Quantum Coding Theory **(UC Berkeley CS294, Spring 2024)**

Lecture 1: Introduction and the Shor 9-qubit code

January 24, 2024

Lecturer: John Wright *Scribe: John Wright*

1 Introduction

1.1 The problem of noise in quantum computing

In 1994, Peter Shor discovered his quantum algorithm for factoring. As we all know, it was met with immediate acclaim and generated a significant increase in the level of interest in quantum computing. What is less remembered is that it was also met with a significant degree of skepticism, based on the belief that although it was a nice mathematical theorem, it would never be useful in practice. Why? The answer is *noise*.

An ideal quantum computation can be represented as a circuit, as in Figure 1.

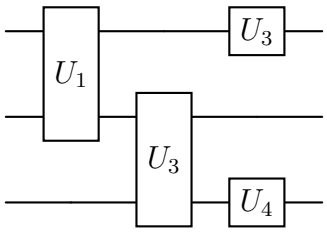


Figure 1: A quantum circuit.

In this circuit, the wires represent qubits, and the squares represent quantum operations, or gates, which are applied to them. Although this ideal circuit may work perfectly in theory, in practice a number of things can go wrong. For example, the hardware might be imperfect, and occasionally a gate might fail and do something completely different than what it was supposed to do. Another possibility is that a stray particle from the environment might interact with one of the wires, producing an error on that qubit. All of these are examples of noise, and they all have the potential to ruin the computation, resulting in an output which is useless.

One possible way to address this is to design hardware which is so precise that even lengthy computations can be expected to experience no errors. Roughly, if the ideal quantum circuit consists of T quantum gates, then we might hope that our quantum computer experiences an error on each gate with probability at most $p \leq O(1/T)$. In practice, though, things are much worse. For example, in 1995, one year after Shor's algorithm, an experimental quantum computer achieved an error probability of 20% per gate [**?**] (which means it could

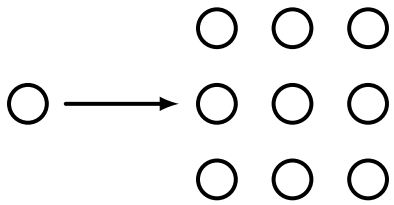
be expected to run for around 5 gates before introducing an error). Even now, 30 years later, the very best quantum computers still only achieve error rates in the range of roughly 0.5-1%, much worse than what we would need to carry out a typical quantum computation.

This raised the serious possibility that noise might preclude quantum computers from ever working, and researchers of this time took this possibility very seriously. For example, several papers considered physically-relevant models of noise and how they interacted with Shor's factoring algorithm, and they discovered that it does indeed render the computation useless [**?**, **?**]. One of the most prominent voices stressing the importance of studying noise in quantum computers was Rolf Landauer. Among other things, he was well-known for suggesting that any paper on quantum computation should come with the following disclaimer [**?**]:

"This proposal, like all proposals for quantum computation, relies on speculative technology, does not in its current form take into account all possible sources of noise, unreliability and manufacturing error, and probably will not work."

1.2 Enter quantum error correcting codes

One year later, in 1995, Shor [**?**] discovered that one could use tools from error correction to protect quantum states from the effect of noise. The result was his *9-qubit code*, which we will see later in this lecture. In the 9-qubit code, one encodes a single qubit, which might be error-prone, into a blob of 9 qubits, as in the picture below.



This encoding has the property that if exactly one qubit in the blob undergoes an error, the error can be detected and corrected. This gave a proof-of-concept that quantum noise could be overcome with error correction, even if the 9-qubit code only protects one qubit. Also in 1995 (and published in 1996), Andrew Steane independently discovered a different error correcting code known as the *7-qubit Steane code*, which we will see later in this semester.

One year later, Shor showed how to error-correct not just a single qubit, but an entire quantum computation. In [**?**], he gave a method to compile a general T-gate ideal quantum computation into a quantum circuit with $\text{poly}(T)$ gates which is resilient to noise. In particular, the compiled circuit will successfully simulate the original, ideal circuit so long as the error rate p on each gate is at most $O(1/\text{polylog}(T))$. (Here, $\text{poly}(T)$ refers to a function which is $O(T^k)$ for some k; similarly, $\text{polylog}(T)$ refers to a function which is $O(\log(T)^k)$ for some k.) Thus, the compiled circuit is not much larger than the original circuit, and it can tolerate much higher rates of noise than the rate of $p = O(1/T)$ which would be required if there were no error correction. However, this scheme still requires the noise rate to decrease as the size of the computation grows, which is not a reasonable assumption in practice.

The final breakthrough came later in 1996/7, independently by Kitaev [**?**], Aharanov and Ben-Or [**?**], and Knill, Laflamme, and Zurek [**?**], who improved on Shor's result and proved what is now known as the *fault-tolerance/threshold theorem*. They showed how to take an ideal T-gate quantum computation and compile it into a new circuit of $T \cdot \text{polylog}(T)$ gates—a minimal bloweup—with can correct against even *constant* levels of noise. This demonstrated once and for all that quantum computers could indeed overcome the problem of noise, and showed that developing scalable quantum computers is merely an astronomically difficult engineering task rather than an impossible engineering task. The error rate p below which error correction kicks in is known as the *threshold*, and its value depends on both the compilation scheme and the error model. Early estimates of p were quite low (in the ballpark of $p = 10^{-6}$; see [**?**] for a discussion), but more recent estimates of placed it in the range of 0.5-1%, tantalizingly close to the error rates that are currently achievable in practice.

1.3 Outline of the course

A large part of this course will be devoted to explaining these developments from the mid-1990's in order to prove the fault tolerance theorem. However, we will also cover more recent developments in quantum error correction. Very roughly, the course will be structured as follows.

1. Basics of quantum error correction. This will include defining quantum error correcting codes as well as giving examples of basic quantum error correcting codes, such as Shor's 9-qubit code and Steane's 7-qubit code.

2. "Good" quantum codes. The last few years have seen the surprising development of new quantum error correcting codes with extremely good parameters, which are known as "good" quantum LDPC codes. We will cover the development of these codes, starting from the toric code.

3. The fault tolerance and threshold theorem.

4. Applications of quantum error correction to other areas. The main application I have in mind is to the quantum PCP and NLTS conjectures.

Feel free to suggest other topics if you are interested in them! We have a diverse audience and I'd like to know what topics other people find interesting.

2 Going from classical to quantum codes

2.1 The 3-bit repetition code

To build up to our first quantum error correcting code, we will develop some intuition by studying the most basic of all classical error correcting codes, which is known as the *repetition code*. In the classical world, our data consists of bits $b \in \{0, 1\}$, and the noise we would like

to protect against is *bit flip noise*, which sends 0 to 1 and 1 to 0. To do so, we will encode our bit b by adding extra bits containing redundancy; these extra redundant bits will allow us to recover b even if one of the bits is corrupted.

Definition 2.1 (Repetition code)**.** The repetition code encodes a single bit $b \in \{0, 1\}$ into three bits by duplicating the bit three times, as follows.

$$0 \longrightarrow 000$$
$$1 \longrightarrow 111$$

The two strings on the right are known as *codewords*. 000 is the codeword representing the bit 0 and is therefore referred to as "logical 0", and similarly, 111 is referred to as "logical 1". We will sometimes use the following two pieces of notation for these codewords:

$$000 = \overline{0} = 0_L,$$
$$111 = \overline{1} = 1_L,$$

though I will probably switch between these notations throughout the course as I have not yet decided whether I prefer $\overline{0}$ or 0_L.

Suppose that one of our codewords experiences a single bit flip, such as:

$$000 \overset{\text{noise}}{\longrightarrow} 010.$$

Then we can correct this bit flip by looking at the majority of the three bits. Since only one of our bits has been flipped, this will give us the correct value for the three bits:

$$000 \overset{\text{noise}}{\longrightarrow} 010 \overset{\text{majority}}{\longrightarrow} 000.$$

On the other hand, this scheme will fail if the codeword experiences *two* bit flip errors:

$$000 \overset{\text{lots of noise}}{\longrightarrow} 011 \overset{\text{majority}}{\longrightarrow} 111 \neq 000.$$

Hence, the 3-bit repetition code allows us to correct one bit flip error, although it fails to correct two or more bit flip errors.

One final thing: note that 3 bit flips are required to move from 0_L to 1_L. (In other words, 0_L and 1_L have Hamming distance 3.) This means that the repetition code has *distance 3*. The distance of a code is an important parameter that we will discuss more in future lectures. But for now, it is useful to know that the bigger the distance of a code is, the more errors that one can correct at any given time.

2.2 What makes quantum error correction hard

We would like to use our intuition from the 3-bit repetition code to design a quantum error correcting code. Intuitively, this code should encode a qubit $|\psi\rangle = a \cdot |0\rangle + b \cdot |1\rangle$ so that it is resilient to quantum noise. However, there are three generally agreed upon issues that make designing quantum codes much more difficult than designing classical codes.

1. (No cloning theorem) Analogously to the repetition code, we might want to encode $|\psi\rangle$ by duplicating the state multiple times in order to add redundancy. However, the no cloning theorem states that the map

$$|\psi\rangle \to |\psi\rangle\,|\psi\rangle\,|\psi\rangle$$

is not possible. So how can we add redundancy?

2. (Quantum noise) Classically, there is only one source of noise: bit flips. Quantumly, however, there are many more types of noise. For example, there are *phase flips*, which act as follows:

$$a \cdot |0\rangle + b \cdot |1\rangle \xrightarrow{\text{noise}} a \cdot |0\rangle - b \cdot |1\rangle\,.$$

These have no classical analogue. Indeed, there are an *uncountably infinite* number of different types of quantum noise, since there is a continuum of different quantum operations. Designing one code that protects against all of them seems impossible!

3. (Collapse of the wavefunction) If an error *does* occur on our encoded state, the natural thing to do is to perform a measurement on the state in order to diagnose the error. However, quantum measurements can collapse the state, which might destroy the information we were trying to protect.

These three issues mean that even the mere *existence* of quantum error correcting codes is unclear, and they mean that designing quantum codes can be a very subtle task. Note that according to Dave Bacon [**?**], these three issues did not present *too* much difficulty to the earlier pioneers of quantum error correction, but they are still obstacles that any quantum error correcting code must overcome.

2.3 A simplified noise model

For the rest of this lecture, we will sidestep issue (2) above by considering a simplified noise model in which only *bit flip* and *phase flip* noise can occur.

Definition 2.2 (Bit flip noise)**.** Consider the unitary matrix

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

This acts on the standard basis as follows:

$$X \cdot |0\rangle = |1\rangle\,, \quad \text{and} \quad X \cdot |1\rangle = |0\rangle\,.$$

As a result, this is known as a *bit flip* error. In the Hadamard basis, it acts as follows:

$$|+\rangle = |0\rangle + |1\rangle \mapsto |1\rangle + |0\rangle = |+\rangle\,,$$
$$|-\rangle = |0\rangle - |1\rangle \mapsto |1\rangle - |0\rangle = -\,|-\rangle\,.$$

Definition 2.3 (Phase flip noise)**.** Consider the unitary matrix

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

This acts on the standard basis as follows:

$$Z \cdot |0\rangle = |0\rangle, \quad \text{and} \quad Z \cdot |1\rangle = -|1\rangle.$$

As a result, this is known as a *phase flip* or a *sign flip* error. In the Hadamard basis, it acts as follows:

$$|+\rangle = |0\rangle + |1\rangle \mapsto |0\rangle - |1\rangle = |-\rangle,$$
$$|-\rangle = |0\rangle - |1\rangle \mapsto |0\rangle + |1\rangle = |+\rangle.$$

Note that phase flip noise acts exactly as bit flip noise, only in the Hadamard basis, as it exchanges the roles of $|+\rangle$ and $|-\rangle$. In fact, we have the following relationship between these two matrices.

$$Z = HXH \quad \text{and} \quad X = HZH.$$

Once we allow qubits to undergo bit flips and phase flips, we should also allow them to undergo *both* types of noise at the same time. This results in a third type of noise which, as far as I know, doesn't have a nice name like "bit flip" or "phase flip".

Definition 2.4 (Bit and phase flip noise)**.** Consider the unitary matrix

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = i \cdot XZ.$$

This first applies a phase flip and then applies a bit flip (and then multiplies by a random factor of i for some reason). As a result, this acts on the standard basis as follows:

$$Y \cdot |0\rangle = i \cdot |1\rangle, \quad \text{and} \quad Y \cdot |1\rangle = -i \cdot |0\rangle.$$

Why include the factor of i? As a matter of fact, it's not necessary at all. Indeed, in John Preskill's quantum error correction notes [**?**], he simply defines the Y matrix as

$$Y = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

and omits the i. Daniel Gottesman, in his excellent lectures on quantum error correction [**?**], states that the definition with the i gives you extra headaches in some parts of quantum coding theory, but that the definition without the i gives you extra headaches in *other* parts of quantum coding theory. We'll presumably see this as we go along in the semester. Of course, one reason to include the factor of i is that the matrices X, Y, Z are the famous *Pauli matrices*, which recur throughout quantum computation.

Remark 2.5 (Completeness of Pauli noise)**.** As it turns out, these Pauli errors are *complete* for quantum noise, in the sense that if you design a quantum code that can correct for Pauli errors, then your quantum code corrects for all other types of errors for free. We will see this statement justified in an upcoming lecture. For now, we will just assert it is true. But this makes it a lot easier to design quantum codes using classical intuition, as it means that quantum error correction is exactly the same as classical error correction, except you have to worry about one more basis (i.e. the Hadamard basis).

3 The 3-qubit bit flip (/repetition) code

In order to build up to Shor's 9-qubit code, we will first construct a 3-qubit code which corrects against a single bit flip error but not against any phase flip errors. In designing this code, we will address issues (1) and (3) from above, by avoiding the no-cloning theorem and the collapse of the wavefunction.

3.1 Definition of the code

Definition 3.1 (The 3-qubit bit flip code)**.** To define the 3-qubit bit flip code, we will first define it on the standard basis, and then we will extend it to general states via linearity. Given a basis state $|b\rangle$, for $b \in \{0, 1\}$, we encode it by duplicating the bit three times, as follows.

$$|0\rangle \longrightarrow |0\rangle \otimes |0\rangle \otimes |0\rangle$$
$$|1\rangle \longrightarrow |1\rangle \otimes |1\rangle \otimes |1\rangle$$

We will often drop the tensor products and write $|000\rangle$ as shorthand for $|0\rangle \otimes |0\rangle \otimes |0\rangle$. The state $|000\rangle$ is known as the "logical 0 state" and can be written as either $\left|\overline{0}\right\rangle$ or $|0\rangle_L$; similarly, $|111\rangle$ is known as the "logical 1 state" and can be written as either $\left|\overline{1}\right\rangle$ or $|1\rangle_L$. A general state is encoded as follows.

$$|\psi\rangle = a \cdot |0\rangle + b \cdot |1\rangle \longrightarrow a \cdot |000\rangle + b \cdot |111\rangle .$$

The state on the right is the *logical qubit* and can be written as $|\psi\rangle_L$ or $\left|\overline{\psi}\right\rangle$; it is the representation of the original qubit in our error correcting code. On the other hand, the three qubits which are used to encode the state are known as *physical qubits*.

We note that this code circumvents the no cloning theorem by not actually cloning the qubit state $|\psi\rangle$. In other words,

$$a \cdot |000\rangle + b \cdot |111\rangle \neq (a \cdot |0\rangle + b \cdot |1\rangle) \otimes (a \cdot |0\rangle + b \cdot |1\rangle) \otimes (a \cdot |0\rangle + b \cdot |1\rangle).$$

What it *does* do is it uses entanglement to copy only the standard basis information across the three qubits, which is not disallowed by the no cloning theorem.

The encoding map for the 3-qubit bit flip code can be carried out by the quantum circuit in Figure 2.

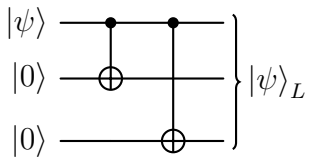


Figure 2: The encoding circuit for the 3-qubit code.

The quantum gate which is used twice in that circuit is the CNOT gate, which acts in Figure 3 for all $a, b \in \{0, 1\}$.

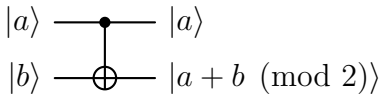


Figure 3: The CNOT gate.

3.2 Bit flip error

Now let us investigate how this code behaves under bit flip errors. Suppose that a bit flip has occurred on the second qubit. This gives us the state

$$I \otimes X \otimes I \cdot |\psi\rangle_L = a \cdot |010\rangle + b \cdot |101\rangle .$$

The state on the right-hand side is the *corrupted state* and we will write it as $|\widetilde{\psi}\rangle_L$. To detect that an error has occurred, a natural thing to do would be to measure the three qubits in the standard basis. This would give us the string 010 with probability $|a|^2$ and the string 101 with probability $|b|^2$. In either case, we can observe that the three bits are not the same, and so an error must have occurred. However, in doing so, we have collapsed the superposition, meaning that we cannot recover $|\psi\rangle$. So although we have detected the error, we cannot correct it.

The issue is that we measured too much: not only did we learn that there was a bit flip, we also simulated a standard basis measurement on $|\psi\rangle$. So we should instead try to measure whether there was a bit flip and *only* whether there was a bit flip. To do this, we will measure (i) the parity of the first and second qubits and (ii) the parity of the second and third qubits. These parities are 1 and 1, respectively, in both branches of the superposition, and so measuring them will not collapse the state. In other words, we perform the circuit given Figure 4.

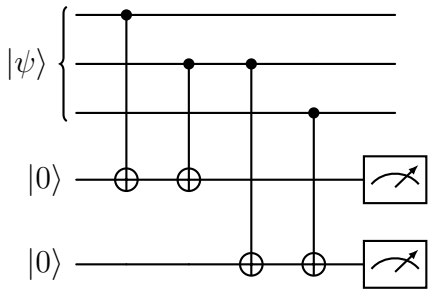


Figure 4: Circuit for bit flip detection.

This circuit acts on the state $|\psi\rangle_L$ as follows.

$$\begin{aligned}
&|\psi\rangle_L \otimes |00\rangle \\
&= (a \cdot |010\rangle + b \cdot |101\rangle) \otimes |00\rangle \\
&= a \cdot |010\rangle \otimes |00\rangle + b \cdot |101\rangle \otimes |00\rangle \\
&\to a \cdot |010\rangle \otimes |11\rangle + b \cdot |101\rangle \otimes |11\rangle \\
&= a \cdot |010\rangle \otimes |00\rangle + b \cdot |101\rangle \otimes |11\rangle \\
&= |\psi\rangle_L \otimes |11\rangle .
\end{aligned}$$

The circuit concludes by measuring the second register, which results in the measurement outcome 11. This measurement outcome is known as the *syndrome*, and it gives the information necessary to diagnose where an error has occurred. In this case, the syndrome specifies that the qubits 1 and 2 disagree, as well as qubits 2 and 3, which indicates that a bit flip error has been applied to the second qubit. In general, the four possible syndromes specify the following errors.

Syndrome	Error
00	no error
01	bit flip on 3rd qubit
10	bit flip on 1st qubit
11	bit flip on 2nd qubit

It will be convenient to associate the syndrome bits with the error message that they specify. So in the future, we may write $|00\rangle = |\text{no error}\rangle$, $|01\rangle = |X \text{ on 3rd}\rangle$, and so forth.

In this case, since the syndrome specifies that a bit flip has been applied to the second qubit it, we can correct this by applying a second bit flip to that qubit. Then, as

$$I \otimes X \otimes I \cdot |\widetilde{\psi}\rangle_L = |\psi\rangle_L ,$$

we have recovered our original state and corrected the error. As a result, this code can correct one bit flip error. (Note that like the classical 3-bit repetition code, it cannot correct two bit flip errors.)

3.3 Phase flip error

Suppose, on the other hand, that a phase flip error occurs on one of the qubits—the first qubit, for example. Writing ZII as shorthand for $Z \otimes I \otimes I$, this gives the state

$$\begin{aligned} ZII \cdot |\psi\rangle_L &= ZII \cdot (a \cdot |000\rangle + b \cdot |111\rangle) \\ &= a \cdot |000\rangle - b \cdot |111\rangle \,. \end{aligned}$$

Can we detect this error? Unfortunately, the answer is no: $a \cdot |000\rangle - b \cdot |111\rangle$ is the encoding of the state $a \cdot |0\rangle - b \cdot |1\rangle$, and so this appears to be a legitimate encoding of a state. As a result, although this code can detect a bit flip error, it cannot detect a phase flip error.

3.4 Encoded operations

Suppose we wanted to apply a unitary operation to $|\psi\rangle$, but we only have access to the encoded state $|\psi\rangle_L$. One thing we could do is unencode $|\psi\rangle_L$ to produce $|\psi\rangle$, apply the unitary, and then reencode the resulting state. However, this defeats the purpose of encoding the state to begin with, as unencoding it makes it vulnerable to errors again. Instead, we would like to apply the unitary by acting directly on the encoded state $|\psi\rangle_L$.

Suppose $|\psi\rangle = a \cdot |0\rangle + b \cdot |1\rangle$ and $|\psi\rangle_L = a \cdot |000\rangle + b \cdot |111\rangle$. If we wanted to apply a bit flip to $|\psi\rangle$, this would correspond to applying the unitary XXX to $|\psi\rangle_L$, as

$$XXX \cdot |\psi\rangle_L = a \cdot |111\rangle + b \cdot |000\rangle \,,$$

which is the encoding of $X \cdot |\psi\rangle$. In this case, XXX is referred to as a "logical X" and can be written as X_L or $\overline{X}$.

Similarly, suppose we wanted to apply a phase flip to $|\psi\rangle$. Then the corresponding operation on $|\psi\rangle_L$ would be any of ZII, IZI, IIZ, or ZZZ. Any of these could be our choice for the "logical Z" operation $Z_L = \overline{Z}$. Note that our logical X operation just applies X to each physical qubit, and this last logical Z operation just applies Z to each physical qubit. These are known as *transversal gates* because they apply the desired unitary to each qubit in the encoding. They are important for fault tolerance, and we will see them in more detail later in the class.

3.5 Conclusion

We have seen the 3-qubit bit flip code, which protects against a single bit flip error (but not against two or more bit flip errors or a phase flip error). This code was actually first discovered by Asher Peres in 1985 [**?**] in a work studying how to simulate classical computations on quantum computers and make them robust to noise. (See the interview by Shor [**?**] where he discusses the relationship of this work to his own.) Next class, we will see another code for correcting phase flip errors, and then we will see how to combine the two to create the 9-qubit code.