# Quantum Error Correction - Lecture 5

with Dan Browne

March 28, 2016

## Contents

## 1 Introduction

In this lecture, we shall introduce the concept of fault-tolerant quantum computing. We will especially focus on the threshold theorem and emphasise how important efficient scaling methods becomes for fault-tolerant computing.

## 2 The Clifford group

The concept of a Clifford algebra might sound related. This is a type of associative algebra that generalises the notion of real numbers, complex numbers, quaternions and so on.

The Clifford group is something else, however. It is the set of unitary gates that maps to stabiliser states.

What then are stabiliser states? They are states, or codewords in a stabiliser code with $k = 0$. That is, the stabiliser code that does not encode any qubits. This code has the property $n = m$, thus the number of independent stabiliser generators is $m = n$.

For example, the very simplest code contains the following states and stabilisers.

| State | Stabiliser |
|---|---|
| $\|0\rangle$ | $Z$ |
| $\|1\rangle$ | $-Z$ |
| $\|00\rangle + \|11\rangle$ | $XX, ZZ$ |
| $\|000\rangle + \|111\rangle$ | $XXX, ZZI, IZZ$ |

For example, the Hadamard gate acts as a stabiliser in that it maps a codeword to the same codespace.

$$H|0\rangle \to |+\rangle \tag{1}$$

$$H|1\rangle \to |-\rangle \tag{2}$$

We find that $H$ is a Clifford group unitary. So it the $Z$ operator,

$$Z|0\rangle = |0\rangle \tag{3}$$

$$Z|1\rangle = -|1\rangle \tag{4}$$

Note the global phase here. It does not affect the stabiliser group since the states $|1\rangle$ and $-|1\rangle$ are experimentally indistinguishable. The CNOT gate is another example of a member of the Clifford group.

Now that we have listed a few gates, we might perhaps rather have a simple way to demonstrate the characteristics of Clifford group gates. We can do so by considering a general operator $\sigma$ as part of the Clifford group and then use a set of unitary matrices $\{U\}$ in a similarity transformation to map the elements of one Puali group to other elements in the Pauli group. That is,

$$\sigma \to \sigma' = U\sigma U^\dagger \tag{5}$$

where $\sigma$ is some Pauli operator (holds for the $N$-qubit case as well). That is, the Clifford group is the group of unitary matrices that maps elements in the Pauli group to another element in the Pauli group.

A convenient fact which we shall now prove is that this operation is equivalent to conjugation of the operators. So, a stabiliser $S_j$ fulfils the following relationship

$$S_j|\psi\rangle = |\psi\rangle \tag{6}$$

for all $j$. We say that $S_j \in$ the stabiliser group. Consider now

$$US_j|\psi\rangle = U|\psi\rangle \tag{7}$$

But since we know that $U^\dagger U = \mathbb{I}$, we can insert this into the expression to find,

$$US_jU^\dagger U|\psi\rangle = U|\psi\rangle \tag{8}$$

which again holds for all $j$. In fact, $US_jU^\dagger$ is just another stabiliser.

In order to prove that all the objects that we can obtain for $US_jU$ with different unitary matrices form a group, we need to prove the group properties. First, we have the identity. This is quite clearly filled

by the identity matrix $\mathbb{I}$. Secondly, there exists an inverse since we are dealing with unitary matrices. By conjugating the matrices, we obtain the inverse. Finally, the group is closed because, as we showed, the product $US_jU^\dagger$ still maps the state $U|\psi\rangle$ to itself. Thus, $US_jU^\dagger$ is also a stabiliser, and the group is closed.

Let us have a look at another gate which is a member of the Clifford group. This is the $S$-gate, given by

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{9}$$

We find that $S \in$ the Clifford group, since

$$SZS^\dagger = S \tag{10}$$

That is, $S$ maps $Z$ to itself, and thus stays in the Clifford group. This can be easily confirmed since $[S, Z] = 0$.

The $S$ gate has the following effect.

$$S|+\rangle = |+i\rangle \tag{11}$$

$$S|+i\rangle = |-\rangle \tag{12}$$

On the Bloch sphere, the $S$ gate can be thought of as a rotation of 90 degrees on the equator.

For a single qubit, the Clifford group is generated by $H$ and $S$. This is all we need.

Recall that the stabiliser states in the single qubit group lie on an octahedron in the Bloch sphere - one for each axial extremal point. Thus, the symmetry group for the stabilisers is the symmetry group of the octahedron. Here, all $X, Y, Z$ are rotations of 180 degrees around a certain axis. The Hadamard gate is a reflection about a diagonal line.

However, we want to be able to go from the single qubit Clifford group to the $N$-qubit Clifford group. It turns out that it is generated by the set

$$\{H, S, CNOT\} \tag{13}$$

which must act on every qubit and every pair of qubits. Note the similarity between this set and the universal quantum computing gate set.

We now wish to prove that the CNOT gate is a meber of the Clifford group. It suffices to check that

$$U\sigma U^\dagger = \sigma' \tag{14}$$

where $\sigma$ is some Pauli operator. We use the following notation $CNOT_{CT}$ where $C$ is the control qubit and $T$ is the target qubit. We know that CNOT is self-inverse. We can check that

$$CNOT_{CT}(Z \otimes I)CNOT_{CT} = I \otimes Z \tag{15}$$

The control commutes with $Z$ gates. WE also find that

$$CNOT(X \otimes I)CNOT = X \otimes X \tag{16}$$

$$CNOT(I \otimes Z)CNOT = Z \otimes Z \tag{17}$$

Note that the commutator relations are preserved. Finally,

$$CNOT(I \otimes X)CNOT = I \otimes X \tag{18}$$

Since we proved this for generators of the Pauli group, we know it applies to the entire group.

# 3 Gottesmann-Knill Theorem

Theorem: The Clifford group circuits acting on stabiliser states are easy to simulate classically. Any quantum circuit on $n$ qubits acting on an initial stabiliser state consisting of a polynomial in $n$ different gates can be effectively simulated in $\mathcal{O}(n)$ gates on a classical computer.

Proof idea: We shall here not prove the entire theorem, but rather give a flavour of the proof ingredients.

- Represent states by stabiliser generators and show that we end up with unitary update rules.
- Then, must show that the measurement update rules are all efficiently simulatable.

The consequences are substantial. It shows that Clifford gates, stabiliser states and the Pauli group is not enough for universal quantum computing.

So given stabiliser states, Clifford unitaries and Pauli measurements, what must we add for true universal computing?

It turns out that we need to add a single non-Clifford unitary. We can consider using a non-stabiliser states as an ancilla, or performing some non-Pauli measurements to achieve this. It is usually enough to add just one of these components.

We paraphrase the following theorem:

**Theorem**: (Nebe, Rains, Sloane) Given a non-Clifford unitary $U$, the set

$$\{U, U^\dagger, CNOT, H, S\} \tag{19}$$

is approximately universal.

It means that we can approximate universality with other sets. A gate set is approximately universal if any unitary $U$ can be approximated to arbitrary accuracy by a gate sequence contained in the set.

So, by using the set above, we can efficiently simulate any other set, thus gaining approximate universality. As long we are happy with a little bit of error.

# 4 The Solovay-Kitaev theorem

Again, we paraphrase this theorem: Given a set (single qubit) of unitarieis which is approximately universal for single qubit quantum computing, we can represent an arbitrary single qubit unitary to arbitrary accuracy efficiently.

That is, if $U$ is our target unitary, $S$ is our unitary achieved as a sequence of gates, and we want

$$\|S - U\| < \epsilon \tag{20}$$

where the norm is

$$\|S - U\| = \max_{|\psi\rangle} \|S|\psi\rangle - U|\psi\rangle\| \tag{21}$$

It turns out that it suffices to use a sequence of length $\mathcal{O}\left(\log^c \frac{1}{\epsilon}\right)$ where $c \simeq 1.7$. Here, the scaling is the important thing – it shows that the process is efficient.

So we just need a unitary $U$ gate set to efficiently simulate anything.

What then, shall we use? We can make use of the so-called $T$ gate. It is defined as

$$T = \sqrt{S} \tag{22}$$

If we write

$$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix} \tag{23}$$

Then

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \tag{24}$$

It is also known as a $\pi/8$ gate because we could write

$$T = e^{i\pi/8} \begin{pmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{pmatrix} \tag{25}$$

However, the name $T$-gate is the more common jargon. This is a non-Cliford gate with many more properties. We know that

$$T^2 = S \tag{26}$$

Also, for $T$ and $T^\dagger$, it follows

$$T^\dagger = T^\dagger T^\dagger T = S^\dagger T = S \dagger S \dagger ST = ZST \tag{27}$$

So we can obtain the inverse by multiplication of other gates. This becomes very clear if we view it in the Bloch sphere.

As a result, we can conclude that the set $\{CNOT, H, T\}$ is an approximately universal set. For fault tolerance, it is very useful to have a universal gate set. In general, we can approximate any small evolution as

$$e^{iHt} \simeq I + iHt \tag{28}$$

That is, we divide it up into gates.

# 5 Fault-Tolerant Quantum Computing

We first introduce the Threshold Theorem. There exist many implementations of it, but we shall use a simple error model to demonstrate its purpose.

**Threshold Theorem**: Assume the following simplified error model: Every component in our computation, e.g. the preparation of input states, unitary gates, and measurements.

Assume that every component succeeds with probability $1-p$ and fails with probability $p$. We take this probability to be very small $p \ll 1$.

Here, fail means that the gate is followed by an arbitrary error – anything can happen. If we use quantum error correcting codes, concatenation or similar, frequent error correction, we need a fault-tolerant construction to stop errors from spreading.

We find that certain methods cause errors to spread to the extent that they multiply. We thus need to prevent single errors from turning into multiple errors.

Given all this, efficient, reliable quantum computation is possible provided

$$p < p_{th} \tag{29}$$

where $p_{th}$ is a threshold rate. Then, the errors will be suppressed through correction.

How would we do this? It turns out that we only need a code that can correct a single error. The key idea is that we use a distance 3 code that allows us to correct one error and call the set of qubits supporting the code a code block.

For example, we can take the Steane code, $n = 7, d = 3, k = 1$). Here, let 7 qubit represent a code block - this is one single encoded qubit. Then, in order to implement a fault-tolerant design, we would try and ensure that the probability of more than one error in each code block is heavily suppressed.

So fault-tolerant constructions want to design circuits such that one component failure leads to at most one error on one single qubit in one separate code block.

The key idea is the following. It means that if this is achieved, a code block will only fail to have its error corrected if two or more independent component failures occur. This notion of **independence** is very important.

If a component failure rate is $p$, then the probability of wo independent failures is $p^2$, which is indeed very small. We can do the same kind of analysis with $d > 3$ codes.

This means estimating the probability of a code block failure occuring. It will be

$$p^2 \times \text{no. of pairs or error locations} \tag{30}$$

where the second term simply means all the possible things that can go wrong. We must consider every component as an error location., e.g. the number of pairs of failed components which cause code blocks to fail.

For example, we can look at the encoded CNOT gate in the Steane code. Since the CNOT gate operates on two logical qubits, it must act on two separate code blocks. After we act with the CNOT gate, we must measure each code block for an error and then correct them individually.

Counting all pairs of errors that can occur gives us about $\sim 10^4$. If we then calculate the probability of an error occuring times the number of ways in which this can happen, we get

$$\left(10^4 \cdot 10^{-4}\right)^2 = 1 \tag{31}$$

We would want this quantity to be really small, preferably smaller than 1.

But note that our one layer of encoding has brought us from $p$ to $p^2$. We write

$$cp^2 \tag{32}$$

where $c$ denotes pairs of error locations.

# 6 Code concatenation

We replace every qubit by a code-block and every component by an encoded CNOT gate. We worked out before that we got $cp^2$ for one layer. For two layers of encoding, we gain

$$cp^2 \to c\left(cp^2\right)^2 = c^3p^4 = \frac{(cp)^4}{c} \tag{33}$$

Then, we go from $2 \to 2 \cdot 7 \to 2 \cdot 7^2$ qubits in the case of the Steane code.

We can then repeat the concatenation. After $n$ layers of encoding, we have

$$\frac{(cp)^{2^n}}{c} \tag{34}$$

Now, if $cp < 1$, this quantity gets exponentially suppressed. However, we do end up having to use $2 \cdot 7^n$ qubits for the encoding. We can show though that the error grows slower than the number of qubits.

More generally, let $d$ be the size of a code block. We then get a number of $d^n$ physical qubits and gates that scale with $d^n$. It is not too bad, considering that the error quantity scaled with $2^n$, which is much faster.

Consider then a quantum algorithm with problem size $n$ and $f(n)$ number of polynomial components. We can accept an overall error $\mathcal{E}$, where $\mathcal{E}$ must not scale with $n$. Thus, if we repeat the algorithm, we bring down the error. This is the case for e.g. Grover's unstructured search algorithm. Note that by problem size, we mean the parameter that determines the complexity of hte problem.

The encoded component error at the top layer is $p$. Then, the probability of all components succeeding for a number of $f(n)$ operations is

$$(1-p)^{f(n)} \simeq 1 - pf(n) \tag{35}$$

So here, we want $pf(n) \ll \mathcal{E}$. To explore this more closely, set $p = \epsilon$, where

$$\epsilon = \frac{(cp)^{2^n}}{c} \tag{36}$$

is the quantity we considered before. So then, for $k$ levels of encoding (we switch the name of the variable since $n$ is already taken), we get

$$\frac{(cp)^{2^k}}{c} f(n) \ll \mathcal{E} \tag{37}$$

We can solve this for $k$, see Nielsen and Chuang Section 10.6 for details. However, the overhead from encoding is $d^k$, and we can show that

$$d^k = \mathcal{O}\left(\sqrt{}\updownarrow\dagger \left(\log\left\{(\backslash)/\epsilon/\mathcal{E}\right)\left\{(\backslash)\right.\right.\right) \tag{38}$$

which is approximately $Polyl(\log n)$, so it's actually pretty good. Thus, provided we can encode everything, all our gates and all the qubits, we can have fault-tolerant computing.

However, the architecture for such a system is difficult, and the overhead is difficult.

Question: What is meant by overhead? I don't seem to have written that down.