

Classical error correction

The most general classical single-bit error is the *bit-flip*:

$$0 \leftrightarrow 1.$$

We assume a simple error model, in which bit flips errors occur on each bit independently with probability p per unit time. We expect a bit to be corrupted after $O(1/p)$ steps. In general, we assume $p \ll 1$.

To get around this problem, we use *redundant coding* to help detect and correct errors. The simplest version of this is just to keep multiple copies of each bit.

If we keep two copies of each bit, 0 and 1 are *encoded* in the pairs 00 and 11, respectively. If an error occurs to one of the two bits, we will get the pair 01 or 10. Since these pairs should never occur, if we see them we know that an error has happened.

Slightly more technically, the strings 00 and 11 have *even parity*; if we detect a string with *odd* parity, we know that an error has occurred.

Detecting errors is all very well, but we would really like to do more than that: we would like to *correct* them as well. We can do that by increasing redundancy and keeping 3 copies of each bit:

$$0 \rightarrow 000, \quad 1 \rightarrow 111.$$

If an error occurs, we get one of the strings 001, 010, 100, 110, 101, 011. In this case, we correct the bit by using the majority value:

$$001, 010, 100 \rightarrow 000,$$

$$110, 101, 011 \rightarrow 111.$$

This is the simplest possible error-correcting code, the *majority-rule* code.

We can frame the majority rule code in terms of parities as well. In this case, we look at two parities: the parity of the first two bits, and the parity of the second two bits. For the legitimate code words 000 and 111, these both have parity 0; for all other strings, at least one of them has parity 1. We call these values *parity checks* or *error syndromes*:

00 = no error;

01 = bit 3 flipped;

10 = bit 1 flipped;

11 = bit 2 flipped.

If we know the error syndrome, we can correct the error by flipping the corrupted bit again.

The majority rule code lets one recover from a single error. But if two errors occur (to different bits) then the correction step will work incorrectly; the encoded bit will be flipped. E.g.,

$$000 \rightarrow 101 \rightarrow 111.$$

If three errors occur, then the error is not even detectable. E.g.,

$$000 \rightarrow 111.$$

In addition to this, the absolute probability of error has increased because of the encoding. Since we are representing each bit by several bits, the number of possible errors grows. The probability of a single-bit error goes from p to $3p$.

However, since we can correct single-bit errors, it is really two-bit and three-bit errors that are important. The probability of a two-bit error is $3p^2$, and the probability of a three-bit error is p^3 . If

$$3p^2 + p^3 < p,$$

then it pays to do error correction. If p is fairly small, then this will always be true. The key point is that for low levels of noise, error correcting codes can give a big improvement in performance for a relatively small overhead. We have changed the error probability to a higher power of p .

There are far more sophisticated and effective codes than the majority rule code. But the essential properties of all codes are well illustrated by this very simple example:

1. The state of single bits (or, more generally, of words) is embedded in a larger number of bits, exploiting a redundant representation.
2. Errors are detected by seeing that the bit string is not a legitimate code word.
3. These errors can be characterized by parity-checks or *error syndromes*.
4. If these syndromes give enough information, it is possible to conclude which error occurred and correct it.
5. Error correcting codes can only correct a limited number of errors; but if the intrinsic error rate p is small, they can reduce the overall error probability to higher order in p .

Quantum error correction

Naively it would seem that this kind of error correction is impossible for quantum systems. Redundancy seems to require the ability to copy the state of the quantum system, which is banned by the no-cloning theorem:

$$\hat{U}|\psi\rangle \otimes |0\rangle \otimes |0\rangle \neq |\psi\rangle \otimes |\psi\rangle \otimes |\psi\rangle.$$

Also, finding the syndromes requires making measurements, which would disturb the state. It seems that error correcting codes should have no quantum analogue.

Needless to say, that naive intuition is dead wrong.

The bit-flip code

For the moment, let us limit ourselves to *bit-flip* errors. We have seen how such error processes can arise from decoherence and/or imperfect gates. In the quantum context, a bit-flip is the same as an \hat{X} gate.

$$|0\rangle \rightarrow \hat{X}|0\rangle = |1\rangle, \quad |1\rangle \rightarrow \hat{X}|1\rangle = |0\rangle,$$

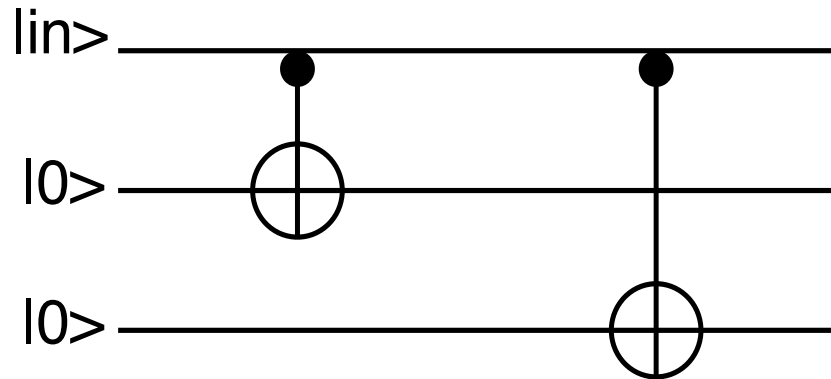
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow \hat{X}|\psi\rangle = \alpha|1\rangle + \beta|0\rangle.$$

We protect from such errors by unitarily *embedding* this single q-bit state in the state of three q-bits:

$$(\alpha|0\rangle + \beta|1\rangle)|0\rangle|0\rangle \rightarrow \alpha|000\rangle + \beta|111\rangle.$$

Note that we have not *copied* $|\psi\rangle$, so this doesn't violate the no-cloning theorem.

A simple circuit that does this encoding is



Suppose a single bit-flip error occurs on (say) bit 1. The state becomes

$$\alpha|100\rangle + \beta|011\rangle.$$

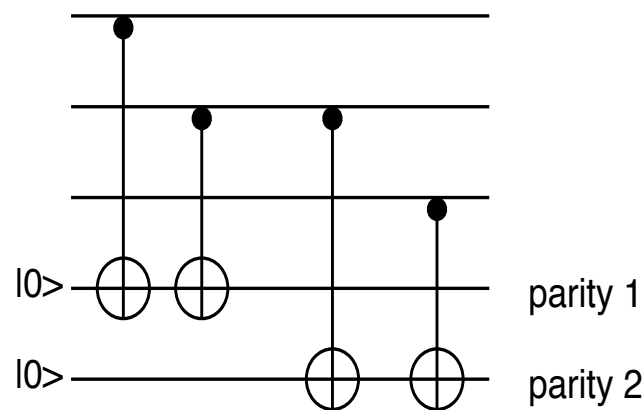
Similarly, errors on bits 2 and 3 result in states

$$\alpha|010\rangle + \beta|101\rangle, \quad \alpha|001\rangle + \beta|110\rangle.$$

We could determine which error (if any) occurred if we knew the parities of bits 1 and 2 and bits 2 and 3.

Now come the key insights that make quantum error correction possible.

1. It is possible to measure the parity of two bits without measuring the bits themselves.



2. It is possible to undo the \hat{X} error by means of a unitary gate.

By measuring the syndrome, we know if our system is in the space spanned by $|000\rangle$, $|111\rangle$, by $|100\rangle$, $|011\rangle$, by $|010\rangle$, $|101\rangle$, or $|001\rangle$, $|110\rangle$. We then either do nothing in the first case, or an \hat{X} on bit 1, 2, or 3 in the other three cases.

What if instead of a bit-flip \hat{X} , our error causes an X -rotation on one of the bits? This looks like $\cos(\theta/2)\hat{I} - i\sin(\theta/2)\hat{X}$, and our state becomes (e.g.,)

$$\begin{aligned} &\cos(\theta/2)(\alpha|000\rangle + \beta|111\rangle) \\ &-i\sin(\theta/2)(\alpha|010\rangle + \beta|101\rangle) \end{aligned}$$

if the rotation is applied to bit 2.

When we do the error syndrome measurement, with probability $\cos^2(\theta/2)$ we detect no error, and with probability $\sin^2(\theta/2)$ a bit-flip on q-bit 2 (which we then correct). In either case, we are left with the correct state! So this code protects not just against \hat{X} , but any error involving only \hat{I} and \hat{X} .

The reason that this works is that bit-flip errors on a single bit move the state into an orthogonal subspace. By measuring which subspace we are in (i.e., by measuring the parities), we can tell whether or not an error has occurred without disturbing the encoded state $|\psi\rangle$.

Just as in the classical case, if two or more errors occur our error correction fails. But again, just as in the classical case, we have reduced the probability of this from p to $\sim 3p^2$. So long as the error probability p is small, we have gained by doing error correction.

Unlike the classical case, though, there are more errors than just bit-flips in quantum mechanics.

The phase-flip code.

Suppose that instead of an error which multiplies a bit by \hat{X} , we have a process that multiplies by \hat{Z} :

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow \hat{Z}|\psi\rangle = \alpha|0\rangle - \beta|1\rangle.$$

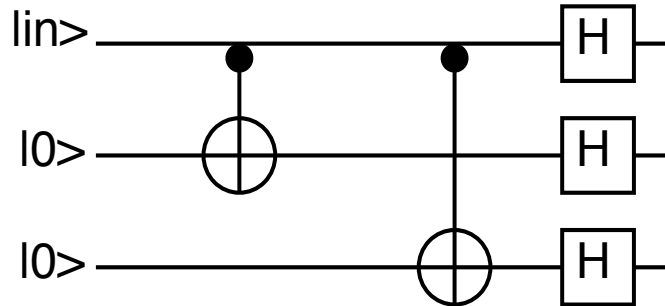
In this case, the code we have just seen is totally useless:

$$\alpha|000\rangle + \beta|111\rangle \rightarrow \alpha|000\rangle - \beta|111\rangle.$$

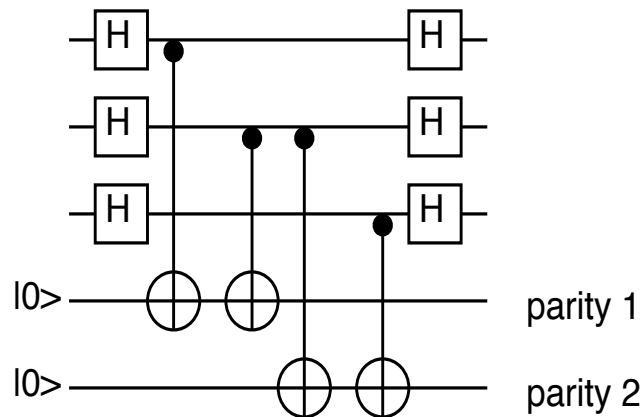
However, if we recall that X and Z are interchanged by the Hadamard, we can see how to protect against phase-flip errors as well:

$$\begin{aligned} (\alpha|0\rangle + \beta|1\rangle)|00\rangle &\rightarrow \frac{1}{2^{3/2}} (\alpha(|0\rangle + |1\rangle)^{\otimes 3} \\ &+ \beta(|0\rangle - |1\rangle)^{\otimes 3}). \end{aligned}$$

Here is a circuit which does this encoding:



The syndromes are read by the following:



The phase-flip code is just the same as the bit-flip code, only with the X and Z bases interchanged. This works because phase-flips look like bit-flips in the X basis, and vice-versa.

While these two codes demonstrate that in principle quantum error correction is possible, in practice they are not particularly useful. This is because in QM there are many more errors than in classical information theory. The bit-flip code will protect against any error operator involving only \hat{X} and \hat{I} , but is useless against \hat{Z} or \hat{Y} . Similarly, the phase-flip code will protect against any error operator involving only \hat{I} and \hat{Z} , but not against \hat{X} and \hat{Y} . We can build a similar code against \hat{Y} , but it is useless against \hat{X} and \hat{Z} .

Can we do better than this? Is it possible to design a code which protects against multiple *kinds* of errors?

The Shor Code

Consider the following encoding of a single bit in *nine* bits:

$$\begin{aligned} \alpha|0\rangle + \beta|1\rangle &\rightarrow \frac{\alpha}{2^{3/2}}(|000\rangle + |111\rangle)^{\otimes 3} \\ &+ \frac{\beta}{2^{3/2}}(|000\rangle - |111\rangle)^{\otimes 3}. \end{aligned}$$

These 9 bit code words have the structure of a phase-flip code, but with each of the $(|0\rangle \pm |1\rangle)$ replaced by a bit-flip code $(|000\rangle \pm |111\rangle)$.

A code of this type—one code nested inside of another—is called a *concatenated code*. This turns out to be a very important concept in making quantum computers (or indeed, classical computers) robust against noise. By concatenating codes we can make the error rates as low as we like, provided the initial rate is sufficiently low.

Error correction for the Shor code works as follows:

1. Do bit-flip error correction by measuring two parities for each triplet of bits and undo any bit-flip errors detected. The six parities measured are: $p_{12}, p_{23}, p_{45}, p_{56}, p_{78}, p_{89}$. This will detect up to one bit flip in each triplet.
2. Now measure the parity of the phases of *triplets* 1 and 2, and triplets 2 and 3. This will detect one phase flip in any of the bits.

We see that this code can detect a bit flip *and* a phase flip. But it can do more than that! Recall that $i\hat{Y} = \hat{Z}\hat{X}$. A \hat{Y} error is the same as an \hat{X} error followed by a \hat{Z} error. So this code can correct \hat{Y} errors as well.

Since any 1-bit operator can be written $\hat{O} = a\hat{I} + b\hat{X} + c\hat{Y} + d\hat{Z}$, this code can correct *any error on a single bit*. The Shor code protects against *general* errors.

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*:

$$|0\rangle \rightarrow \frac{1}{\sqrt{8}} \left(|0000000\rangle + |1010101\rangle \right. \\ \left. + |0110011\rangle + |1100110\rangle + |0001111\rangle \right. \\ \left. + |1011010\rangle + |0111100\rangle + |1101001\rangle \right),$$

$$|1\rangle \rightarrow \frac{1}{\sqrt{8}} \left(|1111111\rangle + |0101010\rangle \right. \\ \left. + |1001100\rangle + |0011001\rangle + |1110000\rangle \right. \\ \left. + |0100101\rangle + |1000011\rangle + |0010110\rangle \right).$$

This code is widely used because it has a number of very nice properties.

Stabilizer codes

Most of the codes demonstrated so far were developed by generalizing from classical error-correcting codes. The bit-flip and phase-flip codes were derived from the simple majority-rule code, and the Shor code produced by combining them. The Steane code was also generalized from a classical Hamming code. The question could then be asked: is there a general procedure for constructing quantum error-correcting codes?

The answer is that there are several such procedures. One of the most useful is that of *stabilizer* codes (or *additive* codes), which are analogous to classical *linear* codes. The Steane code is an example of a stabilizer code.

In classical linear codes, m bits are *encoded* as a string of n bits by writing the m bits as a Boolean m -vector and multiplying it by a Boolean $n \times m$ matrix,

$$x \rightarrow Gx.$$

The codewords representing x will be linear combinations of the columns of G ; the zero vector $x = 0$ will always be represented by a zero vector.

We check for errors using a parity-check matrix H chosen such that if y is a valid codeword then

$$Hy = 0.$$

G and H are obviously related; we want H to be a matrix whose rows are all linearly independent and are all orthogonal to the columns of G . This implies that if G is $n \times m$, then H is $(n - m) \times n$. It thus suffices to specify either G or H to give the code.

In constructing quantum error-correcting codes from classical linear codes, each row of the parity-check matrix is converted into a quantum parity-check operator. For example, in the bit flip code the parity-check operators are $\hat{Z} \otimes \hat{Z} \otimes \hat{I}$ and $\hat{I} \otimes \hat{Z} \otimes \hat{Z}$. By measuring these operators, errors are detected, which can then be corrected by unitary transformations.

Unfortunately, not all classical linear codes can be converted into well-defined quantum codes; it is necessary that all the parity-check operators commute with each other. The details of the construction are in Nielsen and Chuang. (Prof. Daniel Lidar will also be offering a course next semester on quantum error correction.)

The threshold theorem

The key result in the theory of quantum error correction is *the threshold theorem*: if a quantum computer has an intrinsic error rate per gate which is less than a certain threshold (currently estimated to be $\sim 10^{-4}$), it is possible by means of error correcting codes to make the total error probability arbitrarily low. That is, the overall probability of error for the whole computation can be made less than ϵ for any value of $\epsilon > 0$; and the overhead for doing so scales like $O(\text{polylog}(1/\epsilon))$.

This means that once it is possible to build q-bits and gates with sufficiently low decoherence, quantum computations of unlimited size are possible!

Next time: experimental implementations of quantum computing.