Google Willow Chip Simulation

Zhu Xizhe, Qin Zhijia, Zhang Shihan, and Wu Yuran
CKC College Zhejiang University, Hangzhou 310058

(Dated: June 1, 2025)

This paper investigates the quantum error correction (QEC) performance of the surface code via numerical simulation, comparing findings with Google's Willow chip experiments. After introducing surface code theory, Minimum Weight Perfect Matching (MWPM) decoding, and a statistical model for logical error rates (LER), we developed two noise models: one for evaluating below-threshold surface code memory (d = 3 to d = 21, calculating the error suppression factor Λ), and another for real-time decoding scenarios (d = 3, 5, 7). Simulations show an exponential decay in LER with increasing code distance and a slight rise in detection probability, trends consistent with Google's observations. However, specific Λ values and LERs differed from Google's experimental data, primarily attributed to simplified noise models and decoder algorithm differences. This study offers a simulation perspective on surface code QEC capabilities, highlighting the importance of accurate noise modeling and advanced decoding strategies for high-performance QEC.

Quantum computing holds immense promise for solving complex problems, yet qubit fragility and high sensitivity to environmental noise remain significant hurdles for large-scale, reliable systems. Quantum Error Correction (QEC) codes, particularly the surface code, offer a crucial solution by redundantly encoding information to build fault-tolerant logical qubits. The surface code is a leading candidate due to its high error threshold, local interaction requirements, and compatibility with 2D qubit arrays. Notably, the Google Quantum AI team recently demonstrated a below-threshold surface code memory on their "Willow" processor[1], integrating it with real-time decoding. Their results show a logical error rate decreasing exponentially with increasing code distance when the physical error rate is below threshold.

I. INTRODUCTION

This paper aims to systematically study and analyze the quantum error correction performance of the surface code through numerical simulation methods. We will first review the fundamental theory and decoding methods of the surface code, and derive the approximate relation between the logical error rate and the number of physical qubits per logical qubit

$$P_L \propto (\frac{p}{p_{\text{thr}}})^{\lceil d/2 \rceil}. \tag{1}$$

Building on this theoretical foundation, we will construct two different noise models to simulate two key scenarios: "a below-threshold surface code memory" and "real-time decoding." By running extensive computer simulations, we have obtained key performance indicators such as logical error rates and detection probabilities at various code distances. Finally, we will compare and analyze our simulation results with publicly available experimental data from Google Willow chip.

II. BACKGROUND

A. Surface code theory

1. Definition of the surface code

We will use the following method to represent the surface code.[2] The surface code can be defined on a square grid, where data qubits sit on the edges. Then there are two types of stabilizers: the vertex stabilizers(Z-type stabilizer), defined on every vertex of the lattice as a cross of four Z Pauli operators; the plaquette stabilizers(X-type stabilizer), defined on every face as a square of four X Pauli operators. Fig. 1 shows the two types of stabilizers[1].

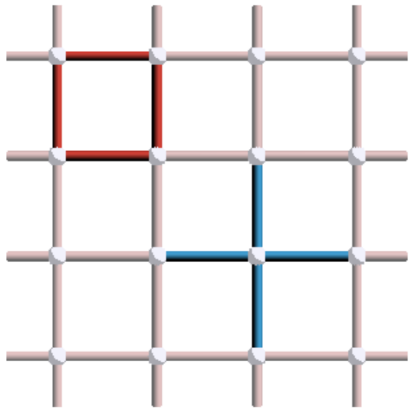


FIG. 1: Red signifies an X-type stabilizer acting on the four edges of its face. Blue signifies a Z-type stabilizer acting on the four edges connected to its vertex.

Then, we analyze the specific quantum circuit con-

[1] The diagram is drawn by the tools provided in a blog website https://arthurpesah.me/blog/2023-05-13-surface-code.

structions for both stabilizer types. In Fig. 2(a), we see the implementation of the stabilizers.[3] An ancilla qubit is used for measurement. This projective measurement forces the four adjacent data qubits into an eigenstate of the stabilizer.

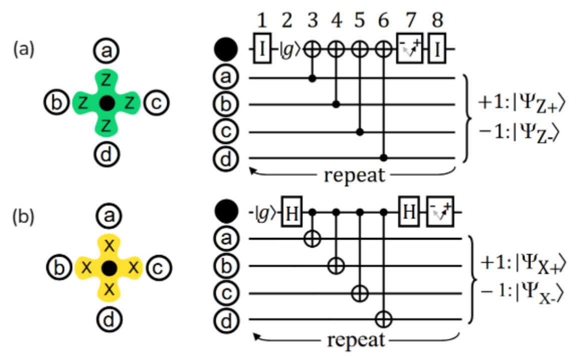


FIG. 2: X&Z-type stabilizers quantum circuit. (a) The Z-type stabilizer $Z_aZ_bZ_cZ_d$. Clearly, when the ancilla measurement result is $|0\rangle$, it signifies an eigenvalue of $+1$; otherwise, it's -1. (b) A similar implementation for an X-type stabilizer. The additional I gate in Fig.a is to synchronize with the Hadamard gate applied in Fig.b, reducing errors caused by measurement timing mismatches.

By definition in Appendix A 1, to form a stabilizer Abelian subgroup, these stabilizers must commute pairwise. The proof is provided in the Appendix A 1.

The X-type and Z-type stabilizers are completely symmetric. We can rotate each edge by 90° around its midpoint. This exchanges plaquettes and vertices. Formally, this is called representing the operators on the dual lattice.

2. *Error Detection*

Inserting some Pauli X errors into the surface code, the syndrome diagram and corresponding errors are shown in Fig. 3. Yellow vertices represent some Z-type stabilizers S_i that anticommute with the error. They are part of the error syndrome, satisfying $\sigma_{S_i}(E) = 1$. They are often called excitations or defects.

We call this connected set of errors an error chain. It can be observed that defects typically appear at the endpoints of an error chain. Therefore, when the error chain forms a closed loop, the defects disappear. In this case, the error commutes with all stabilizers, i.e., $E \in \mathcal{C}(\mathcal{S})$.

3. *Logical Qubits and Logical Operators*

Represent all data qubits with a logical state $|u\rangle_L$. In Fig. 4a, the error chain cannot be generated by any stabilizer and commutes with all stabilizers. Acting with it flips the logical state $|u\rangle_L$ without changing the stabilizer measurement results. It is called a logical X operator(X_L). A chain connecting the left and right boundaries or the top and bottom boundaries is a non-trivial logical operator.

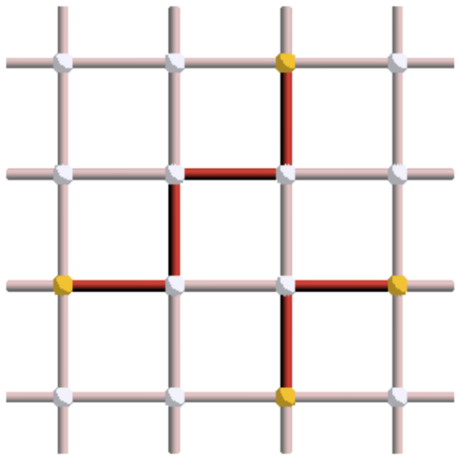


FIG. 3: X error and syndrome

For the surface code in the diagram, there are horizontal and vertical logical operators X_{1L}, Z_{1L} and X_{2L}, Z_{2L} respectively, encoding 2 logical qubits. The code distance of this surface code is 4 (since it connects top-bottom and left-right).

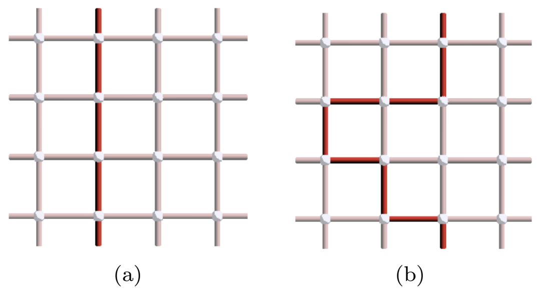


FIG. 4: Logical X operators. (a) is a standard logical X operator. (b) simply adds some X-type stabilizers with the same acting result.

B. Surface code decoding

According to the error correction theory discussed in Appendix A 4, stabilizer measurements produce an error syndrome $\sigma(E)$. The decoder must use this syndrome to select a correction operator R such that $RE \in S$.

Suppose that we observe the syndrome shown in the yellow vertices of Fig. 5. There can be many possible error chains that produce this syndrome, such as the three diagrams. Assume that the true error is the first diagram, and we choose the correction operator as the second diagram. The result of applying it is equivalent to a stabilizer and the correction succeeds. However, if we choose the third diagram for correction, the result is obviously a logical X operator. Correction fails, producing a logical error.

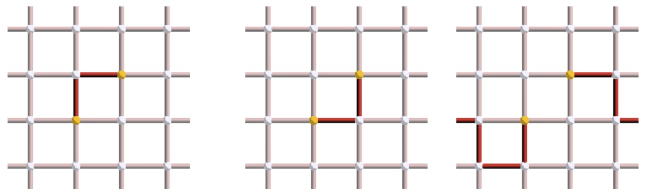


FIG. 5: Three possible error chains producing the same syndrome.

Therefore, decoding means finding the most probable error, which is equivalent to finding the smallest weight error consistent with the syndrome (shorter error chains have a higher probability). For the surface code, this translates to matching defects with minimum-weight chains, which is entirely equivalent to solving the minimum-weight perfect matching (MWPM) problem.

The decoding process can be decomposed into two parts with a MWPM decoder[5]:

1. Use the Local Dijkstra algorithm to find the shortest paths connecting the defects, constructing a complete graph.

2. Use the Blossom algorithm[6] or other optimized versions[7, 8] to obtain a set of perfect matches. Based on step 1, construct the corresponding error chain to form the correction operator.

C. Statistical model for the logical error rate

In practice, the surface code exhibits a threshold phenomenon. The threshold is a critical point for its physical error rate. When the physical error rate exceeds this threshold, the logical error rate increases with code distance; otherwise, increasing the code distance exponentially reduces the logical error rate, enabling fault-tolerant quantum computation.

It is important to note that the surface code does not have a single threshold: it depends significantly on the noise channel and decoder used.

The magnitude of the logical error rate can be estimated using simple statistical arguments[3], considering only errors on the data qubits. Chains of errors will give the same measurement outcomes if they are complementary in the way shown in Fig. 7. The MWPM decoder will misidentify the error type if the true error is Fig. 7b, leading to a logical error.

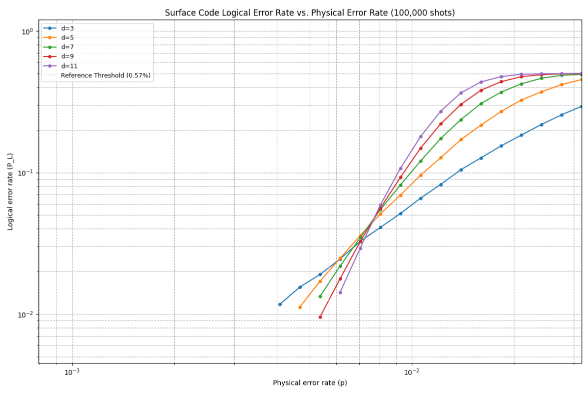


FIG. 6: Numerical simulations of surface code error rates, and how these error rates scale with the distance d of the array. Per-step error rates p less than the per-step threshold error rate of $p_{\text{thr}} = 0.57\%$ (dashed line) yield surface code logical error rates P_L that vanish rapidly with increasing d.

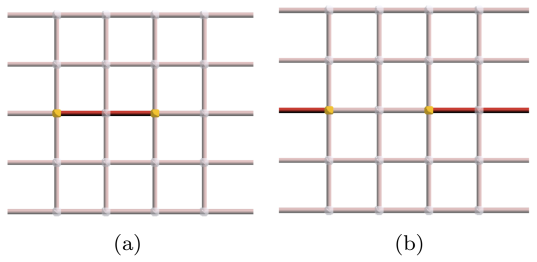


FIG. 7: An example where two Z-type stabilizers report errors in a single row of a 2D array. The code distance of this surface code with open boundaries is 5. This error report could be generated by (a) two pauli X errors appearing in the same surface code cycle on the 2nd and 3rd data qubit from the left, or (b) three X errors appearing in the other three data qubits in the row.

Longer error chains could also lead to misidentification. However, since their probability decays exponentially with length, they are negligible. Thus, the logical error rate can be obtained by calculating the probability of this type of error chain occurring:

$$P_L = d \cdot \frac{d!}{(d-d_e)! \cdot d_e!} \cdot p^{d_e}, \tag{2}$$

where $d_e = \lceil \frac{d}{2} \rceil$ and p is the physical error rate.

Taking into account the threshold phenomenon, we can obtain

$$P_L \propto (\frac{p}{p_{\text{thr}}})^{d_e}. \tag{3}$$

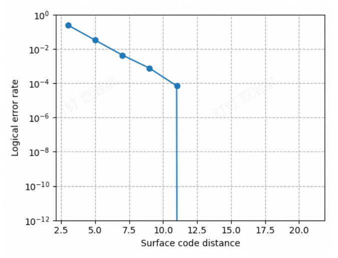


FIG. 8: Logic Error Rate vs. Surface Code Distance

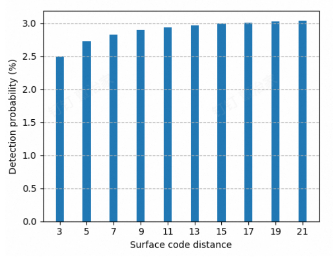


FIG. 9: Detection Probability vs. Surface Code Distance

III. SIMULATION

A. A Surface Code Memory Below Threshold

We define our error model with the following parameters:

- single-qubit gate depolarizing rate: 0.001
- reset/idle error(bit-flip) probability: 0.002
- pre-measurement bit-flip error rate: 0.004
- data depolarizing error rate preceding two-qubit gates: 0.0028

With the number of rounds set to 250 and the number of shots to 10,000, we obtained the following results:

According to Figure 8, we observe that the logical error rate significantly decreases with an increase in the surface code distance, d. This downward trend appears approximately linear on a semi-logarithmic scale, indicating that the logical predictions of quantum error correction.

As the code distance increases from 3 to 11, the logical error rate decreases by approximately four orders of magnitude. Notably, when applying a surface code with a distance of d = 13, the logical error rate drops precipitously, implying that logical errors are almost entirely suppressed and highly unlikely to occur at this distance. Conversely, at d = 3, the logical error rate is relatively high. We attribute this to the fact that a surface code with distance 3 can only correct a single physical error, while multiple physical errors or more complex error patterns are likely to result in logical errors.

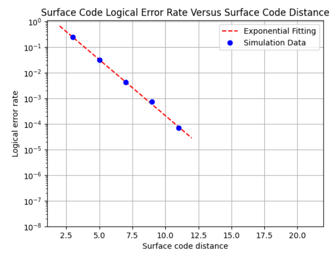


FIG. 10: Surface Code Logical Error Rate Scaling with Distance

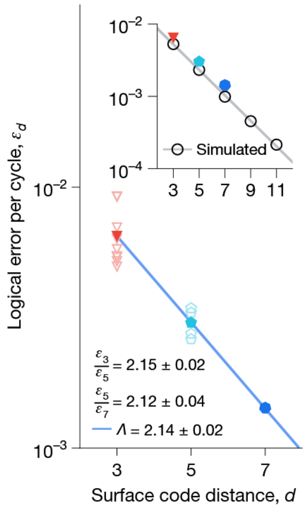


FIG. 11: Logic Error Rate vs. Surface Code Distance (Google's work)

Furthermore, according to Fig. 9, the detection probability exhibits a slight upward trend with increasing code distance, rising from approximately 2.5% at d = 3 to about 3.05% at d = 21.

To quantitatively analyze the exponential decay of the logical error rate with increasing code distance, we performed an exponential fit on the simulation data points. As shown in the Fig. 10, the exponential fitting curve aligns well with the simulation data points over the selected range of code distances(d = 3 to d = 11). This confirms that the logical error rate adheres to an exponential decay pattern. The fitting parameter(let's denote it λ) was calculated as 1.0075 ± 0.0010. Since the code distance d increases in steps of 2, the actual logical error rate suppression factor, Λ, is interpreted from this parameter as 2λ. Thus, $\Lambda = 2.0150 \pm 0.0020$.

A comprehensive comparison reveals that our simulation results are consistent with the work conducted by Google(Fig. 11):

- Below a certain physical error rate threshold, the logical error rate decays exponentially with increasing code distance. The suppression factor Λ reported by Google was 2.14 ± 0.02, whereas our work yielded a suppression factor of $2.0150 \pm$

0.0020. We attribute this discrepancy to differences in the noise simulation models; our depolarizing noise model may not fully capture the complex, non-local, or correlated error mechanisms present in Google's physical systems (e.g., crosstalk, coherent errors, impacts from high-energy particles). Furthermore, we employed only a standard Minimum-Weight Perfect Matching (MWPM) decoder. Advanced decoders, or those optimized for specific hardware noise characteristics (such as Google's neural network decoder or an enhanced MWPM decoder), could potentially offer different performance levels.

- The detection probability shows a slight upward trend with increasing code distance. This phenomenon is similar to the trend observed by Google in their research, which they attributed to finite-size effects and potential parasitic couplings. At larger code distances, this growth tends to saturate or level off.

B. Real-time Decoding

We configured the noise model as follows: a depolarizing noise probability of 0.0028 was applied after each Clifford gate execution, and a depolarizing noise probability of 0.0028 was applied before the measurement at the beginning of each round. For error correction on the surface code, we utilized the PyMatching library, which is based on the minimum-weight perfect matching (MWPM) algorithm, as the decoder.

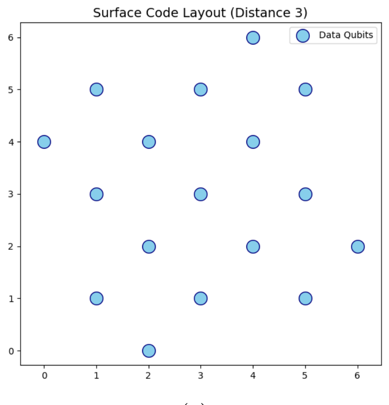
We set the surface code distances to d = 3, 5, and 7, respectively, and the number of samples was set to 100,000. The following figures illustrate the surface code layouts for these distances, showing only the data qubits.

The figures below(Fig. 13) clearly demonstrate the activation states of different detectors across various shots. This also provides an intuitive visualization of the noisy nature of quantum systems. It can be observed that under our defined noise model, the error distribution is relatively sparse.

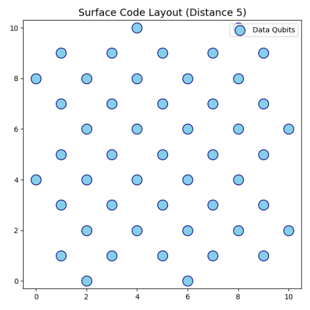
The following diagrams(Fig. 14) display the paths before decoding; the blue dashed lines represent all possible paths. When invoking the PyMatching library's decoding function, the algorithm transforms the quantum error syndrome (i.e., activated stabilizers) into a matching problem in graph theory. By finding the minimum-weight matching, it infers the likely physical error chains and, consequently, predicts the occurrence of logical errors.

After decoding, we obtained the following computational results:

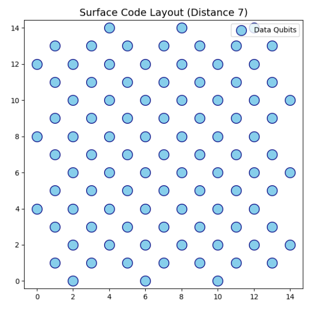
- For a surface code distance of d = 3, the logical error rate was 4.0×10^{-3}
- For a surface code distance of d = 5, the logical error rate was 2.0×10^{-3}



(a)



(b)



(c)

FIG. 12: Surface Code Layout. (a) d = 3. (b) d = 5. (c) d = 7.

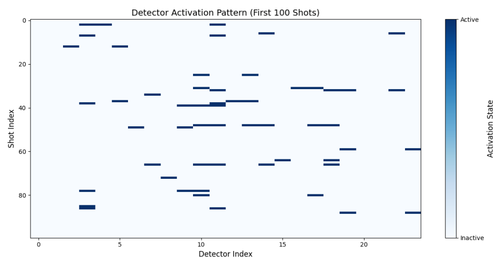
- For a surface code distance of d = 7, the logical error rate was 1.0×10^{-3}

In Google's real-time decoding experiments on the Willow Chip(Fig. 15), Google reported a logical error rate of approximately 0.35% ± 0.01%, which is 3.5×10^{-3}, for a distance-5 ($d = 5$) surface code. Our simulated logical error rate for $d = 5$ (2.0×10^{-3}) is lower than Google's experimentally achieved real-time decoding result.

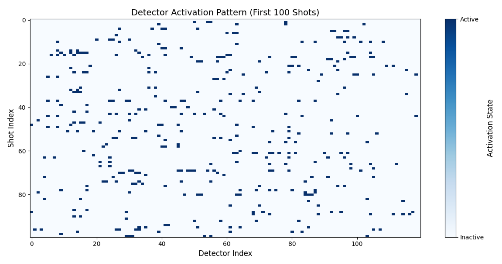
Several factors may contribute to this difference:

- Firstly, the error model used in our simulation may not perfectly align with that of Google's physical quantum processor. The error behavior in physical quantum processors is typically more complex and challenging to model with high precision than simplified theoretical models. Google's work details various error sources and their characterization, including Pauli errors for gates, idle errors, and measurement errors, which contribute to the overall physical error rate.

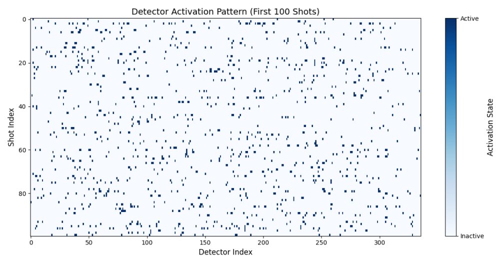
- Secondly, Google's real-time decoder employs an optimized sparse Blossom algorithm and might include additional hardware-specific optimizations. In contrast, our simulation uses a standard MWPM algorithm via PyMatching. Different decoding



(a)



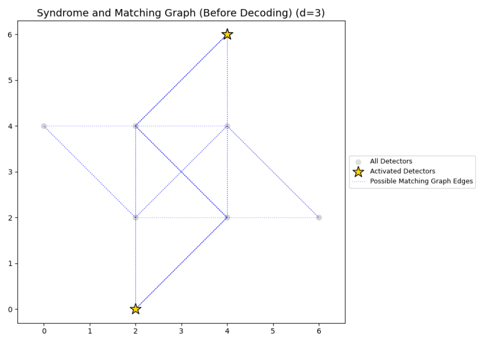
(b)



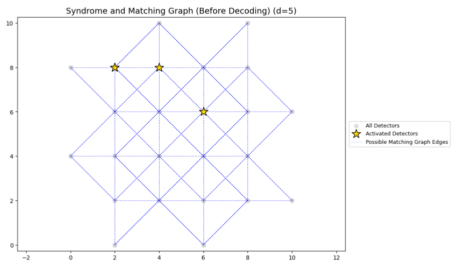
(c)

FIG. 13: Detector Activation Pattern (First 100 Shots). (a) d = 3. (b) d = 5. (c) d = 7.

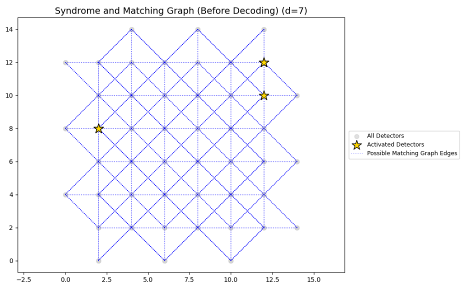
strategies and implementation details directly impact the final logical error rate. Google's paper also indicates that their offline, more accurate neural network decoder can achieve even lower error rates than their real-time decoder, with the neural network decoder achieving $\epsilon_5 = 0.269\% \pm 0.008\%$ for d = 5 offline. Google also utilized an ensembled matching synthesis decoder for offline analysis.


Syndrome and Matching Graph (Before Decoding) (d=3)
All Detectors
Activated Detectors
Possible Matching Graph Edges

(a)


Syndrome and Matching Graph (Before Decoding) (d=5)
All Detectors
Activated Detectors
Possible Matching Graph Edges

(b)


Syndrome and Matching Graph (Before Decoding) (d=7)
All Detectors
Activated Detectors
Possible Matching Graph Edges

(c)

FIG. 14: Detector Activation Pattern (First 100 Shots). (a) d = 3. (b) d = 5. (c) d = 7.

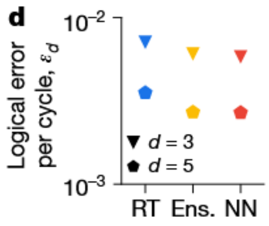


FIG. 15: Comparison of Different Decoder Performances (Google's Work)

IV. CONCLUSION

In this work, we have systematically investigated the quantum error correction performance of the surface code under different noise models and code distances through numerical simulations, and compared our findings with the experimental results from Google's Willow Chip.

In the simulations for "a surface code memory below threshold", we observed a significant exponential decrease in the logical error rate as the code distance increased (from d = 3 to d = 11), by approximately four orders of magnitude. At d = 13, logical errors were almost entirely suppressed. Concurrently, the detection probability showed a slight increase with increasing code distance. Through exponential fitting, we obtained an error suppression factor Λ of 2.0150 ± 0.0020. These trends are qualitatively consistent with Google's experimental results, which also show exponential decay of logical error rates below threshold and a modest increase in detection probability with distance. However, our Λ value differs from Google's reported 2.14 ± 0.02, likely due to the depolarizing noise model we employed not fully capturing the complex, non-local, or correlated error mechanisms present in the physical system, and our use of a standard MWPM decoder, whereas Google may have utilized more advanced or optimized decoding strategies.

In the simulations for "real-time decoding" scenario, we evaluated surface codes with distance d = 3, 5, 7. The results indicated that the logical error rate also decreased with increasing code distance; for instance, at d = 5, our logical error rate was 2.0×10^{-3}. This value is slightly lower than the 3.5×10^{-3} reported by Google in their Willow chip real-time decoding experiments. We surmise that this difference is similarly related to simplifications in the error model and differences in the decoding algorithms. Google's real-time decoder employs an optimized sparse Blossom algorithm and potentially includes hardware-specific optimizations, whereas our simulation used an MWPW decoder via PyMatching.

In summary, our simulations have successfully reproduced the core error correction characteristics of the surface code and exhibit good qualitative agreement in trends with Google's experimental findings. The quantitative discrepancies between simulation results and experimental data highlight the critical importance of accurate physical noise modeling and advanced decoding algorithms in achieving high-performance, practical quantum error correction systems.

Future work could focus on developing more realistic noise models that incorporate a broader range of error types and their correlations found in actual physical systems, as well as exploring and implementing more efficient decoding algorithms, to more accurately predict and approach the performance of physical experiments.

Appendix A: Stabilizer Theory[9]

1. Pauli Group, Stabilizer Group, Codespace

The Pauli group $\mathcal{P}_n$ is the n-fold tensor product of Pauli operators (X, Y, Z),

$$\mathcal{P}_n = \{\pm 1, \pm i\} \cdot \{\pm 1, \pm iI, \pm X, \pm Y, \pm Z\}^{\otimes n}. \tag{A1}$$

It satisfies:

1. Any two elements in $\mathcal{P}_n$ either commute or anti-commute
2. Any element is either Hermitian or anti-Hermitian
3. Any element is unitary

A stabilizer group $\mathcal{S}$ is an Abelian subgroup of the Pauli group where $-I \notin \mathcal{S}$.

Definition A.1 (Codespace)**.** *The codespace $\mathcal{T}(\mathcal{S})$ is the space of states invariant under all stabilizers:*

$$\mathcal{T}(\mathcal{S}) = \{|\psi\rangle \mid S|\psi\rangle = |\psi\rangle, \forall S \in \mathcal{S}\}. \tag{A2}$$

Theorem A.1. *In the surface structure described in Fig. 1, X-type stabilizers and Z-type stabilizers always commute.*

Proof. Clearly, X-type stabilizers commute with each other, and Z-type stabilizers do so. For an X-type stabilizer and a Z-type stabilizer always have either zero or two edges in common.

If they share no common edge, they act on completely disjoint sets of data qubits and obviously commute. If they share two common edges, meaning they measure the same two data qubits (assume a, c are the shared edges), then

$$\begin{aligned}
&(X_a X_b X_c X_d)(Z_a Z_c Z_e Z_f) \\
&= (X_a Z_a) \otimes X_b \otimes (X_c Z_c) \otimes X_d \otimes Z_e \otimes Z_f \\
&= -(Z_a X_a) \otimes X_b \otimes -(Z_c X_c) \otimes X_d \otimes Z_e \otimes Z_f \\
&= (Z_a Z_c Z_e Z_f)(X_a X_b X_c X_d).
\end{aligned} \tag{A3}$$

Simply put, the anticommuting pairs X_a, Z_a and X_c, Z_c cancel each other out, resulting in the X-type stabilizer commuting with the Z-type stabilizer. □

2. Centralizer, Normalizer

The centralizer $\mathcal{C}(\mathcal{S})$ consists of all Pauli operators that commute with every element of $\mathcal{S}$:

$$\mathcal{C}(\mathcal{S}) = \{E \in \mathcal{P}_n \mid [E, S] = 0, \forall S \in \mathcal{S}\}. \tag{A4}$$

The normalizer $\mathcal{N}(\mathcal{S})$ consists of operators that preserve $\mathcal{S}$ under conjugation:

$$\mathcal{N}(\mathcal{S}) = \{E \in \mathcal{P}_n \mid E^\dagger S E \in \mathcal{S}, \forall S \in \mathcal{S}\}. \tag{A5}$$

Theorem A.2. *In the Pauli group, $\mathcal{C}(\mathcal{S}) = \mathcal{N}(\mathcal{S})$.*

Proof. For any $E \in \mathcal{P}_n$ and $S \in \mathcal{S}$:

$$E^\dagger S E = \pm E^\dagger E S = \pm S. \tag{A6}$$

Since $-I \notin \mathcal{S}$, we must have $E^\dagger S E = S$, which implies $[E, S] = 0$. Thus $\mathcal{N}(\mathcal{S}) \subseteq \mathcal{C}(\mathcal{S})$. The reverse inclusion is immediate. □

For any $E \in \mathcal{C}(\mathcal{S}) \setminus \mathcal{S}$, it represents an undetectable logical error. That is, E acting on the codespace $\mathcal{T}(\mathcal{S})$ remains within the codespace but changes the encoded state. Hence, it can be called a logical operator, capable of performing logical (X, Y, Z) operations.

Definition A.2 (Code Distance)**.** *The weight $|P|$ of a Pauli operator $P \in \mathcal{P}_n$ is the number of qubits on which it acts non-trivially (not as identity).*

The code distance d is the minimum weight among elements in $\mathcal{C}(\mathcal{S}) \setminus \mathcal{S}$:

$$d = \min\{|P| \mid P \in \mathcal{C}(\mathcal{S}) \setminus \mathcal{S}\}. \tag{A7}$$

3. Generators and Error Syndrome

The generators $S_1, S_2, ..., S_r$ of the stabilizer group $\mathcal{S}$ are called check operators.

Definition A.3 (Error Syndrome)**.** *For a Pauli error $E \in \mathcal{P}_n$, the syndrome $\sigma(E)$ is defined as:*

$$\sigma_{S_i}(E) = \begin{cases} 0 & \textit{if } [S_i, E] = 0 \\ 1 & \textit{if } \{S_i, E\} = 0 \end{cases} \tag{A8}$$

$$\sigma(E) = (\sigma_{S_1}(E), \dots, \sigma_{S_r}(E)) \tag{A9}$$

4. Error Correction Theory

Given an error syndrome $\sigma(E)$, the decoder uses it to select a correction operator $R \in \mathcal{P}_n$. If $RE \in \mathcal{S}$, then $RE\,|\psi\rangle = |\psi\rangle$, and correction is successful. If $RE \in \mathcal{P}_n \setminus \mathcal{C}(\mathcal{S})$, then $RE\,|\psi\rangle$ is not in the $\mathcal{T}(\mathcal{S})$, representing an incorrect encoding, and correction fails.

Theorem A.3. *If correction operator R has $\sigma(R) = \sigma(E)$, then $RE \in \mathcal{C}(\mathcal{S})$.*

Proof. For any $S \in \mathcal{S}$:

$$\begin{aligned} RES &= (-1)^{\sigma_S(E)} RSE \\ &= (-1)^{\sigma_S(R)+\sigma_S(E)} SRE \end{aligned} \tag{A10}$$

Since $\sigma_S(E) = \sigma_S(R)$, we have $[RE, S] = 0$ for all $S \in \mathcal{S}$. $\square$

If now $RE \in \mathcal{C}(\mathcal{S}) \setminus \mathcal{S}$, a logical error occurs, which changes the logical state of the logical qubit.

Appendix B: Codes

We use Stim[4] and Pymatching[5] to do the simulation. Our code is available in .

[1] Google Quantum AI and Collaborators. Quantum error correction below the surface code threshold. *Nature* **638**, 920-926 (2025).

[2] Dennis, E., Kitaev, A. Y., Landahl, A. & Preskill, J. Topological quantum memory. *J. Math. Phys.* **43**, 4452-4505 (2002).

[3] Fowler, A. G., Mariantoni, M., Martinis, J. M. & Cleland, A. N. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A* **86**, 032324 (2012).

[4] Gidney, Craig. Stim: a fast stabilizer circuit simulator. *Quantum* **5**, 497 (2021).

[5] Higgott, Oscar. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching. *ACM Transactions on Quantum Computing* **3**, 16 (2022).

[6] Edmonds, Jack. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, 125 (1965).

[7] Higgott, Oscar and Gidney, Craig. Sparse Blossom: correcting a million errors per core second with minimum-weight matching. *Quantum* **9**, 1600 (2025).

[8] Wu, Yue and Zhong, Lin. Fusion Blossom: Fast MWPM Decoders for QEC. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)* 928-938 (IEEE, 2023).

[9] Gottesman, D. Stabilizer Codes and Quantum Error Correction. Preprint at (1997).