Fault tolerant quantum computation

Guest lecture for 'Quantum Computation' MATHM0023

Dominic Verdon
School of Mathematics, University of Bristol
dominic.verdon@bristol.ac.uk

07/03/2019

In this lecture I will introduce some of the theoretical tools that are being used in attempts to build a fault-tolerant universal quantum computer.

Notation. $P \subset U(2^n)$ is the group generated by tensor products of Pauli operators $\{I, X, Y, Z\}$ on n qubits. We write $X_i, Y_i, Z_i \mid 1 \leq i \leq n$ for operations which perform Pauli X, Y or Z on qubit i and the identity elsewhere. $\mathcal{Q} := \mathbb{C}^2$ is the Hilbert space of a qubit. We will need to distinguish between logical and physical states, subspaces and operators, and so will signify 'logical' by overlining, e.g. $\overline{\mathcal{Q}}$ is the Hilbert space of a logical qubit.

Stabiliser codes. In the last lecture you learned about Shor's nine-qubit code. This encodes one logical qubit in nine physical qubits.

I'll begin by reviewing this code from a slightly different perspective: it's a *stabiliser code*. Stabiliser codes are a particularly nice class of codes which can be treated using abelian group theory. Because they are so nice to work with, most of the codes people use in practice, such as *surface codes*[1], are stabiliser codes.

Let's define the code space $\overline{\mathcal{Q}} \subset \mathcal{Q}^{\otimes 9}$ again, this time in stabiliser language. In the last lecture, we saw expressions for the logical computational basis states $\{\overline{|0\rangle}, \overline{|1\rangle}\} \in \mathcal{Q}^{\otimes 9}$, and the code space $\overline{\mathcal{Q}}$ was defined as the linear span of these. However, we can define the code space just in terms of the operators in P which measure the error syndrome. If you check your notes from last time you will see that, for the nine-qubit code, these are:

[1] Surface codes are a fascinating and widely used type of topological stabiliser code. See e.g. [4].

$$\begin{array}{c|ccccccccc}
S_1 & Z & Z & I & I & I & I & I & I & I \\
S_2 & Z & I & Z & I & I & I & I & I & I \\
S_3 & I & I & I & Z & Z & I & I & I & I \\
S_4 & I & I & I & Z & I & Z & I & I & I \\
S_5 & I & I & I & I & I & I & Z & Z & I \\
S_6 & I & I & I & I & I & I & Z & I & Z \\
S_7 & X & X & X & X & X & X & I & I & I \\
S_8 & X & X & X & I & I & I & X & X & X
\end{array}$$

You can easily check that these all commute — this is good news because otherwise they wouldn't be simultaneously measurable, and so our error syndrome would depend on the measurement order! Because they commute, they generate an abelian group $S < P$. Obviously, the eigenstates for the generators are the eigenstates for any operator in this group, and their eigenvalues for any operator in the group are completely determined by the eigenvalues for the generators.

Definition 0.1. An n-qubit *stabiliser code* is defined by a *stabiliser* — an abelian subgroup $S < P$ not containing $i\mathbb{1}$ or $-\mathbb{1}$.[2] The *code space* is the joint eigenspace of S where all operators have $+1$ eigenvalues.[3]

We already know that the code space is 2-dimensional, but we can see this just from the generators $\{S_1, \ldots, S_8\}$. They are all independent, and have eigenvalues ± 1; each therefore divides $\mathcal{Q}^{\otimes 9}$ into a different pair of eigenspaces. The eigenspaces of S are the intersections of these, so are $2^9/2^8 = 2$-dimensional. Therefore, our code space $\overline{\mathcal{Q}}$ is 2-dimensional, and can be used to encode a qubit.[4]

Definition 0.2. For any stabiliser $S \subset P$, the size of the associated code space is 2^{n-r}, where r is the size of any independent generating set of S.[5]

How many errors does this code correct? In the last lecture we saw that it certainly corrects for all single-qubit errors. Can we see this just by considering the stabiliser?

We saw in the last lecture that, for a correction operation to correct for an error $aE + bF$, it is sufficient for it to correct both E and F, since by linearity it will act on all states in the superposition. We therefore need only consider a *basis* of error operators. In the setting of stabiliser codes, P is the obvious choice, with the advantage that all error operators are unitary.

Firstly, note that any two operators $x_1, x_2 \in P$ either commute ($x_1 x_2 = x_2 x_1$) or anticommute ($x_1 x_2 = -x_2 x_1$). Let $|\psi\rangle$ be an eigenvector of x_1 with eigenvalue $\lambda_\psi \in \{\pm 1\}$. Then, if x_1, x_2 commute, then

$$x_1 x_2 |\psi\rangle = x_2 x_1 |\psi\rangle = \lambda_\psi x_2 |\psi\rangle,$$

[2]This last condition is just there to avoid redundancy and make the mathematics nicer.

[3]In fact we could use any eigenspace as the code space, but this choice keeps things concrete.

[4]If this intuitive argument didn't satisfy you, there is an automorphism of the Pauli group that maps the generators of S to $\{X_1, \ldots, X_8\}$; the dimension of the code space is then obvious.

[5]Well-definedness of r follows from the classification of finite abelian groups.

and so x_2 preserves the eigenspaces of x_1. On the other hand, if x_1, x_2 anticommute, then

$$x_1 x_2 |\psi\rangle = -x_2 x_1 |\psi\rangle = -\lambda_\psi x_2 |\psi\rangle ,$$

and so x_2 swaps the two eigenspaces of x_1.

This means that every operator x anticommuting with some element of the stabiliser $S < P$ takes the code space to a different joint eigenspace of the operators of S. Errors not commuting with all elements of S — in group theory language, errors not in the *centraliser* of S in P, $Z_P(S)$ — can therefore always be detected by measuring the generators of S.

In fact, since $-\mathbb{1} \notin S$, we have that $Z_P(S) = N_P(S)$, where $N_P(S)$ is the *normaliser* of S in P:

$$N_P(S) = \{x \in P \mid xSx^{-1} = S\}$$

The dangerous error operators — the ones we don't want to occur — are those in $N_P(S)$ but not in S. These will preserve the code space — and therefore be undetectable — but act on it nontrivially. We will see later that we can make use of these operators to perform logical operations, but we certainly don't want them to occur without us knowing about it.

So, we can certainly *detect* any error operator which is in P but not in the normaliser $N_P(S)$, but can we *correct* for these errors? Given some error $x \in P$, it will map the code space $\overline{\mathcal{Q}} \subset \mathcal{Q}^{\otimes 9}$ to some other eigenspace $x\overline{\mathcal{Q}} \subset \mathcal{Q}^{\otimes 9}$. If it is the only error which maps to this eigenspace, then we can easily correct by performing $x^\dagger$ after measuring the syndrome. However, if there are two errors $x_1, x_2 \in P$ which both map to $x\overline{\mathcal{Q}}$, we need to be able to perform just one correction operator $x^\dagger$ for both, since we won't know which occured. In order that the action on the code space after correction is trivial, we need that both $x^\dagger x_1 \in S$ and $x^\dagger x_2 \in S$; this is true iff x_1 and x_2 are in the same left coset of S; which is true iff $x_1^\dagger x_2 \in S$. For a family of errors $\{x_i\}, x_i \in P$ to be correctable, then, the necessary and sufficient condition is that

$$x_i^\dagger x_j \in S$$

for any x_i, x_j such that $x_i\overline{\mathcal{Q}} = x_j\overline{\mathcal{Q}}$. So, how does this work out for the 9-qubit code?

- The gates in $\mathcal{N}_P(S)/S$ all act on three or more qubits.
- For any pair of two-qubit gates x_i, x_j such that $x_i\overline{\mathcal{Q}} = x_j\overline{\mathcal{Q}}$, we have that $x_i^\dagger x_j \in S$.

It follows that the 9-qubit code actually corrects any error on *two* qubits. It's instructive to check this for single-qubit errors.

Exercise 0.3. Prove using the above techniques that single-qubit errors for the nine-qubit code are all correctable.

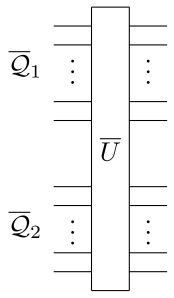
Fault-tolerant quantum computation. We can encode one logical qubit $|\psi\rangle$ in nine physical qubits so that, if any error on two of those nine qubits occurs, we can perform a measurement and unitary correction to recover $|\psi\rangle$. This is useful for building a quantum memory, or transferring a quantum state through a noisy quantum channel. However, for universal quantum computation, it's not enough. We additionally need to:

1. Perform gates from some universal gate set on our logical qubits.
2. Perform projective measurements on our logical qubits.
3. Initialise logical qubits in the $|0\rangle$ state.

We can't decode our logical qubits, because as soon as we do that they will be vulnerable to errors again; in other words, we want to perform operations at the logical level mediated by fault-tolerant physical operations. Here we'll talk about how to perform a universal logical gate set fault-tolerantly, under the simplifying (but somewhat unrealistic) assumption that all noise occurs in between operations.[6]

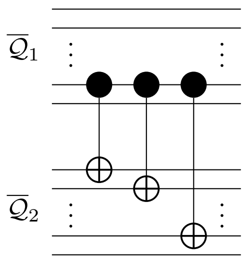
Error propagation. *Fault-tolerance* of operations is about limiting *error propagation*. Error propagation happens when an error on a single physical qubit is spread to other physical qubits by an operation.

Suppose we want to perform a 2-qubit logical gate:



We call the group of physical qubits corresponding to a single logical qubit a *code block*, or just a *block*. Suppose that our chosen physical implementation of this gate involves three CNOT gates from a single qubit in block 1 to different qubits in block 2:

[6]Essentially the same techniques work when the operations are also noisy (see e.g. [6]).



Now suppose there is an error on the control qubit in block 1 going into the logical gate. Following the logical gate, this error will now have spread to three physical qubits in block 2; this error may not be correctable. It may be possible to resolve this problem by running error correction extremely frequently (immediately before each CNOT, for instance), but this will make our computation inefficient. It is better to implement our logical operations so that this uncontrolled error propagation cannot occur.

Definition 0.4. A physical implementation of a logical operation is *fault-tolerant* if a single error in a single block going into the operation causes at most one error in each block coming out of the operation.

The *threshold theorem* [2] tells us that, when fault-tolerant operations are used, errors can be corrected efficiently as long as the error rate is below a certain threshold. To make sure our logical operations are fault tolerant the following conditions are sufficient:

- They shouldn't involve physical gates between pairs of physical qubits in the same block.
- They shouldn't involve physical gates between a given qubit in one block and multiple qubits in another.

Physical operations satisfying the above conditions are called *transversal*.

Fault-tolerant logical gates. We will now sketch how a universal set of logical gates can be implemented fault-tolerantly. The universal gate set we will use is called *Clifford + T*. It is made up of the Clifford group $\mathcal{C}$, generated by H, CNOT and

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

and the single qubit gate

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

First note that we haven't even picked a logical computational basis yet — we just specified the code space. In keeping with the stabiliser philosophy, where we define everything in terms of operators, let's define transversal logical Pauli operators on our code space. The computational basis will then be given as the eigenvectors of logical Pauli Z.

We already saw when considering correctable errors that the operators in $\mathcal{N}_P(S)/S$ induce nontrivial maps from the code space to itself; it is therefore a group of nontrivial logical operators on the code space. But which group is it?

For any stabiliser with generators $\{S_1, \ldots, S_k\}$, I claim that there's an automorphism of P which takes $\{S_1, \ldots, S_k\}$ to $\{X_1, \ldots, X_k\}$.[7] Automorphisms map normalisers to normalisers, but we already know what the normaliser of $\langle X_1, \ldots, X_k\rangle$ is; it's generated by $\{X_1, \ldots, X_k\}$ and all the Paulis on the other qubits. Therefore the group $\mathcal{N}_P(S)/S$ is isomorphic to the Pauli group on $n-k$ qubits, P_{n-k}.[8] This group is represented in the usual way on the code space; we can therefore pick some identification of the logical operators in $\mathcal{N}_P(S)/S$ with the Pauli operators based on their commutation relations. These logical Pauli operators are obviously transversal because they're tensor products of single qubit operators. For the nine-qubit code, for instance, the single-qubit logical Pauli operators can be taken as

$$\overline{X} = X_1X_2X_3 \qquad \overline{Z} = Z_1Z_4Z_7.$$

You can check that these preserve the code space and obey the Pauli commutation relations.

So, we can implement logical Pauli operators, but what about the rest of the Clifford group? The logical Clifford operators depend on the code we are considering. The Clifford operators are precisely those which preserve the Pauli group under conjugation:

$$\mathcal{C} := \{x \in U(2^n) \mid xPx^{-1} = P\}$$

The action of a logical Clifford operator on logical qubits is completely determined by how it permutes the logical Pauli operators under the conjugation action. Although a more general approach to constructing the logical Clifford operators is given in [5, Section 5.5], one simple approach is trial and error: pick transversal physical operations on one or two blocks preserving the code space, and determine how these operations permute the logical Pauli operators corresponding to the blocks, in order to determine which Clifford gate they perform.

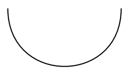
Exercise 0.5 (Original research?)**.** In [5, Section 5.3] there are lots of examples of transversal logical gates for 5- and 7-qubit codes, but none for the 9-qubit code. Find a logical Clifford gate outside of the Pauli group on the 9-qubit code by performing transversal physical CNOTs and swap gates [5, Section 5.3] on one or two blocks of the 9-qubit code, so that the code space is preserved.

[7] To prove this, we view the Pauli group as a *symplectic vector space* over $\mathbb{Z}_2$, where the generators of S are orthogonal vectors, and use the fact that the symplectic group acts transitively on orthogonal bases. See [7] for a nice exposition.

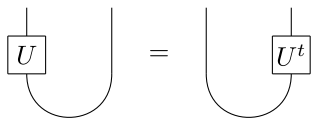
[8] Note that the automorphism we used is non-unique, so for general stabiliser codes there may be different ways to define this basic Pauli structure on your code space.

Magic states and gate teleportation. When we've constructed our fault-tolerant logical Clifford gates, we finally need to construct fault-tolerant logical T. However, there is a problem: it has been proved that it is not possible to implement a universal gate set transversally [3, Corollary 1]. Since we just implemented all the Clifford gates transversally, for T we need to use a different approach involving *magic states* and *gate teleportation.* We will now sketch how this works. In what follows all the operations take place at the logical level, and the diagrams should be read from bottom-to-top, rather than from left to right.

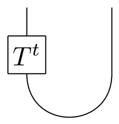
Let



be the maximally entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ of two qubits. You can check yourself that, for any unitary $\overline{U}$,



Suppose that we can initialise the state



.

We call this a *magic T-state*, and we're going to use this state as a resource[9] to perform a T gate on another logical qubit using only Clifford operations and projective measurement.[10]

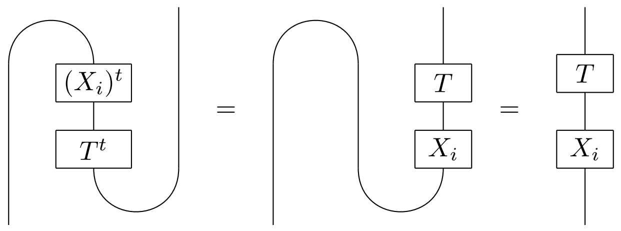
First, note that T is in the *second level of the Clifford hierarchy.* This means precisely that it takes Pauli gates to Clifford gates under conjugation:

$$TPT^\dagger \subset \mathcal{C} \tag{1}$$

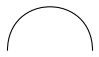
We take the qubit on which we want to perform T, and measure it with the first half of the T state in the maximally entangled basis $|\phi_i\rangle := \frac{1}{\sqrt{2}}(\mathbb{1} \otimes X_i^*)(|00\rangle + |11\rangle)$, where $X_0 := I, X_1 := X, X_2 := Y, X_3 := Z$ are the Pauli matrices. Suppose we measure $|\phi_i\rangle$. Then the resulting state of the other half of the magic T-state is

[9]Creating magic states at the logical level is a major bottleneck in quantum computation. Huge numbers of qubits in experimental prototypes of quantum computers are dedicated to this task, assembled together in so-called *magic state factories.* The field of *quantum circuit optimisation* aims to cut down the number of T or other non-Clifford gates required to perform an operation, reducing the number of magic states required.

[10]I didn't show that we can do this projective measurement fault-tolerantly at the logical level, but we can; see e.g. [5].



where



is the effect $\frac{1}{\sqrt{2}}(\langle 00| + \langle 11|)$, and you can check the second equality yourself.[11]

So, after measuring i we know we have the state $TX_i|\psi\rangle$, where ψ was the state we started off with. But $TX_i|\psi\rangle = TX_iT^\dagger T|\psi\rangle$, and, by (1), $TX_iT^\dagger$ is in the Clifford group! Moreover it's self inverse, and we only have to perform the Clifford correction $TX_iT^\dagger$ to get the state we wanted, $T|\psi\rangle$.

We have therefore performed a logical T gate using only a logical magic T-state, logical Clifford operations, and logical projective measurement in the computational basis.

References

[1] Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*, pages 415–425. IEEE, 2004. `arXiv:quant-ph/0402130`, `doi:10.1109/LICS.2004.1319636`.

[2] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 176–188, New York, NY, USA, 1997. ACM. `arXiv:quant-ph/9906129`, `doi:10.1145/258533.258579`.

[3] Bryan Eastin and Emanuel Knill. Restrictions on transversal encoded quantum gate sets. *Physical Review Letters*, 102(11):110502, 2009. `arXiv:0811.4262`, `doi:10.1103/PhysRevLett.102.110502`.

[4] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012. `arXiv:1208.0928`.

[5] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction.* PhD thesis, California Institute of Technology, 1997. `arXiv:quant-ph/9705052`.

[11] To understand where this nice yanking notation comes from, see [1].

[6] Daniel Gottesman. *Quantum Information Science and Its Contributions to Mathematics: American Mathematical Society Short Course, January 3-4, 2009, Washington, DC*, volume 68 of *Proceedings of symposia in applied mathematics*, chapter An introduction to quantum error correction and fault-tolerant quantum computation, pages 13–58. American Mathematical Society, 2010. `arXiv:0904.2557`.

[7] Jeongwan Haah. Algebraic methods for quantum codes on lattices. *Revista Colombiana de Matemáticas*, 50(2):299–349, 2016. `arXiv:1607.01387`.