CS269Q Final Project: Surface Code Implementation

William McCloskey

June 28, 2019

1 Introduction

This project provides an implementation in NetworkX of the error-free syndrome extraction surface code and calculates its error threshold. We also provide a Jupyter Notebook along with this paper as an instructive resource resource to help others better understand surface codes. Finally, we implement the surface code simulation (without correction) in PyQuil as a proof of concept.

2 Surface Codes

We first give a short introduction to surface codes. Surface codes are one of the leading ideas for quantum memory. A surface code allows a single logical quantum bit of information to be maintained despite noisy individual qubits. The surface code uses a square lattice layout of qubits to maintain the state of this logical qubit. As random errors affect the individual qubits in the lattice, the surface code seeks to measure these errors and correct them as they occur. More specifically, the layout of qubits is given in the following picture:

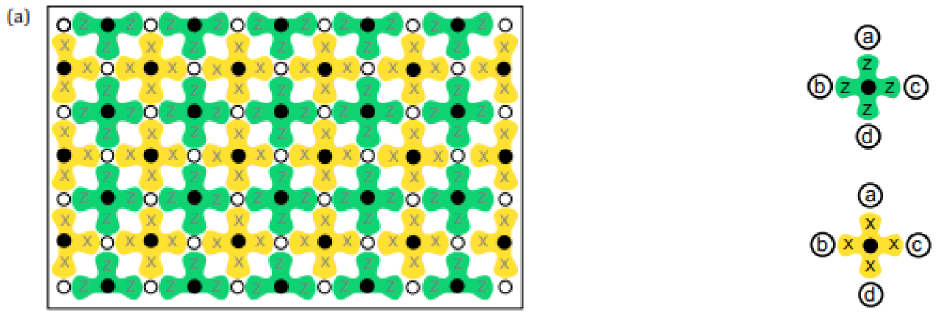


Figure 1: Surface code lattice diagram. (Taken from [1].)

Each dot in the diagram represents a qubit. The white dots represent *data* qubits, and the black dots represent *measurement* qubits. The persistent logical qubit lives in the white dots. We want to correct errors that arise in the white dots that might affect the state of the logical qubit. The black dots measure the surrounding data qubits for errors. Specifically, the green-z black dots measure their surrounding data qubits for bit-flip errors, and the yellow-x black dots measure their surrounding data qubits for phase-flip errors.[1] Specifically, they measure respectively the operator $Z_a \otimes Z_b \otimes Z_c \otimes Z_d$ (where a, b, c, d are the surrounding qubits) and $X_a \otimes X_b \otimes X_c \otimes X_d$ (where a, b, c, d are the surrounding qubits– sometimes there are only three on the boundary). Handling bit-flip errors and handling phase-flip errors are analogous tasks, so we only need to study bit-flip errors. Throughout, we will assume that the probability that a qubit undergoes a bit flip during a given timestep is i.i.d. with probability p.

Correcting bit-flip errors is not as easy as measuring the flipped qubits and flipping them again. A green-z measurement only detects a bit-flip error if there are an odd number of surrounding bit-flip errors. The set of green-z measurements that detect a bit-flip error is called the error syndrome. Many different sets of bit-flip errors may cause the same error syndrome, so it may seem impossible to correct them all.

Luckily, one does not have to unflip all the bits individually. The reason lies in the relationship between the green-z and yellow-x operators ($Z_a \otimes Z_b \otimes Z_c \otimes Z_d$ and $X_a \otimes X_b \otimes X_c \otimes X_d$) and the maintained logical qubit $|\psi\rangle$. The green-z and yellow-x operators are called stabilizers. (This term is from group theory.) They are called stabilizers because they fix the logical qubit $|\psi\rangle$, i.e. $S|\psi\rangle = |\psi\rangle$ for any operator S in the stabilizer. This property is desirable because we do not want our measurements to disturb the maintained logical qubit $|\psi\rangle$. One can show that the stabilizers of $|\psi\rangle$ form a group, and that the green-z and yellow-x operators generate this group. Noting that $Z^2 = X^2 = I$, we see that as a result that cycles of bit-flip errors and cycles of phase-flip errors are in the stabilizer. (See Figure 2.)

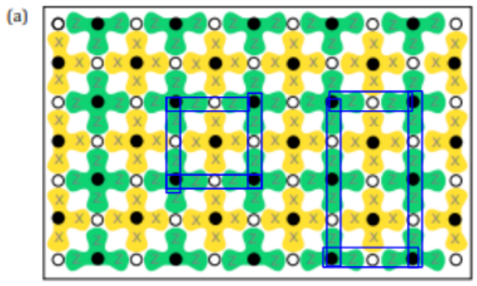


Figure 2: Bit flips at the white dots in the left cycle is just a yellow-x operator. Bit flips at the white dots in the right cycle is a product of the upper yellow-x operator and the lower yellow-x operator contained in the cycle $(X_a \otimes X_b \otimes X_c \otimes X_d) \cdot (X_c \otimes X_e \otimes X_f \otimes X_g) = X_a \otimes X_b \otimes X_d \otimes X_e \otimes X_f \otimes X_g$. (Note $X_c^2 = I$.) Because $X_a \otimes X_b \otimes X_d \otimes X_e \otimes X_f \otimes X_g$ is a product of stabilizer elements, the bit-flips in this cycle leave the maintained logical qubit state $|\psi\rangle$ unchanged.

Thus, as long as we complete the flipped bits to an element in the stabilizer – for example to one of the bit-flip cycles in Figure 2 – the underlying logical qubit state $|\psi\rangle$ is preserved.

[1]Bit-flip errors are errors at qubits where the bit-flip gate X is unintentionally applied, and phase-flip errors are errors at qubits where the phase-flip gate Z is unintentionally applied.

The bit flip error correction fails if it creates a logical bit-flip. Logical bit flips are any sequence of bit flips from the left boundary of the lattice to the right boundary of the lattice with no branches. (See Figure 5.) Any such logical bit-flip acts as a bit-flip on the underlying state. (The logical phase-flip is analogous: replacing bit flips with phase-flips and traveling from the top boundary to the bottom.)

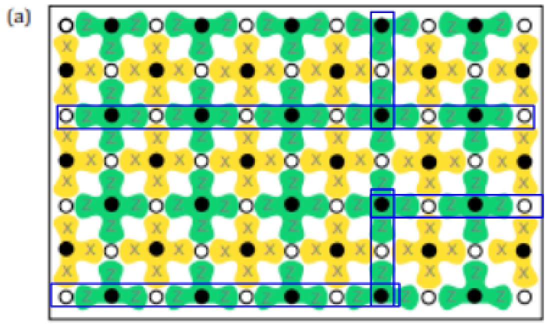


Figure 3: The operator defined by bit flips at the white dots in the top path is not a logical bit flip. The operator defined by the bit flips at the white dots in the bottom path is a logical bit flip.

In summary, in order to correct a set of bit flip errors, we measure them using the green-z operators. The set of green-z operators that detect a bit flip is called the error syndrome. The error syndrome gives us an idea of which data qubits may have undergone bit flip errors, but it doesn't tell us exactly. Given the error syndrome, we use more bit flips to make corrections. We hope that these bit flips, in combination with the bit-flip errors, combine create elements in the stabilizer so that the maintained logical qubit $|\psi\rangle$ remains unchanged. We fail if we create an odd number of logical bit flips.

The most used error correction method (and the one we use in our simulations) is to pair elements in the error syndrome that minimizes the distance between them. (One can also pair with a boundary element.) This pairing makes it likely that elements in a pair are endpoints of the same path of bit flips. Then one flips bits in a path from one element of the pair to the other, in the hope of forming a loop (element of the stabilizer). Empirically, this heuristic is close to optimal. For more details, see [2] or Appendix A of [3].

3 Simulation

One of the main goals of this project was to reproduce the threshold probability calculations in [2]. The threshold probability p_{th} is the probability of a bit flip error at which making a larger lattice size hurts error correction rather than helps. That is, if $p < p_{th}$, then making a larger lattice size reduces the probability of a logical bit flip error, and if $p > p_{th}$ then making a larger lattice size reduces the probability of a logical bit flip error. The optimal threshold has been calculated at about $p_{th} = 0.11$. (This quantity was previously originally calculated by studying the Ising model from statistical mechanics, which is analogous to the surface code.) For our case, following other papers, we simulated under the 'uncorrelated noise model', so our probability of a bit-flip error is $2p/3$. Thus the threshold for us is $p_{th} = 0.165$.

For our simulations, we build a graph where the nodes are green-z operators and the edges represent bit-flip errors at qubits between green-z operators. The blue nodes are boundary nodes, which exist to allow for bit-flip errors at the boundary of the lattice. So blue nodes cannot have edges between them. (Otherwise, any pair of adjacent nodes that contains a green node can have an edge.)[2]

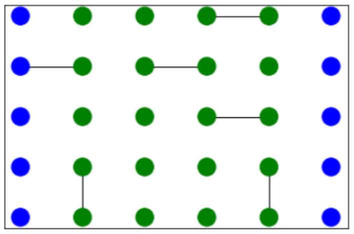


Figure 4: Graph representation.

At each step, the errors occur with probability p. They are then corrected using the algorithm from section 2. We measure the expected number of timesteps to failure using Monte Carlo methods. The results from the paper [2] are below. The distance means how many vertical green nodes are in the graph.

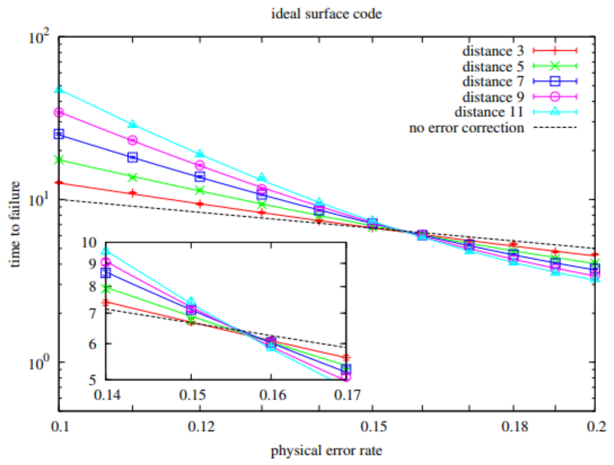
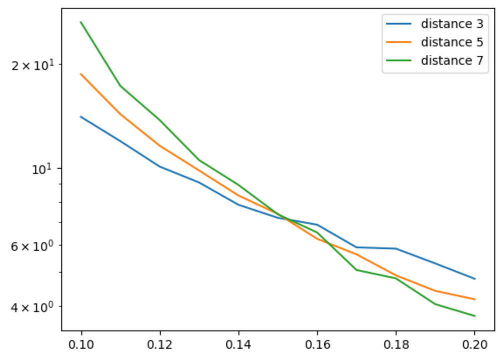


Figure 5: Simulation results from [2]. The distance is the number of green-z operators in the vertical dimension.

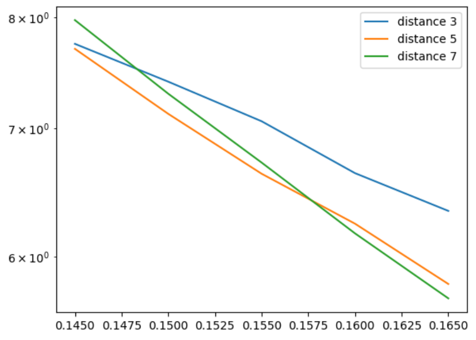
[1]Here, we assume our measurements are perfect – that we know the error syndrome exactly. Of course that is not the case in the real world, but these assumptions allow us to compute an upper bound on the threshold in the most ideal case. The threshold probability with imperfect measurements is on the order of 0.01.

[2]The astute reader might notice that Figures 2 and 3 do not match the dimensions of our graph. This is because Figures 2 and 3 do not show the whole lattice. A full picture of the lattice in the style of Figures 2 and 3 can be found on page 9 of [1]. (We did not use this more complete picture because it has other markings.)

Our results match fairly closely, although we used less simulation time than they did so we are a bit less accurate. They estimated the threshold probability p_{th} to be about $p_{th} = 0.155 \pm 0.005$, not $p_{th} = 0.165$, because we both use a slightly suboptimal (though practical) heuristic to compute error codes, not the optimal decoder.



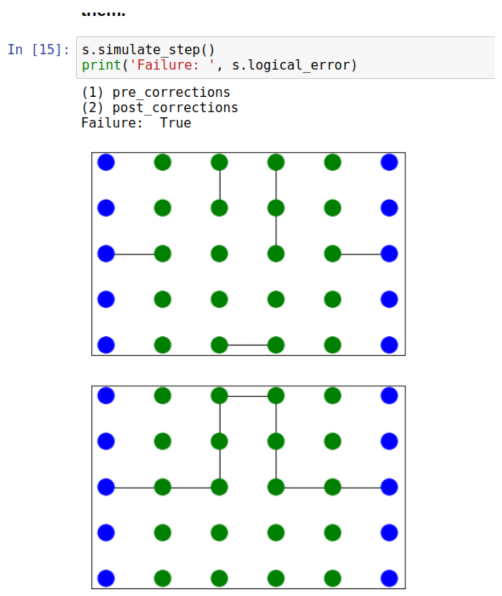
(a) Large view.



(b) Zoom on threshold with more samples.

Figure 6: Simulation results.

We also include a Jupyter notebook with a cell that can be run repeatedly to illustrates the corrections and whether failure occurs.



```
In [15]: s.simulate_step()
         print('Failure: ', s.logical_error)
```

```
(1) pre_corrections
(2) post_corrections
Failure:  True
```

Figure 7: Jupyter output

4 PyQuil Simulation

Finally, we implemented the code without correction in the Python quantum computing package PyQuil. This is quite similar to the NetworkX simulation, except it is much more representative of how a real quantum computer would be programmed. To measure the green-z and yellow-x operators, the program uses the gates in [1].

5 Code

The code can be found here: https://github.com/mcclow12/surface_code_simulation. The code to run the simulation is in main.py.

References

[1] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. 2012.

[2] D. S. Wang, A. G. Fowler, A. M. Stephens, and L. C. L. Hollenberg. Threshold error rates for the toric and surface codes. 2009.

[3] Shota Nagayama, Austin G. Fowler, Dominic Horsman, Simon J. Devitt, and Rodney Van Meter. Surface code error correction on a defective lattice. 2016.

[4] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, 2002.