Lecture 27, Thurs April 27: Quantum Error Correction

At the end of the last lecture, we discussed some of the difficulties with achieving a quantum speedup using currently available quantum computing devices, like D-Wave's. We saw how D-Wave's devices are cooled to 10 milliKelvin, but even that might be too hot, and lead to too much decoherence and error! (Which, in the setting of adiabatic QC, shows up mostly as unwanted level crossings.)

You might have wondered:

If even 10 milliKelvin isn't cold enough---if nothing short of absolute zero and perfect isolation seem to suffice---then why should building a scalable quantum computer (one that achieves actual quantum speedups) be possible at all?

We need to separate two issues. First, there's the "engineering challenge" of building a scalable QC. Everyone agrees that, at a bare minimum, it will be *staggeringly hard* to achieve the required degree of isolation for thousands or millions of qubits, when those qubits also need to interact with each other in a carefully choreographed way. Maybe various practical problems will prevent human beings from doing it in the next 50 or 100 years. Maybe it will be too expensive. Theory alone can't answer such questions.

But then separately, there's the question of whether anything prevents scalable QC *even in principle*. Of course, if quantum mechanics itself were to break down, that could certainly prevent QC---but it would also represent a much more revolutionary development for physics than a "mere success" in building QC!

So, short of a breakdown of QM, on what grounds have people argued that scalable QCs aren't possible?

1) Large entangled states are inherently fragile, so they might be impossible to maintain.

If only one qubit in the computer decoheres, then the entire system could decohere.

If only one qubit in a "Schrödinger cat" type state leaks out into the environment, the quantum coherence between the $|\text{Alive}\rangle$ and $|\text{Dead}\rangle$ components is destroyed.

2) Applying unitary gates may produce lots of errors, which will snowball over time.

Maybe your Hadamard gate rotates by 46° instead of 45°. If so, then over many applications of the gate, the errors would pile up.

If you pick up a CS textbook from the 1950's, you'll see plenty of discussion surrounding "analog vs digital computers". The disappearance of analog computers from the scene had much to do with the difficulty of correcting continuous errors. Is a quantum computer simply another kind of analog computer?

3) The No-Cloning Theorem limits us.

Classically we deal with error by repetition (which implicitly means repeated measurements), but because of the No-Cloning Theorem, it seems that we can't do the same with quantum computers.

However, there were fundamental discoveries in the mid-1990s that addressed all three of these concerns, and that convinced most physicists and computer scientists that, short of some revolutionary change to known physics, the difficulties of building a scalable QC are "merely" difficulties of engineering, and not of principle. The first of these discoveries was the theory of…

Quantum Error Correction

Before discussing quantum error correction, let's briefly review how error correction works classically.

You could take a whole course about this subject.
We'll do the 5-minute version here.

The 3-bit Repetition Code

is a way to encode one logical bit using 3 physical bits.

$\overline{0}$, the logical 0 is encoded as "000"

and $\overline{1}$, the logical 1 is encoded as "111".

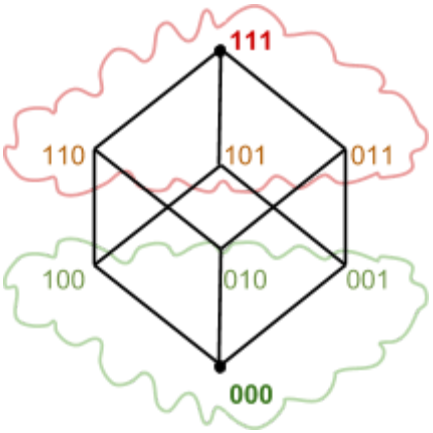
We claim that this code lets us both *detect* and *correct* an error in any one physical bit.

We do **error detection** by checking whether $x = y = z$. Assuming at most one incorrect bit, we'll be able to identify it. If we detect an error (say in x), we can do **error correction** by setting $x :=$ MAJ(x,y,z).

As an exercise, show that any code that can both detect and correct a single bit-flip error must use at least 3 bits. By contrast, if we just want to detect a single bit-flip error, and not correct it, show that 2 bits suffice.

More sophisticated codes are able to encode many logical bits at once, not just a single bit, while detecting and correcting a large number of errors (say, any 10% of the physical bits being flipped). In this lecture, though, we'll focus on encoding just a single bit (or qubit) and protecting against just a single error, in order to get the main conceptual points across.

Here's a useful geometric picture for why the 3-bit repetition code works.



Essentially, the code simply picks two points on the Hamming cube $\{0,1\}^3$ that are maximally far from each other, and declares one to be the encoding of 0 and the other to be the encoding of 1.

000 and 111 can each get corrupted to any point in their respective clouds, but since the two clouds don't overlap we're able to correct the error.

We'll seek to replicate this behavior in the quantum case.

In the early days of classical computing there were skeptics who doubted that the technology would ever scale. "Obviously," they said, "once the machine has enough vacuum tubes, some of them will fail, and that will cause the entire computer to fail."

John von Neumann was annoyed by this line of reasoning, so he went off and proved…

The Classical Fault-Tolerance Theorem

Von Neumann showed that, even given logic gates (like AND, OR, and NOT) every one of which fails with some independent probability ε, as long as ε is sufficiently small, it's possible to build a reliable circuit for any Boolean function *f*. Moreover, the circuit only needs to be a small factor larger than a circuit for *f* built of perfect gates. Informally, you can build a reliable computer out of unreliable parts.

Admittedly, if the output of the circuit is just a single bit, then that bit can't possibly take the correct value with probability greater than 1-ε---for what if the very last gate is erroneous? However, one way to deal with this issue is to let the output be a long string of bits, which is then processed by (say) a simple majority to get the value of *f*. This final majority computation is assumed to be error-free.

How did von Neumann prove the classical fault-tolerance theorem? We won't go through all the details, but the basic idea is to use the 3-bit repetition code recursively---pushing the probability of a catastrophic error down further and further with a majority of majorities (on 9 bits), a majority of majorities of majorities (on 27 bits), and so on.

Now, this seems to raise a conceptual puzzle:

Our error-correction circuits will themselves be subject to error. So when we compute these recursive majorities, won't we simply be introducing more errors than we correct?

Fortunately, von Neumann found that, as long as the physical error probability ε is small enough, each round of error-correction will be a "net win," making things better rather than worse.

Anyway, von Neumann's whole idea ended up being mostly unnecessary when transistors replaced vacuum tubes, since transistors err with such minuscule probabilities (in your entire computer, with its billions of transistors, perhaps one transistor will output a wrong result per year when it's hit by a cosmic ray).

Could such a thing happen with quantum computing: that is, a "QC transistor," which implements 1- and 2-qubit gates so reliably that error-correction is then superfluous?

Maybe! In the last lecture, we'll discuss topological quantum computing, which some physicists think could get us part of the way towards this dream. For now, though, we seem to be stuck in von Neumann's situation, with quantum gates that *do* have significant probabilities of error.

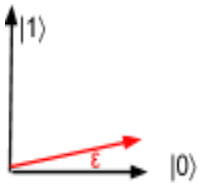
Since we discussed D-Wave in the last lecture: a potentially fateful decision that D-Wave made, early on, was that they weren't going to do any error correction. However, even D-Wave might now be coming around to the view that some degree of error correction is necessary.

Crucially, *until* you get over the hurdle of error correction, it may not look like your QC is doing much of anything useful. This is the main reason why progress in experimental quantum computing has

often seemed slow (with the world record for Shor's algorithm remaining the factorization of 21 into 3□7, etc.). Some people believe that practically important speedups will come only after we've overcome this hurdle.

You might worry that nothing like the 3-bit repetition code could possibly work in the quantum case, because quantum errors form a continuum: it's not a yes-or-no question whether an error has happened. So for example, we might get the error

$$|0\rangle \rightarrow \sqrt{1-\varepsilon^2}\,|0\rangle + \varepsilon|1\rangle.$$



It's said, only half-jokingly, that "80% of the work of building a quantum computer is reliably implementing the identity transformation," to keep the qubits steady.

Early in this course, though, we saw that the Watched Pot Effect (also called the Quantum Zeno Effect) was a way to keep a drifting qubit in check.
The trick is just to keep measuring the qubit in the $\{|0\rangle,|1\rangle\}$ basis.

If you get $|0\rangle$: "Was there an error?" "Well, *now* there's not!"

It's like the joke where someone calls 911 to report a dead body, and the operator asks the caller to check if the person is actually dead. Gunshots are heard; then the caller says, "OK, now what?"

If you get $|1\rangle$, correct it to $|0\rangle$.

But the Watched Pot Effect only solves the problem of quantum error-correction if
(1) the *only* thing we're worried about is continuous drift (rather than, e.g., discrete bit-flips), and
(2) we know a basis in which our qubit is supposed to be one of the basis vectors.
So how can we go further?

If we only needed to correct bit-flip errors, we could just use the obvious quantum analogue of the 3-bit repetition code, namely:

$$\overline{|0\rangle} \rightarrow |000\rangle \text{ and } \overline{|1\rangle} \rightarrow |111\rangle.$$

But bit-flip errors aren't the only things we're worried about!

For an example of what else could go wrong, let's look at the encodings of $|+\rangle$ and $|-\rangle$ under the above 3-bit repetition code:

$$\overline{|+\rangle} \rightarrow \frac{|000\rangle + |111\rangle}{\sqrt{2}} \text{ and } \overline{|-\rangle} \rightarrow \frac{|000\rangle - |111\rangle}{\sqrt{2}}$$

Question: How many qubits of $\overline{|+\rangle}$ do we need to act on to convert it to $\overline{|-\rangle}$?
Answer: Only one! (It suffices to apply a phase gate to any qubit.)

I.e., we don't have separate clouds.

But you might think the problem is even worse. Suppose we *did* manage to design a code that could correct both bit-flip errors and phase-flip errors. Even then, aren't there infinitely many *other* ways for a qubit to err? So won't we need to add an infinite amount of additional redundancy to our code?

Here's where a little piece of magic comes in and saves us. It turns out that, if a quantum error-correcting code protects against both bit-flip and phase-flip errors, then that's automatically enough to protect against *all possible* 1-qubit errors.

Why?

Without loss of generality, let's assume that we have an error on the first qubit of the entangled state $|\Psi_0\rangle$:

$$|\Psi_0\rangle = \alpha|0\rangle|v\rangle + \beta|1\rangle|w\rangle$$

A bit-flip error on the first qubit would result in

$$|\Psi_1\rangle = \alpha|1\rangle|v\rangle + \beta|0\rangle|w\rangle$$

A phase-flip on the first qubit would result in

$$|\Psi_2\rangle = \alpha|0\rangle|v\rangle - \beta|1\rangle|w\rangle$$

And both would result in

$$|\Psi_3\rangle = \alpha|1\rangle|v\rangle - \beta|0\rangle|w\rangle$$

The key observation is now that these four states constitute an orthogonal basis for the 4-dimensional subspace of all possible states that you could reach from $|\Psi_0\rangle$ by transformations on the first qubit.

This is extremely similar to Superdense Coding: in both cases, entanglement doubles the number of independent transformations that we can apply to a given qubit, leaving a perfect record of the fact that the transformation was applied, from 1 to 2, .

So, the simplest possible goal of quantum error-correction is now as follows: *assuming* that the error was in the first qubit only, get us back to $|\Psi_0\rangle$ from wherever we might happen to be in this 4-dimensional "error subspace."

Measuring all the qubits would be a really bad idea--since even if that told us where we were in the subspace, it would be a "pyrrhic victory" that destroyed our quantum state in the process.

Instead, maybe we can do a measurement that just projects onto one of the four basis vectors we saw.

If the outcome is $|\Psi_0\rangle$, we do nothing.

If the outcome is $|\Psi_1\rangle$, we get back to $|\Psi_0\rangle$ by doing a bit-flip.

If the outcome is $|\Psi_2\rangle$, we get back to $|\Psi_0\rangle$ by doing a phase-flip.

If the outcome is $|\Psi_3\rangle$, we get back to $|\Psi_0\rangle$ by doing both.

But what's a code that can detect and correct a bit-flip or a phase-flip error on *any* qubit?

Well, for starters, if we just wanted to detect and correct phase-flip errors, we could do so using the following code, namely "the 3-qubit repetition code except in the Hadamard basis":

$|\overline{+}\rangle \rightarrow |+\rangle|+\rangle|+\rangle$

$|\overline{-}\rangle \rightarrow |-\rangle|-\rangle|-\rangle$

OK, but just like the other 3-qubit code failed to protect against phase-flip erros, this code fails to protect against bit-flip errors. For by linearity,

$\overline{|+\rangle} + \overline{|-\rangle} = \overline{|0\rangle}$

$\overline{|+\rangle} - \overline{|-\rangle} = \overline{|1\rangle}$,

and this means (if you work it out) that applying a bit-flip to any qubit can change $\overline{|0\rangle}$ to $\overline{|1\rangle}$ or vice versa.

This observation brings us to our first serious quantum error-correcting code:

The Shor 9-qubit code (1995)

yes, it's the same Shor

Shor's proposal was simply to *combine* the two codes we've seen. He encodes each logical qubit as a 3□3 square of physical qubits, in such a way that each row corresponds to a 3-qubit repetition code to protect against bit-flip errors, and each column corresponds to a 3-qubit repetition code to protect against phase-flip errors.

Thus we encode $\overline{|0\rangle}$ as $\left(\frac{|000\rangle + |111\rangle}{\sqrt{2}}\right)^{\otimes 3}$

and $\overline{|1\rangle}$ as $\left(\frac{|000\rangle - |111\rangle}{\sqrt{2}}\right)^{\otimes 3}$

We claim that this code lets us detect and correct a bit flip *or* a phase flip (and hence, any possible error) on any one of the 9 qubits.

Why does this detect and correct bit-flip errors?

We just need build a little quantum circuit that checks whether all 3 of the qubits in a given row have the same value ($|0\rangle$ or $|1\rangle$), and if they don't, sets the wayward qubit equal to the majority of all 3 qubits in the row. We then apply that circuit to each of the 3 rows separately.

More interestingly…

Why does this code *also* detect and correct phase-flip errors?

Because we can build a quantum circuit that computes the relative phase between $|000\rangle$ and $|111\rangle$ within each row (+ or -), checks whether all 3 phases have the same value, and if they don't, sets the wayward phase equal to the majority of all 3 phases.

You can check that both of the operations above work, not just for $\overline{|0\rangle}$ and $\overline{|1\rangle}$, but for arbitrary superpositions of the form $\alpha\,\overline{|0\rangle} + \beta\,\overline{|1\rangle}$.

So, the above will fix any stray unitary transformations that get applied to any one qubit. But what about errors that involve decoherence or measurement?

We claim that once we've handled all possible unitary transformations, we've automatically handled *all* possible errors---because an arbitrary error might turn pure states into mixed states, but it still

keeps us within the same 4-dimensional subspace. So a measurement of the error syndrome still projects us down to one of the same four orthogonal states $|\Psi_0\rangle$, $|\Psi_1\rangle$, $|\Psi_2\rangle$, $|\Psi_3\rangle$, and we can still get back to our original state $|\Psi_0\rangle$ by applying bit flips and phase flips as needed.

What if it's not just one qubit, but *all* qubits that are a little bit off?

A key observation: if all qubits are a little bit off, then that's the same thing with having a superposition over many different configurations, in almost all of which only a few qubits are off (possibly a lot off). So we just apply a standard quantum error-correcting code that's able to deal with a few qubits being a lot off, and that lets us return our state *almost* to what it was before the errors happened.

Shor's 9-qubit code was the first quantum error-correcting code. Not long afterward, Andrew Steane found a shorter code that could also detect and correct any error on 1 qubit. Steane's code encoded 1 logical qubit into only 7 physical qubits. Then Raymond Laflamme and others found codes that used only 5 qubits.

5 qubits turns out to be the least possible if you want to detect and correct an arbitrary 1-qubit error, just like 3 bits is the least possible for a classical error-correcting code. We won't prove this.

In the next lecture, we'll discuss the stabilizer formalism, which gives us an amazingly compact and efficient notation for manipulating these sorts of quantum error-correcting codes.

So we can encode a qubit, let the qubit sit around passively, and protect it against a single physical error. How about doing a full fault tolerant quantum computation?
That's the subject of the famous….

Quantum Fault-Tolerance Theorem

Also known as the **Threshold Theorem**

Proved independently by several groups (Aharonov & Ben-Or / Zurek et al. ~1996)

The theorem says the following: suppose that, in your quantum computer, each qubit fails at each time step with independent probability ε. (Where "fails" could mean, e.g., gets swapped out for the maximally mixed state.)
Even then, assuming we're able to:

- apply gates to many qubits in parallel
- measure and discard bad qubits
- pump in fresh $|0\rangle$ qubits
- do extremely fast and reliable classical computation

we can *still* solve any problem in **BQP**, so long as ε is sufficiently small

Moreover, with only polylog(n) overhead of number of gates.

(The initial estimates for ε were $\sim 10^{-6}$. That's since been improved, depending on the assumptions.)

Since its discovery, the Fault-Tolerance Theorem has set much of the research agenda for experimental quantum computing. For it says that, once we can decrease error below a certain threshold,

we'll effectively be able to make it arbitrarily small, by applying multiple recursive layers of error-correction.

Journalists often try to gauge progress in experimental quantum computing by asking about the number of qubits, but at least as important is the *reliability* of the qubits. It's reliability that will determine when (if ever) we cross the threshold that would let us get to arbitrarily small error, and add as many additional qubits as we liked.

We're not there yet, but lots of progress is being made on two fronts:

1. Making physical qubits more reliable

Initially, the probability of failure per qubit per time step was of order 1, meaning that the quantum state would barely hold at all. But over the last twenty years, in multiple architectures (including superconducting qubits and trapped ions), there were improvements by several orders of magnitude. Today, anyone can access a 5-qubit superconducting device over the Internet (IBM's "Quantum Experience") that has decoherence rates that wouldn't have been achievable a decade ago.

Meanwhile, a few years ago the group of John Martinis at Google demonstrated decoherence rates, for one or two qubits in isolation, that are *already* below the fault-tolerance threshold (say, ~0.1%). So is the problem solved? Unfortunately, it's a bit misleading, because integrating more qubits on a single chip pushes the decoherence rate back up. So the challenge now is to figure out how to scale up to 50 or 100 or 300 qubits while still keeping the error rate down.

2. Inventing better error-correcting codes and fault-tolerance schemes

There are many tradeoffs here: to take one example, between the distance of a code (i.e., the rate of error it can handle) and the number of physical qubits it requires per logical qubit. At least heuristically, it's now known how to handle error rates of up to 3-5%, but only via fault-tolerance schemes that use thousands of physical qubits for every logical qubit. So one big challenge is to invent schemes that can handle large error and *also* have low overheads. Topological qubits, to be discussed two lectures from now, could also help with this tradeoff.

Here's one milestone that hasn't *quite* been achieved yet, but that you should look out for in the relatively near future: the use of a quantum error-correcting code to keep a logical qubit alive for longer that the physical qubits comprising it.