Stochastic Processes in Quantum Error Correction

Reagan Choi

Under the direction of

Andrey Boris Khesin
MIT Mathematics Department

Research Science Institute
August 2, 2021

Abstract

We analyze surface codes in quantum error correcting. In these codes, qubits are encoded with a grid of cells, which may be affected by error. These errors cannot be detected directly; rather, we check the stabilizers of the encoding, which correspond to edges on the grid. This allows us to find the loops that surround the errors. We analyze the behavior of various processes that correct the errors on these loops. The absolute zero process is the most stable, and we run simulations to determine that it can correct a square loop of error in an average time of $O(n^3)$. We prove an upper bound for the absolute zero process and prove that the average time complexity of an altered process is $\Theta(n^3)$. Then, we analyze probabilistic algorithms. The behavior shown by the simulation of the probabilistic model indicates that there is a critical probability, approximately 0.175, at which error cannot reliably be corrected. We also analyze the heat bath algorithm, which can introduce errors to the grid but stochastically corrects large errors as long as the temperature is sufficiently small.

Summary

When messages are sent in quantum computing, random errors may occur, making the messages hard to read. To correct these issues, we encode each quantum bit (qubit) as a grid of qubits, some of which may have errors. We analyze a process that randomly adds and removes errors to this grid, with the ultimate goal of removing all of the error. As the process selects a random cell every turn, it does not need to know information about the entire grid, but rather the immediate surroundings. We analyze the time taken for these random processes to correct all of the error on a grid, which is useful in determining how quickly errors in quantum messages can be corrected using a grid encoding.

1 Introduction

Quantum computation is the use of quantum mechanics to store information. The basic unit of data in quantum computing is the quantum bit (qubit). In the computational basis, the two qubits are $|0\rangle$ and $|1\rangle$. Linear combinations, also known as *superpositions*, of these qubits are of the form $\alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$ for complex numbers α and β.

Quantum computation can store more information than classical computation. For example, in quantum computation, when there are N bits, there are 2^N combinations of qubits, ranging from $|00\ldots0\rangle$ to $|11\ldots1\rangle$, which means there are 2^N possible coefficients, each storing information. The complications of quantum computation arise from trying to extract the coefficient α; when a qubit is measured, it collapses into one possible qubit with a probability of $|\alpha|^2$, making it difficult to find α.

In addition, matrices can operate on qubits, transforming them linearly. Four important matrices are known as the Pauli matrices, which are X, Y, and Z. The Pauli matrices satisfy the property that $X^2 = Y^2 = Z^2 = I$ and $Y = iXZ$ [1]. These matrices are central to quantum error correction as any continuous error can be represented in terms of these matrices. The operations are also simple: the operator X transforms the qubit $\alpha|0\rangle + \beta|1\rangle$ into $\beta|0\rangle + \alpha|1\rangle$, and the operator Z transforms the qubit $\alpha|0\rangle + \beta|1\rangle$ into $\alpha|0\rangle - \beta|1\rangle$.

When a quantum message is sent through a noisy channel, there is some probability that an error is inflicted on each qubit. These errors can be corrected through error-correcting codes, which encode each qubit as multiple different qubits; this redundancy helps correct any single error that may occur [2].

Imagine that Alice and Bob are communicating using qubits; i.e. Alice wants to send a stream of qubits to Bob. However, there is noise in the channel, which has a probability p of performing a random Pauli matrix operation on each qubit. This is known as the quantum noisy coding problem, to which the optimal encoding system is not known yet [3].

We analyze *surface codes*, which encode each qubit as multiple qubits on a grid, as in Figure 1.

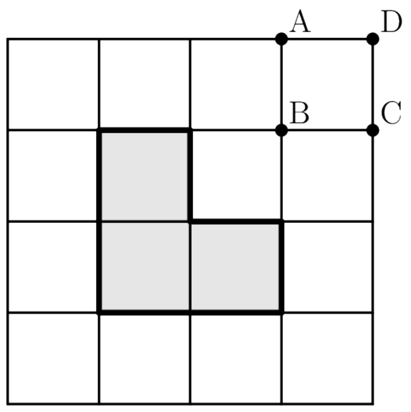

Figure 1: Qubits with uncorrected errors on a grid

We define a *cell* as a unit square on a unit lattice. For example, in Figure 1, square $ABCD$ is a cell. An *edge* is any unit segment along the lattice between integer points; each edge borders two cells. This includes the edges AB, BC, CD, and DA. Finally, any *vertex* is a lattice point, which joins four cells and four edges; these include A, B, C, and D.

Some of the cells may have *error* that need to be corrected. In Figure 1, errors are shaded in light gray. We define a *boundary line* as an edge between two cells of which exactly one has error; these are bolded in the figure. As boundary lines separate cells with error from cells without error, the boundary lines must form *loops* on the grid, each surrounding a droplet of cells with error, as evident in Figure 1.

A stochastic process that toggles a cell based on the number of boundary lines can be used to correct errors on these cells. Each step, this process either corrects or introduces an error to a cell. Since $X^2 = Z^2 = I$, each step applies an X or Z operation to the qubit on a cell, which can correct or introduce an error. Although both X and Z errors are possible, we only consider one at a time, as the correction processes are equivalent and independent.

Dennis, Kitaev, Landahl, and Preskill demonstrate one algorithm, called the *heat bath algorithm*, which stochastically modifies the loop [4]. In this algorithm, a set of non-overlapping cells on a grid are randomly chosen. Then, for some inverse temperature β, this algorithm flips the cell depending on how many boundary lines it has. A cell with k boundary lines is flipped with the probability

$$p_k = \frac{1}{1 + e^{(4-2k)\beta}}.$$

As a consequence, $p_0 + p_4 = p_1 + p_3 = 2p_2 = 1$.

Each step, the heat bath algorithm can *flip* several cells — that is, either a cell with error turns into a cell without error, or vice versa. For our purposes, we simplify the heat bath algorithm to select one cell at a time. The only difference this creates is that our algorithm has a cell selected each step independently, as non-overlapping cells do not influence each other.

In our paper, we study the heat bath algorithm and several simplified algorithms. We simulate the behavior of these algorithms and their ability to correct error cells on a grid. We also prove the time complexity of the expected number of steps for the absolute zero process, as defined in 2.1, to correct error.

2 Simulation of the Absolute Zero Process

We first analyze a special case of the heat bath algorithm for simplicity. We first consider the case where $\beta \to \infty$, which means $p_0 = p_1 = 0$ while $p_3 = p_4 = 1$. Because the temperature is 0 in this case, we name this the *absolute zero process.* The process ends when the set of cells is empty.

Definition 2.1. In the *absolute zero process*, we begin with a finite number of cells with error on an infinite grid. Each step, the following occurs:

1. If there exists a cell with four boundary lines, flip one such cell at random.

2. Otherwise, if there exists a cell with three boundary lines, flip one such cell at random.

3. Otherwise, randomly select a cell with two consecutive boundary lines and flip it.

This process is repeated until there are no more boundary lines, at which point no cells have error. As long as there is at least one error, it is always possible to flip a cell; we show this in Lemma 3.1.

The motivation behind the absolute zero process is simple. The goal of this process is to correct all of the error on the grid. However, it is impossible to directly determine if a given cell has error; instead, stabilizers compare two adjacent cells to determine if exactly one of them have error. The absolute zero process uses a probabilistic method to correct the errors, based solely on the number of boundary lines that cells have.

For the sake of simplicity, we also forgo the probabilistic element of selecting a random cell. This means we actively seek cells with three or four boundary lines to flip. This simplified process is not feasible in quantum computing, but the results we obtain can be extended to the heat bath algorithm, which is a real process in quantum computing. Another simplification we make is disallowing the flipping of any cells with two *opposite* boundary lines, which means they are two parallel sides of the cell. We will consider these flips in Section 3.

We perform a Monte Carlo simulation of the expected number of steps for the absolute zero process to correct an $n \times n$ grid of cells with error for several values of n. Figure 2 shows a graph plotting the number of steps for a square loop to vanish for assorted values of n.

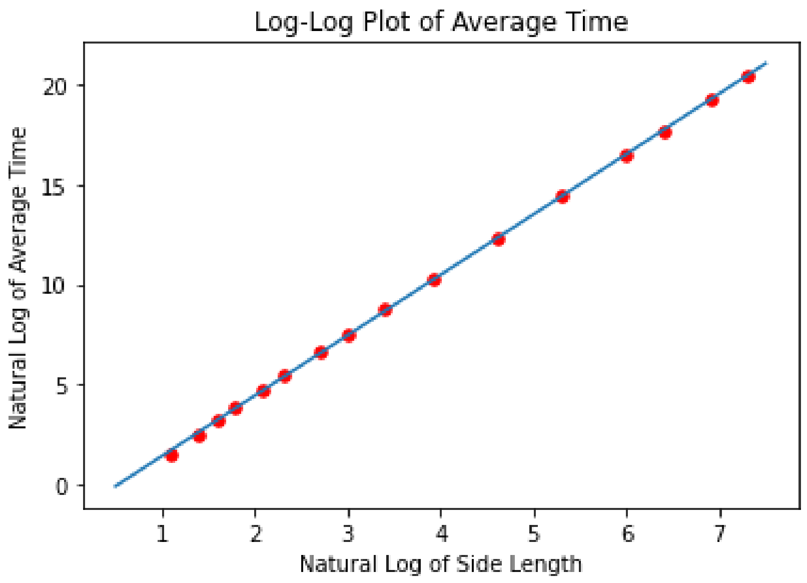

Figure 2: A log-log plot of the simulated number of steps taken for an $n \times n$ grid of cells with error vanish with the absolute zero process

Figure 2 is a log-log plot of the average number of steps taken. The linear regression line for this plot is

$$\ln T \approx 3.02 \ln n - 1.60.$$

The equation has an r^2 value of 0.9998 and a standard error of 0.084. As a result, the slope of the line does not differ significantly from 3. Because the equation $\ln T = 3 \ln n - c$ is equivalent to $T = e^c n^3$, the number of steps the absolute zero process takes to correct an $n \times n$ grid of error has a cubic time complexity.

We also observe the shape of the loop during the absolute zero process. As seen in Figure 3, the loop maintains a roughly circular shape.

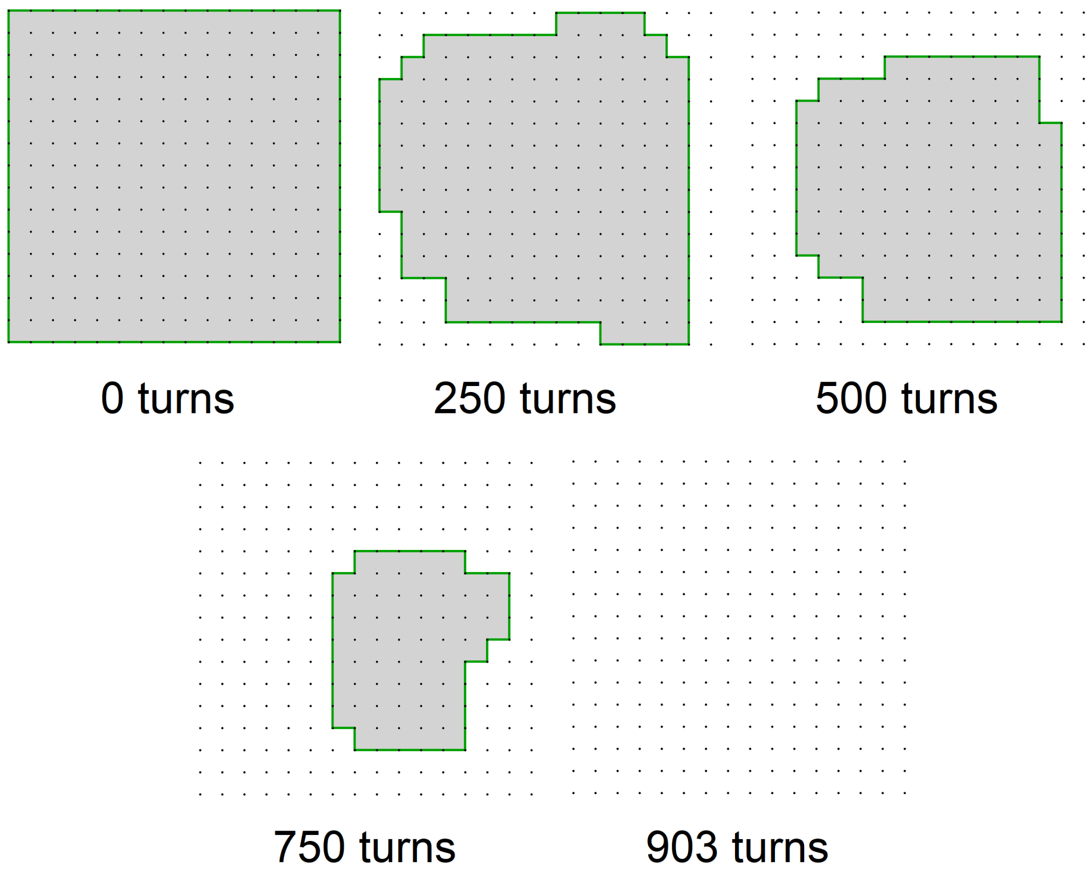

Figure 3: A sample simulation of the absolute zero process. The loop begins as a 15 × 15 grid and vanishes in 903 steps; the loop after every 250 steps is shown.

3 Analysis of the Absolute Zero Process

Standard Absolute Zero Process

We first make several claims about the loop in the absolute zero process, motivated by the computational analysis in Section 2. We begin by proving a claim made in Definition 2.1.

Lemma 3.1. *If the loop is finite and there is at least one cell with error, there is at least one cell with two or more boundary lines.*

Proof. Because there are a finite number of cells with error, by the extremal principle, we consider the cell (x, y) with the largest x-coordinate. If there are multiple such cells, consider the one with the largest y-coordinate.

Because (x, y) is extremal, neither of the cells $(x + 1, y)$ nor $(x, y + 1)$ can have error. This means there is a boundary line between (x, y) and $(x + 1, y)$, and also between (x, y) and $(x, y + 1)$, so (x, y) has at least two boundary lines. This proves our lemma. □

Next, we analyze how the absolute zero process tends to remove error from the grid.

Definition 3.1. We define the *bounding box* of a finite region S on a square lattice as the smallest rectangle, whose sides are parallel to the grid lines, that contains S.

The bounding box of a region S is simple to find. If the bounding box has opposite vertices (x_1, y_1) and (x_2, y_2) with $x_1 \leq x_2$ and $y_1 \leq y_2$, it is clear that x_1 must be less than or equal to the x-coordinate of all points in S. This means

$$x_1 = \min_{P \in S} x(P),$$

where P covers all points in S.

Similarly, we have

$$y_1 = \min_{P \in S} y(P),$$

$$x_2 = \max_{P \in S} x(P), \text{ and}$$

$$y_2 = \max_{P \in S} y(P),$$

where $x(P)$ is the x-coordinate of point P and $y(P)$ is the y-coordinate of point P.

Lemma 3.2. *In the absolute zero process, the bounding box of the cells with error cannot grow. Specifically, if a cell is outside of the bounding box at some point, it cannot have error at any later time.*

Proof. Assume for the sake of contradiction that the bounding box does grow. This means that after T steps the bounding box has opposite vertices (x_1, y_1) and (x_2, y_2) with $x_1 \leq y_1$

and $x_2 \leq y_2$. For the $T+1$th step to increase the bounding box, it must add a cell with error outside of the bounding box.

Without loss of generality, let this be the cell have center (x, y) with $x > x_2$. However, this means the cells at $(x, y-1)$, $(x, y+1)$, and $(x+1, y)$ are all outside of the original bounding box, so the edges separating these cells with (x, y) cannot be boundary lines. This means the cell (x, y) has at most one boundary line, so an error cannot be introduced here in the absolute zero process. □

We also quantify another important monovariant in this system.

Lemma 3.3. *The total number of boundary lines, which is also the total perimeter of the loops, never increases under the absolute zero process.*

Proof. It suffices to show that each of the three steps in the absolute zero process do not increase the perimeter. When a cell is flipped, its four sides (and only these four edges) are flipped. When a cell has four, three, or two boundary lines initially, it will have zero, one, or two boundary lines after the step, respectively. This means the number of boundary lines does not increase. □

We introduce another property that loops satisfy in the absolute zero process.

Definition 3.2. We define a region to be *orthogonally convex* if, given two points within the region with coordinates (x, y_1) and (x, y_2) for $y_2 > y_1$, the points (x, i) for $y_1 \leq i \leq y_2$ are within the region as well. Similarly, for any two points within the region with coordinates (x_1, y) and (x_2, y) for $x_2 > x_1$, the points (j, y) for $x_1 \leq j \leq x_2$ are within the region as well.

Lemma 3.4. *If the region within the loop is orthogonally convex after step T, it must still be orthogonally convex after step $T+1$.*

Proof. For the sake of contradiction, assume the loop is not orthogonally convex after step $T+1$; in particular, the red line in the figure below enters the loop in region B, then leaves in region C, then enters in region D. This red line is horizontal without loss of generality; the case for a vertical red line is symmetric.

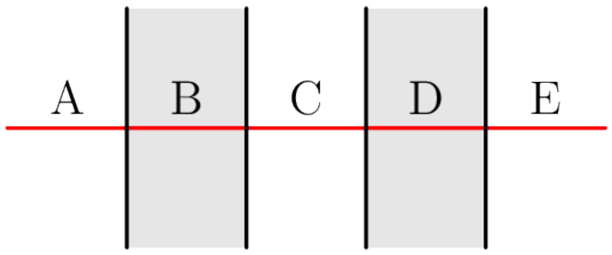

Figure 4: An example of a region that is not orthogonally convex, as the red line enters the region at B, leaves at C, then enteres again at D.

We consider each possibility for how this could have formed, even though the loop was orthogonally convex the previous step. The flip in step $T + 1$ must have been a cell within region B, C, or D; otherwise, the loop was not orthogonally convex the previous step.

Case 1: A cell within region B was flipped. Region D is equivalent.

Let the flipped cell be X. If there is another cell with error to the left or right of X, this contradicts the assumption that the loop was orthogonally convex the previous step. As a result, neither of the edges to the left or right of X were boundary lines in the previous step. This means X cannot have two or more adjacent boundary lines, so it could not have been flipped by the rules of the absolute zero process.

Case 2: A cell within region C was flipped.

This case is similar to Case 1. There cannot be two cells within C that are beside each other horizontally. As a result, if we let the flipped cell be X, both the cells to the left and right of X must have had error. This is again a contradiction, as X has no way of being flipped.

As a result, we have a contradiction, so the loop must stay orthogonally convex. $\square$

Corollary 1. *If the initial loop is square, then the loop will stay orthogonally convex for the* *remainder of the process.*

We have shown that the loop is bounded and cannot grow infinitely in the absolute zero process. Now, we show that the loop tends to shrink by considering the expected number of error cells.

Lemma 3.5. *Assume that the boundary lines form a single loop. If there are no cells with three or more boundary lines, there must be exactly four more error cells that have two boundary lines than non-error cells that have two boundary lines.*

Proof. The loop has only edges parallel to the x and y axes, which means every angle formed must be either $90°$ or $270°$. Let there be a $90°$ angles and b $270°$ angles.

There must be $a+b$ sides, which means the total sum of angles of the loop must be $180°(a+b-2) = 180°(a+b) - 360°$. Equating this to the sum of angles computed directly, we get

$$180°(a+b) - 360° = 90° \cdot a + 270° \cdot b.$$

Simplifying, we get that

$$90° \cdot a - 360° = 90° \cdot b,$$

so

$$a - b = 4,$$

as desired. □

Finally, we can use an expected value argument to prove the upper bound for the time complexity of the absolute zero process.

Theorem 3.6. *The time complexity for the absolute zero process to correct all errors in an $n \times n$ grid is at most $O(n^3)$.*

Proof. We consider the expected value of the number of cells with error. We claim that the expected change in area is $O(n^{-1})$ at worst.

First, any cell with three or more boundary lines is removed automatically by the absolute zero process, so the expected change in area in this case is -1.

Otherwise, a cell with two boundary lines is selected at random. Every such cell borders two boundary lines. By Lemma 3.3, there are at most $4n$ boundary lines, so there are at most $8n$ cells that are adjacent to a boundary line, which means there are $O(n)$ flippable cells. If there are p flippable cells with error and q flippable cells without error, the expected change in area is

$$\mathbb{E}(\text{Area}) = \frac{q-p}{p+q}.$$

By Lemma 3.5, $q - p = -4$, and since there are $O(n)$ flippable cells, $p + q$ is in $O(n)$. Thus, the expected *decrease* in area is

$$-\mathbb{E}(\text{Area}) = \frac{4}{O(n)} = \Omega(n^{-1}).$$

Because the initial area is n^2, the expected number of steps for the loop to shrink to zero is $\frac{n^2}{\Omega(n^{-1})} = O(n^3)$. $\square$

The simulation suggests that the process has a time complexity of $O(n^3)$, which we have matched in the upper bound. As for the lower bound, a weak lower bound of the time complexity is n^2, as there are n^2 error cells to correct. As the process imitates a random walk, however, the time complexity is likely well above this lower bound. We show both the upper and lower bounds of $O(n^3)$ for an altered process.

An Altered Process

We analyze an altered process which models the original heat bath algorithm more closely.

Definition 3.3. In this altered process, we again begin with a finite number of cells with error on an infinite grid. Each step, the following occurs:

1. A random cell with at least one boundary line is selected.

2. If this cell has three or four boundary lines, flip it.

3. If this cell has two boundary lines, flip it with a probability of $\frac{1}{2}$. These two boundary lines do not need to be adjacent.

4. If this cell has one boundary line, do not flip it.

As in the absolute zero process, this process is repeated until no cells have error. Significantly, a step does not always result in a cell being flipped.

Several of the lemmas that we proved for the standard absolute zero process still apply here. In particular, Lemma 3.1, Lemma 3.2, and Lemma 3.3 remain true because we still do not allow cells with one boundary line to be flipped.

Lemma 3.4, however, does not remain true; any step that flips a cell with two opposite boundary lines breaks the orthogonally convex condition. Still, a similar result stays true: each individual loop remains orthogonally convex. A complete proof of this claim is located in Appendix A.

We use this result to prove that both the upper and lower bound of the time complexity is $O(n^3)$.

Theorem 3.7. *The time complexity for the altered absolute zero process to correct an $n \times n$ array of error cells is $O(n^3)$.*

Proof. Similar to Lemma 3.5, we claim that the probability of flipping a cell with error is $\frac{1}{2} + O\left(\frac{1}{n}\right)$. There are $O(n)$ flippable cells, as there are $O(n)$ boundary lines in total. As a result, we wish to show that it is more likely to flip a cell with error than without.

The probability of flipping a cell with three or four boundary lines is 1, while the probability of flipping a cell with two boundary lines is $\frac{1}{2}$. This means cells with three or four boundary lines have double the weight of cells with two boundary lines. In addition, every cell without error has zero, one, or two boundary lines; otherwise, the loops would not be orthogonally convex.

First, we consider loops of size one — i.e. one cell X of error with four boundary lines. By the nature of the loops discussed previously, X is part of a chain of rectangles; thus, at most two of the cells adjacent X are flippable. Because X has twice the probability of being flipped, the probability of flipping a cell with error is at least as much as cells without error for this case.

Now, we consider loops with more than one cell within it. Here, we can use similar reasoning to Lemma 3.5 and Theorem 3.6. Because cells with three boundary lines consist of two corners and have twice the weight and there are four more corners with angle 90° than with angle 270°, for loop on the grid, the probability of flipping a cell with error is larger than the probability of flipping a cell without error.

The final case we have to consider is the dark gray cells that link adjacent loops, as described in Appendix A. However, the number of such cells is less than the number of total loops, so it is overall more likely to select a cell with error. By the same reasoning as in Theorem 3.6, there are $O(n)$ total possible steps, so the expected change in area is $\Omega(n^{-1})$. Thus, the expected number of steps for the altered absolute zero process to terminate is $\frac{n^2}{\Omega(n^{-1})} = O(n^3)$. □

4 The Probabilistic Model

Now, we add a few more complications into our simulations to approach the Heat Bath Algorithm. For some probability $p < \frac{1}{2}$, we define the probabilistic model as follows:

Definition 4.1. In the *probabilistic model*, we again begin with a finite number of cells with error on an infinite grid. Each step, the following occurs:

1. A random cell with at least one boundary line is selected.

2. If this cell has four boundary lines, flip it.

3. If this cell has three boundary lines, flip it with a probability of $1 - p$.

4. If this cell has two boundary lines, flip it with a probability of $\frac{1}{2}$.

5. If this cell has one boundary line, flip it with a probability of p.

Unlike in the altered absolute zero process, this only counts as a step if a cell is successfully flipped. This process is then repeated until no cells have error.

Like in the absolute zero process, we run simulations of this probabilistic model to figure out the average number of steps that the process takes to correct all of the error.

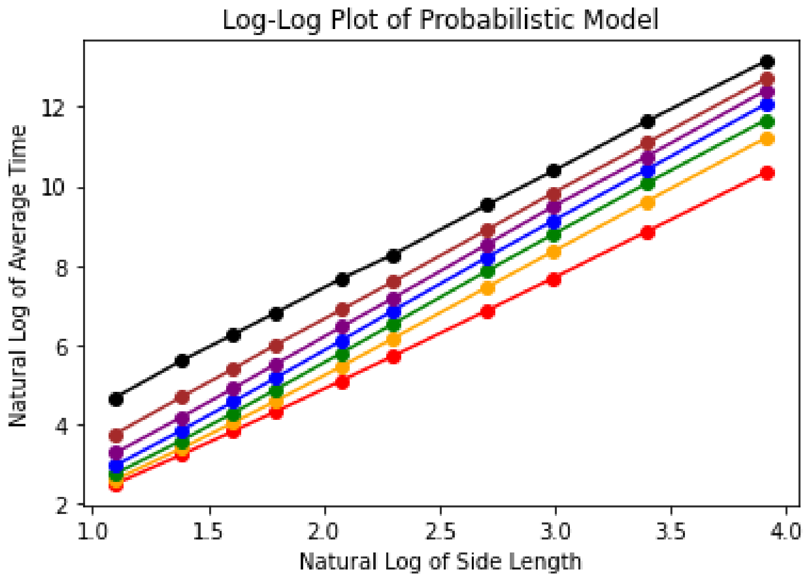

Figure 5: The expected number of steps for the probabilistic model to correct all of the error in a square loop. The bottom line marks a probability of 0; from bottom to top, the probability increases by $\frac{1}{40}$ up to $p = \frac{6}{40} = \frac{3}{20}$ for the topmost line.

In Figure 5, we have seven different data sets which correspond to different probabilities p. Several features about the mechanics of this process are important to understanding the number of steps that error correcting takes.

The plot ends at $p = 0.15$, although larger probabilities, such as $p = \frac{7}{40}$, were not plotted. For these larger probabilities, the simulation for the probabilistic model did not terminate in sufficient time. This implies that the loop continued to grow in size, unlike for lower values of p, for which all of the simulations ended quickly.

This behavior implies that there is a critical probability at which the loop grows without bound often, giving an infinite average number of steps. For the processes we have analyzed so far, the random process could theoretically be unending, but the probability of this occurring is exponentially decreasing, so the average number of steps is not infinite. However, in this case, there is a critical probability at which the average number of steps is infinite. This is behavior similar to problems such as bond percolation. According to the simulations, the critical probability lies between 0.173 and 0.175.

Simultaneous Algorithms

Finally, we end this section on a remark about simultaneous algorithms. Because each cell is locally modified as a result of its neighbors, it is possible to run this process for each cell simultaneously.

However, this algorithm does not have the nice properties that the other algorithms have. For example, consider the most basic case where, at every step, each cell is flipped with a probability of 0, 0, $\frac{1}{2}$, 1, or 1, depending on whether it has zero, one, two, three, or four boundary lines, respectively. This ends up displaying oddities such as the formation of teeth-like structures shown in Figure 6

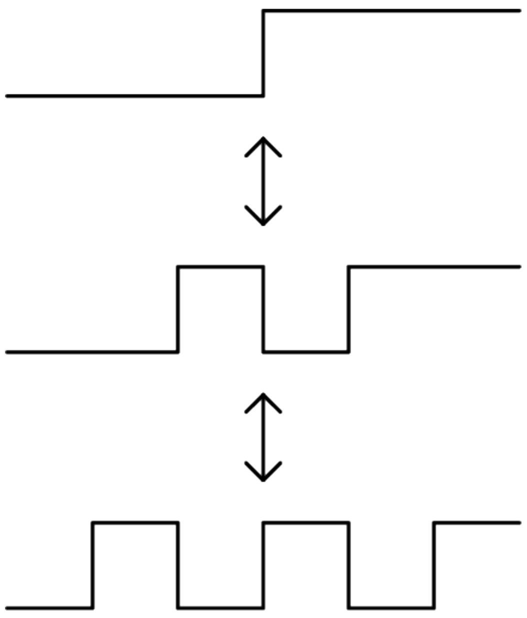

Figure 6: Several boundary lines have a significant probability of forming teeth when cells are flipped simultaneously.

The first boundary line shown in Figure 6, which is a very standard part of a loop, has a $\frac{1}{4}$ chance of changing into the second boundary line, which increases in perimeter (despite not flipping any squares with one boundary line). From there, this has a $\frac{1}{4}$ chance of changing back into the first figure, a $\frac{1}{2}$ chance of moving laterally (and staying the same length), and a $\frac{1}{4}$ chance of changing into the third boundary line, which increases the perimeter further. This means, for a sufficiently long starting boundary line, simultaneous algorithms can be modeled as two fair random walks, with everything in between being teeth as seen in Figure 6. As we had previously used perimeter as a monovariant for the size of the loop, this behavior is undesirable. We instead analyze algorithms where cells are flipped sequentially, which results in more noticeable patterns.

5 Heat Bath Algorithm

Finally, we discuss how the heat bath algorithm is able to correct error. The heat bath algorithm introduces the ability to flip cells that have zero boundary lines, as described by Dennis *et. al.* [4]. To prevent adding error at arbitrary locations in an infinite grid, we model the heat bath algorithm on a finite toric grid, which means that the left edge and right edge are the same, and the top edge and bottom edge are the same.

As the heat bath algorithm models the introduction as well as the removal of error, we begin with no error and study the equilibrium number of errors after a long time.

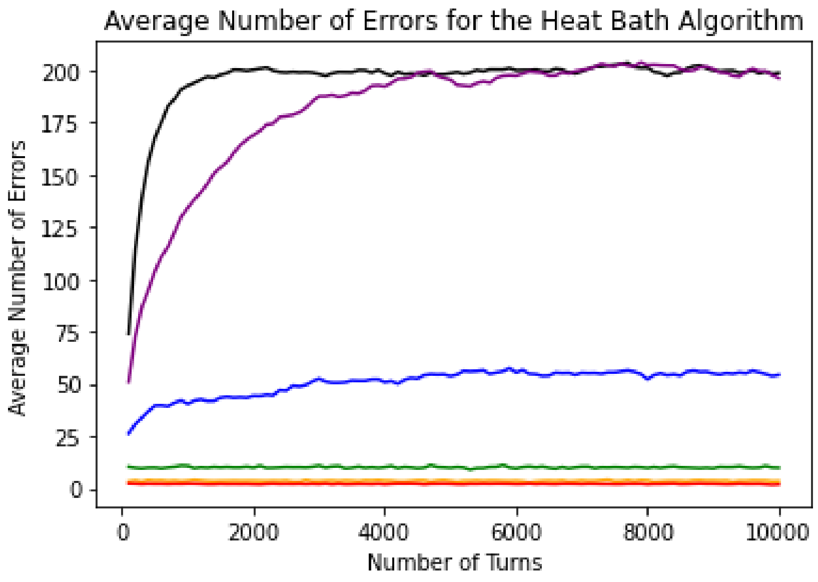

Figure 7: Each line corresponds to a different temperature. The values of $e^{-2\beta}$, where β is the inverse temperature, are 0.5, 0.2, 0.1, 0.05, 0.02, and 0.01 from top to bottom. These approximately correspond to the probability of flipping a cell with one boundary line.

The results of the simulation for this process for a 20×20 loop is shown in Figure 7, which displays several behaviors.

First, when $e^{-2\beta}$ is 0.5 or 0.2, the points quickly reach an equilibrium of approximately 200 errors. This is exactly half of the total number of cells. Our algorithm cannot distinguish cells with error from cells without error; thus, if the expected number of error cells equals the expected number of non-error cells, there is no way to differentiate the two. However, for lower values of $e^{-2\beta}$, this behavior is not present. Instead of increasing steadily to 200, the number of error cells plateaus. This implies the existence of an equilibrium at this point. This means, typically, that a low temperature results in the introduced errors being corrected quickly.

6 Future Work

Each of the algorithms we analyzed have promising future work to be made. The behaviors of these loops over time result in notable patterns, as we discovered with the simulations, but are more difficult to prove mathematically.

For the standard absolute zero process, our current lower bound for the time complexity is $O(n^2)$, which is less than suggested by the simulation. Further work could be done to prove this lower bound. It would suffice to show that the expected change in area is $\Omega(n^{-1})$; this appears to be true in general, with only a few exceptions such as the initial square loop.

A process related to the absolute zero process is the totally asymmetric simple-exclusion processes (TASEPs). In this process, a cell with two boundary lines are randomly removed from a quadrant of cells. However, unlike in the absolute zero process, cells cannot be added back. It has been proven in TASEPs that the boundary lines approximate a quarter-circle [5]. The techniques used regarding TASEPs could be used regarding the absolute zero process, which also approximates a quarter-circle on each corner.

The probabilistic model discussed in Section 4 also has interesting properties to be analyzed. Particularly, there is a critical probability at which the loop is expected to grow without bound. The cells with error tend to form small clusters; when a loop grows without bound, the number of these clusters grows, instead of the size of the clusters growing.

There is also more future work regarding the heat bath algorithm regarding the behavior of the loops for different temperatures. The average number of cells approaches an equilibrium for low temperatures, so we could analyze the behavior of the loop at the equilibrium. Specifically, the expected change in area should be 0, which is dependent not only on the number of error cells but also the way these cells are arranged on the grid.

Finally, there are many more algorithms that could be analyzed. Dennis, Kitaev, Landahl, and Preskill discuss four-dimensional codes in their paper, correcting for both X and Z errors on the qubits. Although this complicates the processes discussed in our paper, the techniques used may work for higher-dimensional cases as well.

7 Conclusion

Inspired by the heat bath algorithm, we analyzed several different stochastic processes that correct error cells on a grid. The first process we analyzed is the most orderly, the absolute zero process. Simulations of this process on a square initial loop showed an approximately cubic time complexity for the loop to vanish. Then, we made several mathematical observations about the nature of this loop and used them to prove that this process has an expected time complexity of at most $O(n^3)$. We then proved that an altered version of this process has an expected time complexity of exactly $O(n^3)$.

To approach the original heat bath algorithm, we analyzed a probabilistic model to correct loop in an error. We found that this process always converges for small probabilities, but grows without bound for large probabilities. Finally, we analyzed the heat bath algorithm on an initially empty toric loop. For small temperatures, the grid reached an equilibrium in the average number of cells with errors; however, for large temperatures, the number of cells approached $\frac{n^2}{2}$.

Encoding qubits as grids is efficient because the qubits are closer together, which allows for simple stabilizers (one for each edge). We analyzed the heat bath algorithm and simplified variants, which are processes that correct errors on these grids. This is useful when using grids to encode messages, providing insights on the time necessary to correct these errors and the temperature required to prevent errors from propagating.

8 Acknowledgments

I would like to thank my mentor, Andrey Boris Khesin, for his plentiful advice and assistance in this project. I would also like to thank my head mentor, Dr. Tanya Khovanova, for her supervision and advice in the project. I am also grateful to Prof. Ankur Moitra and Prof. David Jerison for organizing the MIT Math Department and making this project possible. I am also appreciative of Dr. John Rickert, who provided feedback on the paper and presentations. Thanks also to Dimitar Chakarov and Dev Chheda for their assistance with final revisions. I am also thankful to my sponsors for this project, who are Mr. and Mrs. Brain M. Fairchild, Mr. and Mrs. Daniel A. D'Aniello, Dr. Fei-Ran Tian and Dr. Shanghong Dong, Dr. and Mrs. Jong K. Lee, Ms. Wendy Kershner, and Matthrew Thrasher, Ph.D. This project was possible due to the Massachusetts Institute of Technology (MIT) and the Center for Excellence in Education (CEE), which organized the Research Science Institute (RSI) at which this project was conducted.

References

[1] D. Gottesman. The heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1998.

[2] A. M. Steane. Simple quantum error-correcting codes. *Physical Review A*, 54(6):4741–4751, Dec 1996.

[3] I. L. Chuang and R. Laflamme. Quantum error correction by coding. *arXiv preprint quant-ph/9511003*, 1995.

[4] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, Sep 2002.

[5] Y. Yamada and M. Katori. Velocity correlations of a discrete-time totally asymmetric simple-exclusion process in stationary state on a circle. *Physical Review E*, 84(4), Oct 2011.

A The Altered Absolute Zero Process

When moves that flip two cells with two opposite boundary lines are permitted, lemma 3.4 does not remain true.

Lemma A.1. *After T steps in the altered absolute zero process starting from a single square loop, each loop that surrounds cells with error on the grid is orthogonally convex.*

Proof. We prove that these properties hold after every step. For steps that do not flip a cell with two opposite boundary lines, the argument in Lemma 3.4 maintains that these conditions hold. Thus, we consider the first time a cell with two opposite boundary lines is flipped.

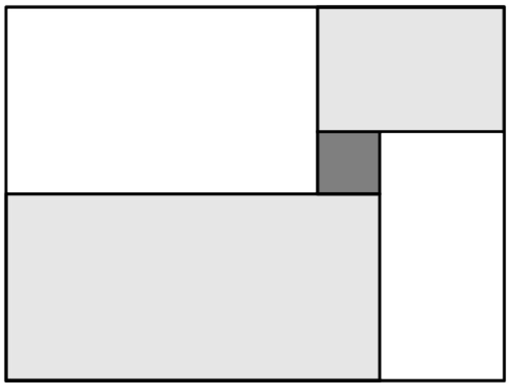

Figure 8: The dark gray cell is flipped from error to no error, forming two potential regions of error in light gray that are individually orthogonally convex

In Figure 8, call the cell with two opposite boundary lines X, which is marked in dark gray. When X can be flipped, without loss of generality, assume that the cells to the left and right of X have no error. Then, since the loop is orthogonally convex, there cannot be error cells in both the southwest and northwest quadrants. The same logic applies to the northeast and southeast quadrants. Thus, without loss of generality, the light gray rectangles mark cells which could be part of the loop; the white cells cannot have error.

If these sections recombine, it must be because X was flipped again; no other cell bounds both rectangles shaded in light gray.

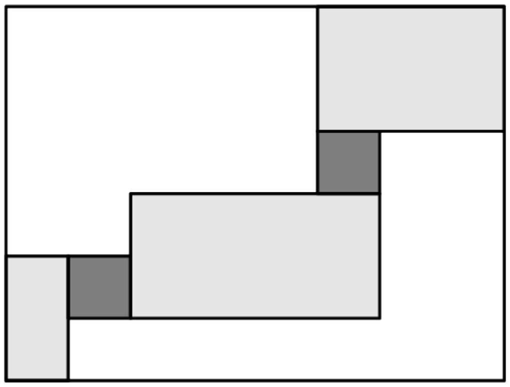

Figure 9: Another dark gray cell is flipped, creating two light gray rectangles as before. This creates a chain of three light gray rectangles, each connected to the next through one dark gray cell.

Now, consider another cell with two opposite boundary lines Y being flipped, as in Figure **??**. In this case, we see that the light gray rectangles form a chain, each going from southwest to northeast. If two consecutive rectangles are combined by flipping a dark gray cell, this decreases the number of rectangles in the chain but does not violate the conditions.

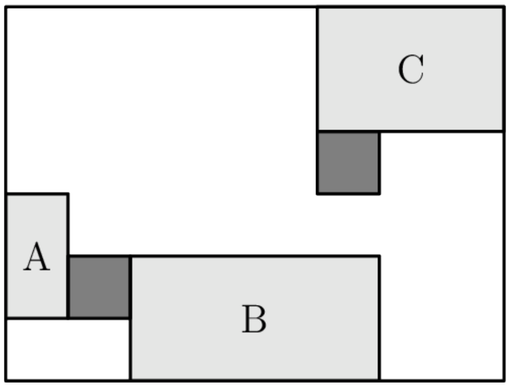

Figure 10: Rectangles A and B are now oppositely oriented, which means the rightmost dark gray cell is only connected to one light gray rectangle, C.

The other case is if the two new rectangles were oppositely oriented; i.e. from northwest to southeast, as shown in Figure 10. In this case, we notice that rectangles A and B are disconnected from rectangle C. Thus, rectangles A and B need to be merged before they can merge with rectangle C. We can use the same argument as before on rectangles B and C to prove that the conditions hold.

As the merging of two regions through a dark gray cell is always the same as the separation of these regions, each contiguous region of error is orthogonally convex. $\square$