the bit $b = 0$. What would the probability Alice decodes 0 be?

Bits flipped	probability	possible values
0	$(1-p)^3$	000
1	$p(1-p)^2$	100, 010, 001
2	$p^2(1-p)$	110, 101, 011
3	p^3	111

If the state $b_0 b_1 b_2$ is one of 000, 100, 010, 001, the majority is 0 so Alice's decoded bit is 0. However, the other 4 cases all have majority 1, in which case Alice's decoded bit has a bit flip. So the probability that the bit b has been flipped is

$$p' = 3p^2(1-p) + p^3$$

If $p = \frac{1}{4}$, then $p' = \frac{10}{64} = \frac{5}{32} < \frac{1}{4}$, so we have a lower probability of error than if we had just sent b alone. Here's the full protocol:



(Error correcting codes)

In the above we encoded a bit b as bbb and decoded three bits $b_0 b_1 b_2$ as $maj(b_0, b_1, b_2)$. This is an example of an **error correcting code** and an associated decoder. We refer to the encodings 000 and 111 as **codewords** and call 000 (resp. 111) **logical 0** (resp. 1).

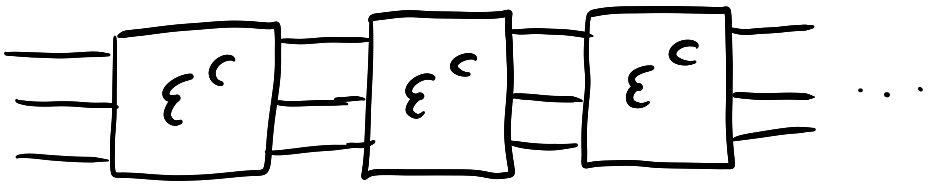
$$0_L = 000$$

$$1_L = 111$$

More generally, an (binary) error correcting (block) code associates to each string of k bits an $n > k$ bit codeword. If after transmission through a noisy channel, the n bit string is not a codeword, some error must have occured. Decoding amounts to determing the most likely code word, given the received word and error model

(Syndrome decoding and linear codes)

What if the memory in our computer was noisy itself? This would be the equivalent of repeatedly applying the bit flip channel to our state



Once we decode our state, we lose protection from errors, but if we don't decode we won't correct any errors that previously occured. This catch-22 can be solved by instead detecting which error occured and then correcting it on the encoded state.

Suppose for instance we had the initial state 000 and the bit flip channel flipped the second bit giving 010. and we could take the majority $maj(0,1,0) = 0$ and then flip each bit that differs from the majority. While fine for the three-bit code, this is generally inefficient, so most practical codes use Syndrome decoding, which efficiently detects which error occurred without explicitly finding the correct codeword. This is possible through the framework of linear codes.

(Syndrome decoding for the three-bit code)

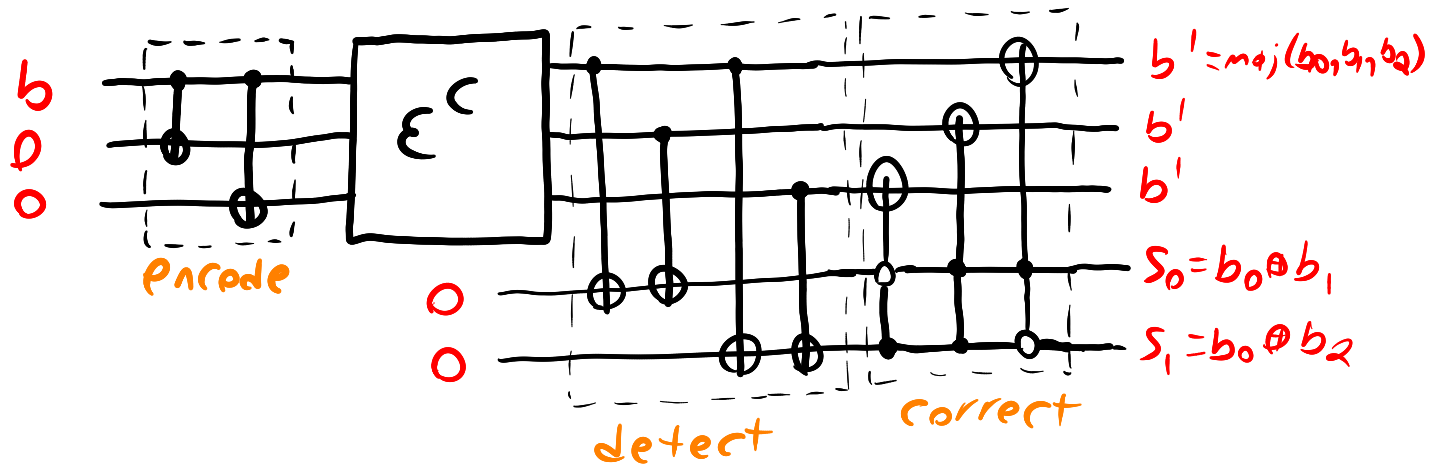
In the three-bit code, observe that if a single bit is flipped, e.g. $000 \to 010$, we can detect exactly where it occurred by taking the parity of b_0 & b_1, and the parity of b_0 & b_2.

$b_0 b_1 b_2$	$b_0 \oplus b_1$	$b_0 \oplus b_2$	bit flipped
000	0	0	none
001	0	1	2
010	1	1	1
100	1	0	0
111	0	0	none
110	0	1	2
101	1	0	1
011	1	1	0

So whether we started with 000 or 111, if 0 bits were flipped the syndrome string

$$s = (b_0 \oplus b_1,\ b_0 \oplus b_2)$$

is 0, and otherwise if 1 bit was flipped it tells us exactly which bit was flipped. If we wanted to implement this error detection and recovery as a circuit, we could do so by computing the syndrome and then applying bit flips conditional on the syndrome:



(Quantum error correction)

There are a number of obvious problems if we want to apply the ideas of classical error correction to quantum information.

1. We can't copy an arbitrary state $|\psi\rangle$ (no cloning theorem)
2. Errors form a continuum rather than discrete values. In particular, we may have
$$\mathcal{E}^Q : |\psi\rangle \longrightarrow \{(p_1, U_1|\psi\rangle), (p_2, U_2|\psi\rangle), \ldots\}$$
where we have infinitely many possible errors
3. In order to maintain superpositions, we have to detect errors without (completely) measuring the state.

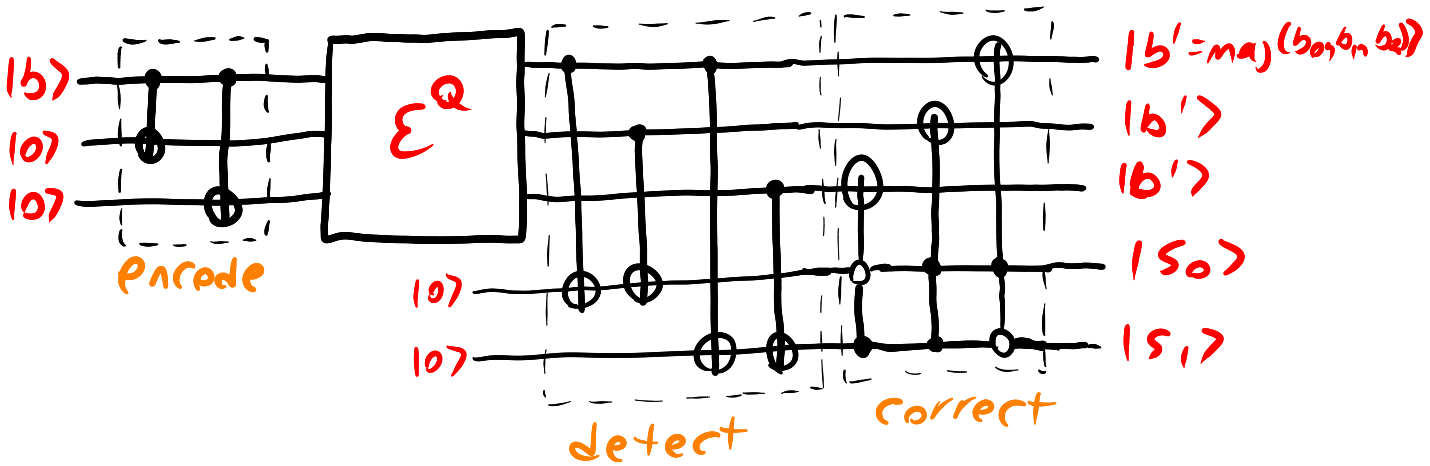
Let's ignore 2 for now and just think about how to correct bit flip errors on a quantum state: that is, correct errors of the form

$$\mathcal{E}^Q : |\psi\rangle \longrightarrow \{(1-p, |\psi\rangle), (p, X|\psi\rangle)\}$$

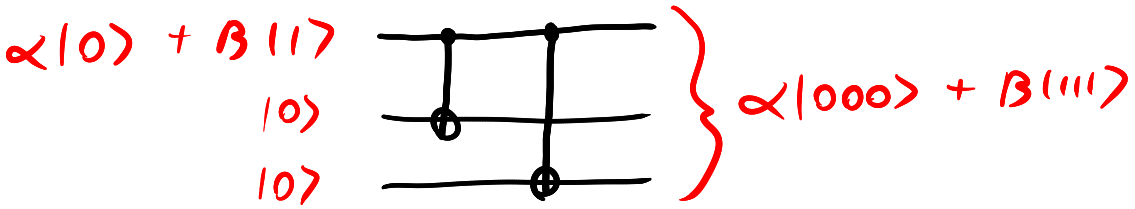
which is a probabilistic process — called a quantum channel — which sends a state $|\psi\rangle$ potentially in a superposition of $|0\rangle$ and $|1\rangle$ to either $|\psi\rangle$ or $X|\psi\rangle$ with probability $(1-p)$ or p resp.

(A quantum bit flip code)

We can't protect a state $|\psi\rangle$ from the bit flip channel by encoding it as $|\psi\rangle|\psi\rangle|\psi\rangle$ due to no-cloning, but we know if $|\psi\rangle = |0\rangle$ or $|\psi\rangle = |1\rangle$ we could encode as $|0\rangle|0\rangle|0\rangle$ or $|1\rangle|1\rangle|1\rangle$ and correct as we did classically:



Notice that we conspicuously designed the encoder and correction without the use of measurements. So, what happens if we encode a state in a superposition of $|0\rangle$ and $|1\rangle$?



Now when we apply the bit flip channel to this state, we get the following possible outcomes:

error	probability	state	syndrome
$I\otimes I\otimes I$	$(1-p)^3$	$\alpha\|000\rangle + \beta\|111\rangle$	$\|0\rangle\|0\rangle$
$I\otimes I\otimes X$	$p(1-p)^2$	$\alpha\|001\rangle + \beta\|110\rangle$	$\|0\rangle\|1\rangle$
$I\otimes X\otimes I$	$p(1-p)^2$	$\alpha\|010\rangle + \beta\|101\rangle$	$\|1\rangle\|1\rangle$
$X\otimes I\otimes I$	$p(1-p)^2$	$\alpha\|100\rangle + \beta\|011\rangle$	$\|1\rangle\|0\rangle$
$X\otimes X\otimes X$	p^3	$\alpha\|111\rangle + \beta\|000\rangle$	$\|0\rangle\|0\rangle$
$X\otimes X\otimes I$	$p^2(1-p)$	$\alpha\|110\rangle + \beta\|001\rangle$	$\|0\rangle\|1\rangle$
$X\otimes I\otimes X$	$p^2(1-p)$	$\alpha\|101\rangle + \beta\|010\rangle$	$\|1\rangle\|1\rangle$
$I\otimes X\otimes X$	$p^2(1-p)$	$\alpha\|011\rangle + \beta\|100\rangle$	$\|1\rangle\|0\rangle$

Now we may note that if we apply the correction corresponding to the syndrome to the state after $\mathcal{E}^Q$, in the first 4 cases (0 or 1 bit flip) we end up in the original encoded state $\alpha|000\rangle + \beta|111\rangle$, otherwise we end up with $\alpha|111\rangle + \beta|000\rangle$, or the encoding of $X|\psi\rangle$.

So what have we shown? That if rather than encode a quantum state $|\psi\rangle$ as $|\psi\rangle|\psi\rangle|\psi\rangle$, we encode **computational basis vectors** with the classical 3-bit code, i.e.

$$|0\rangle_L = |0_L\rangle = |000\rangle$$
$$|1\rangle_L = |1_L\rangle = |111\rangle$$

then as in the classical case we can recover from a single bit flip error. The key lies in the **detection** of the error via the syndrome, which circumvents the need to measure the state directly.

(Syndrome measurement)

In practice, we wouldn't want to implement the correction circuit as above, since it's expensive. Instead, we can observe that **measuring** the syndrome doesn't affect the superposition – it only **reveals** which error (probably) occurred. We can hence measure the syndrome and then perform **classically controlled** corrections, rather than Toffoli gates: