Quantum Error Correction (QEC)

Tan Viet Nguyen, Francesco Virgulti

April 2025

1 Introduction

In quantum computing, *noise* refers to disturbances that disrupt qubit behavior. Qubits are fragile and highly sensitive to environmental interference, making quantum operations error-prone. Despite recent advances achieving entangling gate fidelities of about 99.9%, this still falls short of the ultra-low error rates (below 10^{-10}) required for fault-tolerant quantum computation.

Quantum Error Correction (QEC) offers a solution by detecting and correcting errors without collapsing the quantum state. It focuses on key error types such as *bit-flip* and *phase-flip* errors. One of the most promising QEC strategies is the *Surface Code*, a topological code introduced by Alexei Kitaev in 1997. On July 20, 2022, Google achieved a major milestone by experimentally demonstrating QEC using the Surface Code.

2 Classical Error Correction

Information transmission is vulnerable to noise, which can corrupt bits during communication. Classical error correction addresses this by adding redundancy to the message, allowing the receiver to detect and correct errors.

Repetition Code A simple error-correcting scheme is the *repetition code*, defined by a mapping $[n, k] : \{0, 1\}^k \to \mathbb{Z}_2^n$, where each bit is repeated n times. For instance:

$$0 \mapsto 000, \quad 1 \mapsto 111$$

in the $[3, 1]$ code. These outputs are called *codewords*.

Let p be the probability of a bit flip. In a $[2, 1]$ code:

- Correct transmission (11): probability $(1 - p)^2$
- Single-bit error (01 or 10): probability $2p(1 - p)$ (detectable)
- Double-bit error (00): probability p^2 (rare and often ignored)

Longer codes improve error resistance through *majority voting*, though at the cost of transmission efficiency.

3 Quantum Error Correction

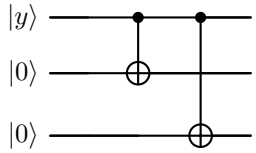
The *No-Cloning Theorem* states that an arbitrary unknown quantum state cannot be copied. This limits the use of classical repetition-based error correction in quantum systems.

However, we can protect quantum information by distributing it across multiple qubits via entanglement. For example, the *three-qubit bit-flip code* encodes:

$$|y\rangle = \alpha|0\rangle + \beta|1\rangle \quad \to \quad \alpha|000\rangle + \beta|111\rangle$$

This encoding guards against single-qubit bit-flip errors.

The encoding circuit uses CNOT gates to entangle the original qubit with two ancilla qubits:



Direct measurement isn't possible due to state collapse, so quantum error correction uses *syndrome measurements*—indirect methods that detect errors while preserving quantum coherence.

Syndrome Decoding with Parity Check In the $[3, 1]$ repetition code, the codewords are:

$$0 \mapsto 000, \quad 1 \mapsto 111 \tag{1}$$

Encoding a bit z uses the generator matrix:

$$G = [1 \quad 1 \quad 1], \quad x_z = z \cdot G \tag{2}$$

To detect errors, we use the parity-check matrix H. A valid codeword satisfies:

$$H \cdot x^T = 0 \tag{3}$$

A non-zero result, called the *syndrome*, indicates an error and helps identify its location.

Example If $z = 0$ yields $x_z = 000$ but we receive $y = 010$, then:

$$H \cdot y^T = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{4}$$

The non-zero syndrome suggests an error in the second bit in y. This method enables single-bit error detection and correction using linear algebra, forming the basis of more advanced codes.

Hamming Code In an $[n, k]$ code, the ability to detect or correct errors depends on the *minimum Hamming distance* d, defined as the smallest number of bit differences between any two codewords in the code $C \subseteq \mathbb{Z}_2^n$.
We define:

- *Weight:* $w(x)$ is the number of ones in $x \in \mathbb{Z}_2^n$.
- *Hamming distance:* $d_H(x_1, x_2) = w(x_1 \oplus x_2)$, the number of differing bits between x_1 and x_2.

A code with minimum distance d can:

- Detect up to $d - 1$ errors
- Correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors

For example, the Hamming $[7, 4]$ code has $d = 3$, allowing it to correct 1-bit errors and detect 2-bit errors. More generally, an $[n, k, d]$ code summarizes its length, data capacity, and error tolerance.
An error corrupts a codeword x into a received word y via:

$$y = x \oplus e \tag{5}$$

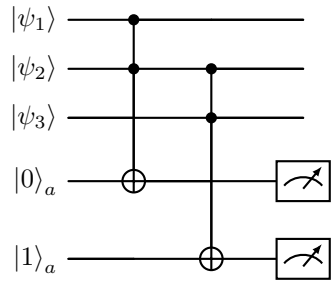
where $e \in \mathbb{Z}_2^n$ is the error vector. If $w(e) < \lfloor \frac{d-1}{2} \rfloor$, the decoder can reliably recover the original x. This idea is central to both classical and quantum error correction.

From Parity Check to Quantum Codes Extending classical parity checks to quantum error correction introduces key challenges:

1. **No direct measurement:** Measuring a quantum codeword collapses its state. Instead, we use ancilla qubits and *stabilizer circuits* to extract syndromes without disturbing encoded data.
2. **More error types:** Unlike classical bits, qubits can suffer from bit-flip (Pauli X), phase-flip (Pauli Z), or both (Pauli Y). Quantum codes must handle all cases.
3. **Errors in superposition:** Errors can affect qubits in superposition, introducing entangled or partial errors, complicating detection and correction.

These difficulties demand techniques like stabilizer codes and entanglement-assisted syndrome extraction to preserve coherence. [1]

Parity Check Firstly, we increase the size of the circuit, adding 2 ancilla qubits. Moreover, we can replace the classical parity check with measurement of the observable $Z1, Z2$ and $Z2, Z3$.



It is important to stress that measuring $Z1, Z2$ is not equivalent to measuring $Z1$ independently from $Z2$, as the state will collapse into $|000\rangle$ or $|111\rangle$ It is clear that without noise we have : $Z_1 \otimes Z_2 |\Psi\rangle = Z_2 \otimes Z_3 |\Psi\rangle = |\Psi\rangle$. We presume that the error could be applied only in one of the data qubits, and with this precondition, the measurement result can uniquely identify if and where an error has occur.

Phase Flips Unlike classical bit, qubits can even suffer from phase-flip, that can't be detected by Parity check. The main idea is to adapt the Parity check also for this kind of error.
If we apply a Z types error we have : $\varepsilon_e(\rho) = (1 - p_o)\rho + \sum_j p_j Z_j \rho Z_j$ that is the probability that no error occurs, plus the probability that a phase flip error occurs on one data qubits. Each term in the sum corresponds to a possible Z_j error.
Since $Z = H \otimes H$ we have : $H^{\otimes 3} \varepsilon_z (H^{\otimes 3} \rho H^{\otimes 3}) H^{\otimes 3} = (1 - p_o)\rho + \sum_j p_j X_j \rho X_j$
Which is a bit flip error channel that we know our code is able to handle.

3.1 Qiskit implementation

3.1.1 Bit Flip Error Correction

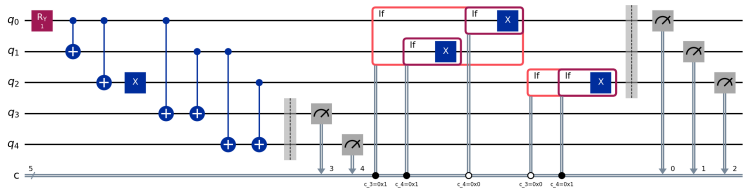


Figure 1: Bit-Flip Error Correction Circuit Implemented in Qiskit

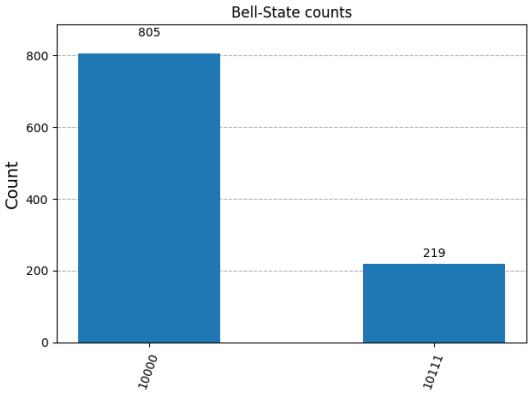


Figure 2: Bit-Flip Error Circuit Simulation

We implemented a bit-flip error correction scheme using Qiskit's simulator, as shown in Fig. 1. The circuit demonstrates the detection and correction of bit-flip errors in a three-qubit quantum system.

The process began by encoding an initial qubit in a superposition state into a three-qubit entangled state. A bit-flip error was then intentionally introduced on one of the qubits (q2) to test the correction mechanism. Error detection was performed using syndrome measurements via two ancilla qubits (q3 and q4), which allowed us to identify the location of the error.

Based on the syndrome outcome, in Fig. 2, a corrective operation was applied using conditional gates. Final measurements confirmed the effectiveness of the correction procedure—the output state was consistently either $|000\rangle$ or $|111\rangle$, demonstrating that the logical qubit remained in a valid codeword state.

3.1.2 Phase Flip Error

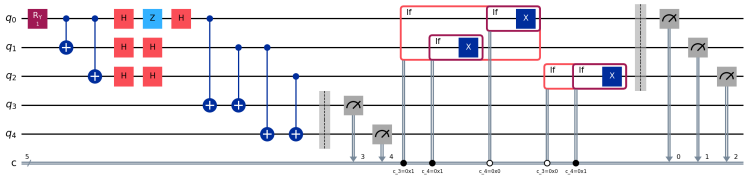


Figure 3: Phase Flip Error Correction Circuit Implemented in Qiskit

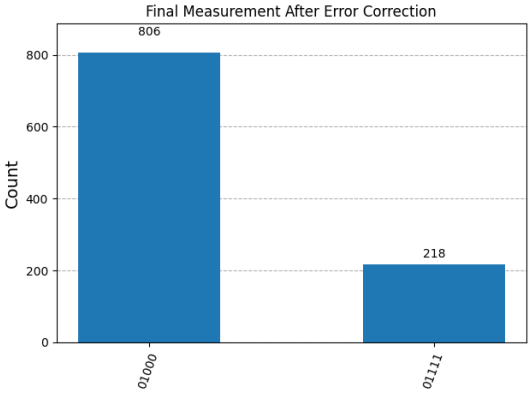


Figure 4: Phase Flip Error Circuit Simulation

We implemented a quantum error correction scheme to detect and correct phase-flip (Z-type) errors Fig. 3.

Since phase errors shift the relative phase between $|0\rangle$ and $|1\rangle$, we applied Hadamard gates to convert them into bit-flip errors using the identity:

$$H \cdot Z \cdot H = X \tag{6}$$

This allowed us to reuse the same CNOT-based syndrome extraction circuit as in the bit-flip case. A Z error was applied to qubit q0, and ancilla qubits (q3, q4) were used to detect it. Based on the syndrome, we applied a corrective X gate.

Final measurements Fig. 4 yielded $|000\rangle$ or $|111\rangle$, confirming that the logical qubit was successfully corrected and remained in a valid codeword state.

4 Some concepts in topology

4.1 Euler Characteristic and Genus

Following [2], in order to have a simple explanation, we will always consider the topological properties of a torus. For a torus, Euler characteristic is an invariant quantity meaning however you stretch or squeeze the space this quantity is still the same. It is denoted as $\mathcal{X}$ and is a formula of edges E, faces F, vertices V of a lattice equivalent to a torus:

$$\mathcal{X} = V - E + F \tag{7}$$

As topology is the study of holes, any objects that have the same holes are equivalent. To be more percise, we have g called the genus that counts number of handles of the object. For example, with a torus defined for toric code, we have two holes: the circular hole (the visible one) in the middle which is the handle, and the tubular one inside the

belt of the torus (the hidden one), so $g = 1$ and we have:

$$\mathcal{X} = V - E + F = 2(1 - g) = 2(1 - 1) = 0 \tag{8}$$

Therefore, the number of logical qubits encoded in toric code is said to be $2 - \mathcal{X} = 2$ qubits (we can define 2 pairs logical operators $\bar{X}_i\bar{Z}_i$, that wrap around 2 holes of the torus). However, because the surface code is a planar surface, the number of handles g is 0 and $\mathcal{X} = 2 \mapsto 2 - \mathcal{X} = 0$, but the logical qubits encoded is 1 because it has closed boundaries, so we can define a pair logical operators $\bar{X}_1\bar{Z}_1$, that goes from one side to the other of the surface.

4.2 Homology in $\mathbf{Z}_2$

Now we want to classify every paths (faces, loops, strings and vertices) on the torus into different classes called homology. If the topological invariance quantities of the paths are the same, then they have equivalent homology. We also have the following definition of cells: 0-cells are vertices, 1-cells are edges, 2-cells are surfaces.
The values of each element e_i in n cells are in $\mathbf{Z}_2$. For example, we have a chain of edges E' (1-cells) on the lattice that can be represented by binary vectors $\{0, 0, 0, 1, 1, 1, 0, 0\}$ or all the $e_i s$ form a basis for 1-cells:

$$c = \sum_i c_i\, e_i, \qquad\qquad c_i = \begin{cases} 0 & e_i \in E' \\ 1 & e_i \in E' \end{cases} \tag{9}$$

The example above is a chain of edges denoted C_1, we also have chains of surfaces C_2, chains of vertices C_0. We define the boundary operators that bring the n-chains to (n-1)-chains:

$$\partial_n : C_n \mapsto C_{n-1} \tag{10}$$

Now we define the kernel of ∂_1 and the image of ∂_2 as $Z_1, B_1 \subset C_1$, respectively. Z_1 is the set of 1-chains z that have no boundary: $\partial_1 z = 0$, the kernel of ∂_1. They are collections of closed curves called cycles. In terms of toric codes for Z-error, these cycles are logical operators $\bar{Z}_1, \bar{Z}_2, \bar{Z}_1\bar{Z}_2$. B_1 is the image of ∂_2, the set of 1-chains b that are a boundary of a 2-chain, $b = \partial_2 c$ for some $c \in C_2$. Because of:

$$\partial^2 := \partial_1 \circ \partial_2 = 0. \tag{11}$$

Therefore, we have that all boundaries are also cycles, namely $B_1 \subset Z_1$. In terms of toric codes, because the boundaries of the surfaces on the lattice are stabilizer generators, it does not affect the logical qubits (therefore, its logical operators). We have the first homology group:

$$H_1 := Z_1/B_1. \tag{12}$$

From this group, we have the homological equivalence between cycles up to a boundary. For example, 2 cycles c, d are equal if there is a boundary e that makes $c = d + e = \bar{z}$ with $\bar{z}$ is the representative element of c, d in H_1. For chains that are not cycles, we can define its group as C_1/B_1 with $\bar{c} := \{c + b | b \in B_1\}$ is one representative element of the group. For torus, the lattice equivalence can be explained by the isomorphism of group H_1. The genus $g = 1$, therefore, we have:$H_1 \simeq Z_2^{2g} = Z_2^2 = Z_2 \times Z_2$.

4.3 Cohomology

Normally, we use H_1 for the lattice of Z-error and cohomology H_1^* for X-error (a similar anology is bras and kets quantum states). We have a lattice and dual lattice with following components: edges and dual edges are 1-cells, vertices and dual plaquettes are 0-cells, plaquettes and dual vertices are 2-cells.
With a,b are a pair of string and dual string on the lattice and dual lattice, we have the commutation rule encoded in the lattie as:

$$Z(a)X(b) = 1^{\langle b,a\rangle} X(b)Z(a) \tag{13}$$

For 2 groups of operator $X_{vertice} = X_1 \otimes X_2 \otimes X_3 \otimes X_4$, and $Z_{platquette} = Z_1 \otimes Z_2 \otimes Z_3 \otimes Z_4$, they always commute because they can be overlaped by 2 qubits or 0 qubits (see figure 5)[3]:

$$[Z_{platquette}, X_{vertice}] = 0 \tag{14}$$

However, the logical operators, for example $\bar{X}_i, \bar{Z}_i$, are not defined by group of vertices and platquettes but a string products of the edges from one side to the other side of lattice. Therefore, if they cross each other an odd number of times, they will anti-commute:

$$\{\bar{Z}_i, \bar{X}_i\} = 0 \tag{15}$$

By definition, the logical $\bar{X}_i$ is any column from top to bottom, and the logical $\bar{Z}_i$ is any row from left to right.

5 Hamiltonian of toric and surface code

The stabilizer states are the groundstates of this Hamiltonian[2]:

$$H = -\prod_v X_v - \prod_f Z_f \tag{16}$$

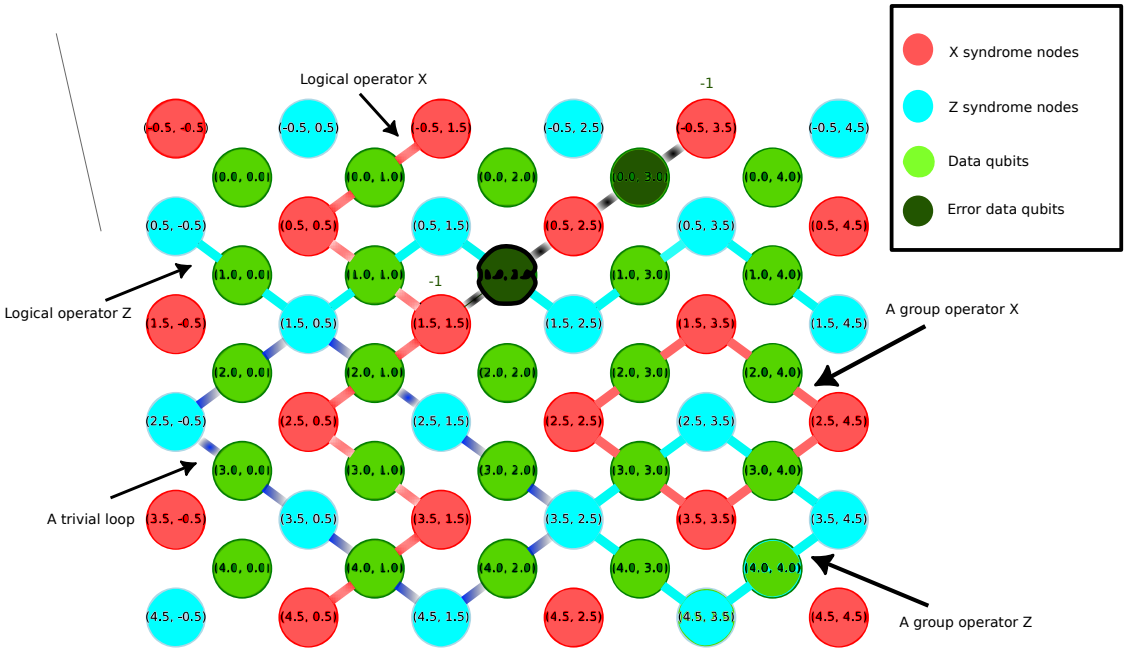


Figure 5: The lattice layout of the code including every node types. Examples of trivial loops, error string, logical operators, group operators are illustrated on the graph.

With X_v, Z_f are group operators defined like in 14. To find the groundstate we need to find the lowest energy of 16. Moreover, the 2 group operators X_v, Z_f commute we have the groundstates are the eigenstates of those 2 operators:

$$X_v|G\rangle = +1|G\rangle, \quad Z_f|G\rangle = +1|G\rangle \tag{17}$$

The wavefunction $|G\rangle$ can be written as:

$$|G\rangle := \prod_v \frac{1+X_v}{2} \prod_f \frac{1+Z_f}{2} \; |00..0\rangle = \prod_v \frac{1+X_v}{2} \; |00..0\rangle \tag{18}$$

For qubits $= 4$, this is the generalized GHZ state:

$$|G\rangle = \frac{1+X_1 \otimes X_2 \otimes X_3 \otimes X_4}{2} \; |0000\rangle = \frac{|0000\rangle + |1111\rangle}{2} \tag{19}$$

If there is an error X_i applied to $|G\rangle$, it will bring the state out of code space:

$$Z_f|\phi\rangle = Z_f X_i|G\rangle = -X_i Z_f|G\rangle = -X_i|G\rangle = -|\phi\rangle \tag{20}$$

So,

$$\langle\phi|Z_f|\phi\rangle = -1 \tag{21}$$

When Z_f measurement applied to the error qubit, it can detect the error syndrome in the eigenvalue of the observable Z_f. Interestingly, when we apply twice the error operator into the same qubits, we regain the original state (anyon particle connection [4]). From this, if we have an error chain, for the error in the middle of the chain, it will be self-annihilated leaving only 2 error on both end of the string, in a way, we can move the error on the lattice. This can also explain why the trivial loops on the torus do not affect the logical qubits because the two ends of the loop are in the same position,so it is self-annihilated.

6 Decoder

Code: We simulate the decoding for the surface code. On the bottom, left and right of the lattice, we attach virtual nodes indicating the boundary of the surface. These nodes can connect to normal syndrome nodes with non-zero weights but they connect with each other with 0 weight.

Noise Model: We used a Pauli error noise model. With probability p, the data qubits have a bit flip. Each bit flip will affect 2 syndrome nodes on the Z syndrome graph; we record themn to create the error Z graph; the phase flip can be treated seperately with similar procedure. Given the error set E such that $ES \notin \mathcal{S}$, where $S, \mathcal{S}$ are stabilizer and stabilizer group, respectively, the probability of $P(E)$ is described as the binomial distribution on the

data qubits:

$$p(E) = \prod_i (1-p)^{(1-\mathbf{e}[i])} p^{\mathbf{e}[i]} = (1-p)^n \prod_i \left(\frac{p}{1-p}\right)^{|\mathbf{e}|}, \tag{22}$$

Where $\mathbf{e}[i] = 1$ if there is an error on qubit i, otherwise $\mathbf{e}[i] = 0$, $|\mathbf{e}|$ is the set of all error qubits.

Perfect weight matching decoder (PWM)[5]: We want to maximize the probability likelihood of equation 22, so that we can find the most probable error string:

$$\max p(E) = \prod_i \max_{e_i} \{(1-p)^{(1-\mathbf{e}[i])} p^{\mathbf{e}[i]}\} = \max_{\mathbf{e}} (1-p)^n \prod_{e \in E} \left(\frac{p}{1-p}\right)^{|\mathbf{e}|}, \tag{23}$$

Equation 23 can be written as:

$$p(E) \propto \max_{\mathbf{e}} \prod_e \left(\frac{p}{1-p}\right)^{|\mathbf{e}|} \propto \max_{\mathbf{e}} \sum_{e \in E} |e| \ln(\frac{p}{1-p}) \propto -\sum_{e \in E} |e| \tag{24}$$

The minus sign is there because $\ln(\frac{p}{1-p}) \ll 1$.

We also take into account the path degeneracy where there are multiple ways that error chains can reach from one node to the other node, thus increasing the error probability. We denoted $\Omega(e)$ to count this degeneracy:

$$p(E) \propto \max_{\mathbf{e}} \prod_e \Omega(e) \left(\frac{p}{1-p}\right)^{|\mathbf{e}|} \propto \max_{\mathbf{e}} -\sum_{e \in E} |e| - \ln(\frac{\Omega(e)}{\ln(\frac{1-p}{p})}) \tag{25}$$

We use the library qtcode [6] as the base code for our implementation. To initialize the algorithm, we randomly flip the data qubits and collect the affected syndrome qubits that have odd parity (they are called excited syndromes) to create the error Z graph syndrome. In this graph, we assign $P(E)$ as the edge weights between excited syndrome nodes and connect all the nodes together. Then, we calculate the shortest distances of every excited syndromes in the error Z graph. After that, we use the blossom algorithm to find the perfect matching pairs of nodes that maximize equation 25. After the correction, we check the logical operator by creating a second Z syndromes graph $\mathcal{G}$ that has the edges as the error data qubits and correction qubits. We find the connected components of $\mathcal{G}$ and count the number of paths that go from the left most to the right most of the lattice. If the count is odd, there is a logical error, otherwise there is no error.

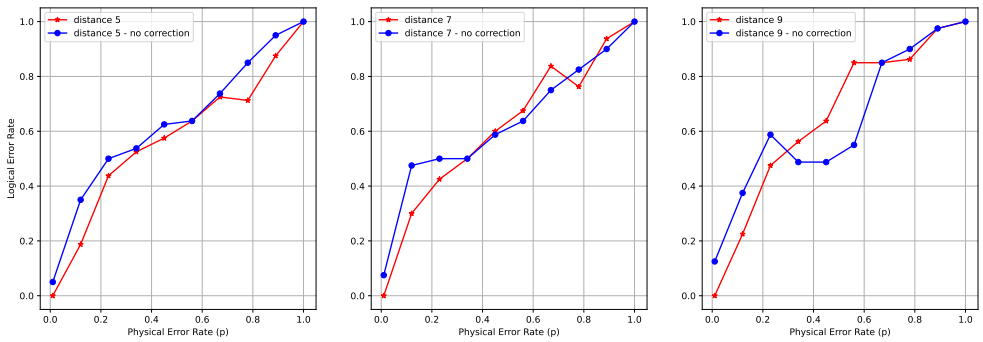


Figure 6: Logical Error Rate vs Physical Error Rate for Different Distances d.

Benchmark: Figure 6 shows the results of the PWM algorithm on distances $\in \{5, 7, 9\}$. For each distance, we gradually increase the physical error rate, and run 80 times for each case. In the figure, the red lines are results of one round error correction, while the blue lines have no correction at all. It can be seen that the decoder manage to bring down the error when the physical error rates $p \in \{0, 0.2\}$ across the distances.

References

[1] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition edition, 2010.

[2] H. Bombin. An introduction to topological quantum codes, 2013.

[3] 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), September 2012.

[4] A.Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, January 2003.

[5] Ben Criger and Imran Ashraf. Multi-path summation for decoding 2d topological codes. *Quantum*, 2:102, October 2018.

[6] et al. Phionx. https://github.com/yaleqc/qtcodes, 2020.