Tomas Bruno
EECS 498 - Hybrid Quantum Systems
Final Project Report

A Circuit Level Noise Analysis of the Clique Decoder

1. Introduction

Quantum computing has the potential to revolutionize computation by exploiting unique quantum mechanical properties like superposition, entanglement, and interference. This technology offers exponential speedup in classically intractable problems across fields like cryptography, quantum chemistry, and combinatorial optimization.

In today's Noisy Intermediate Scale Quantum (NISQ) era, quantum computers are not capable of executing large-scale quantum algorithms due to environmental interactions that alter the fragile quantum state. Qubits are inherently susceptible to a wide range of complex noise mechanisms such as decoherence, crosstalk, gate infidelity, measurement error, and more. These errors can accumulate rapidly, corrupting the qubit's state and thus making quantum computers impractical outside of the NISQ era.

Quantum Error Correction (QEC) is a framework that can lower qubit error rates to levels needed by large-scale quantum algorithms. First, physical qubits are encoded into logical qubits according to a particular QEC code. Then, a circuit is performed which entangles stabilizer qubits with their associated data qubits to obtain a syndrome. Next, a decoder predicts the location of physical qubit errors based on the measured syndrome and recommends a set of corrections to maintain the state of the logical qubit. Using QEC, logical qubit error rates will approach the Fault Tolerant Quantum Computing (FTQC) era, where quantum computers can support algorithms like Shor's, Grover's, and more, demonstrating true quantum supremacy.

Decoding is a crucial step in QEC, and the logical error rate depends on the accuracy of the decoder. However, decoding is especially challenging because qubits have limited coherence times and are continuously exposed to spatially and temporally correlated noise, requiring decoders to act quickly and accurately under hardware and latency constraints. Often, designers of decoders will approximate qubit error mechanisms under simpler noise models. While this does drastically reduce the decoding problem, it hides complex errors that could propagate through the syndrome extraction circuit, possibly hindering the performance of the decoder and thus the logical error rate.

In this project, my overarching goal is to understand how realistic noise models can impact decoder performance for the surface code. My main objectives are summarized as follows:

1. Simulate the surface code under various noise models using Stim.
2. Analyze the surface code's syndrome distribution under various noise models.
3. Evaluate the Clique decoder's coverage under a realistic noise model.
4. Motivate development of a hook-aware Clique decoder.

2. Background and Motivation

2.1 Quantum Error Correction

In order to achieve the logical error rates needed for FTQC applications, several noisy physical qubits are encoded into a logical qubit according to a particular QEC code. Physical qubits are composed of data qubits and stabilizer qubits. Data qubits contain the data needing protection, and stabilizer qubits are entangled with data qubits during syndrome extraction to obtain error information non-destructively. The measured syndrome is then sent to a decoder, which will predict the location of data errors based on known error mechanisms and offer a correction. The distance of a QEC code is the minimum number of physical errors needed to cause a logical error. Consequently, the threshold theorem states that if the physical error rate is below a certain threshold, then increasing the code distance will suppress the logical error rate. For this reason, achieving FTQC is possible. By selecting a QEC code with a threshold above the near-term physical error rates, one could continuously add more qubits to increase the distance, arbitrarily suppressing the logical error rate.

2.2 Surface Code

The surface code is a popular choice for many architectures due to its relatively high threshold (~1%), nearest neighbor connectivity, and efficient decoding protocols. The topology involves a distance d square grid of data qubits, interconnected by X and Z stabilizers, which detect Z and X errors respectively. During syndrome extraction, each stabilizer is entangled with its neighboring data qubits and then measured. For any given stabilizer, if an odd number of entangled data qubits has the error corresponding to the stabilizer's basis, then the stabilizer's measurement will be 1. For this reason, chains of consecutive data errors on the surface code grid will only appear as endpoints during syndrome extraction. A logical error occurs if a chain of errors reaches from one boundary to another. Thus, a common algorithm for decoding surface code patches is minimum-weight perfect matching (MWPM), which connects endpoints together such that the sum of the distance between all of the matched endpoints is minimized. [1]

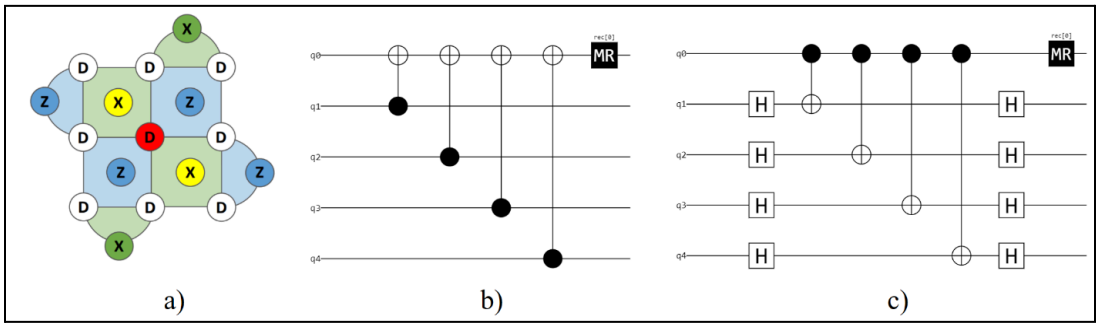


Fig. 1: a) Distance 3 surface code, with Z error on center data qubit, and surrounding X stabilizers activated [1]
b) Z syndrome extraction circuit, c) X syndrome extraction circuit, q0=stabilizer, q1-4=data

2.3 Noise Models

Physical qubits have a variety of noise mechanisms that are difficult to simulate classically. Decoders are also limited by the syndrome extraction window, and thus cannot consider all error mechanisms when predicting error locations. Consequently, realistic noise mechanisms are approximated based on averaging methods to create simpler noise models for simulation and decoding graphs. A common error distribution to use is the depolarizing noise model, which considers X, Y, and Z errors to occur with equal probability.

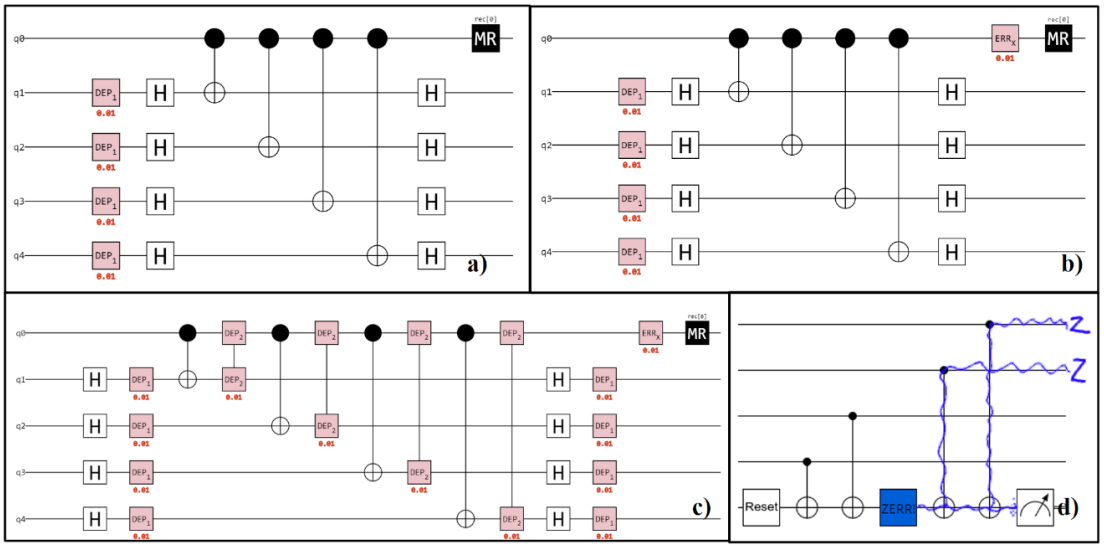


Fig. 2: Noise Models for Z-stabilizer syndrome extraction circuit + Hook error
a) Code Capacity, b) Phenomenological, c) Circuit-level, d) Hook error propagates 2 data errors [4]

When simulating quantum circuits, there are three popular models for implementing noise with varying degrees of complexity (Fig. 2). The first is the code-capacity noise model, where data qubits have a probabilistic depolarizing error applied at the beginning of each syndrome extraction round. The second is the phenomenological noise model, which adds the possibility of bit-flip errors on stabilizer qubits right before measurement. Finally, the circuit level noise model applies depolarizing noise after each circuit operation, matching the behavior of realistic noisy gates. An important observation with the circuit-level noise model is when a Z error occurs on a Z stabilizer (or X error on X stabilizer) as a result of a noisy two-qubit gate in mid-syndrome extraction, the process of phase kickback will propagate the error to the data qubits during their subsequent entanglement, leading to two data errors in the following round. [2] This is known as a hook error, where a single error mechanism propagates as two data errors. [4]

2.4 Clique Decoder

Clique is a resource-efficient decoder that can detect and correct common, trivial errors on-chip, while sending rare complex errors to an off-chip decoder. [1] Motivated by the "better-than-worst-case" philosophy, this decoder exploits the fact that at near-term code distances and physical error rates, a large fraction of syndrome shots feature isolated data errors. These trivial, length-1 errors are detected by clique on chip and corrected, reducing the bandwidth going to off-chip decoders. However, Clique's original analysis used a phenomenological noise model, which does not consider mechanisms that lead to hook errors. Hook errors are especially dangerous to Clique, as the resulting two data errors will form an error string deemed complex, requiring the entire code block to be sent for complex decode. A single error mechanism not captured by this simpler noise model would have a significant impact on Clique's syndrome coverage.

To better understand the coverage of the Clique decoder, this work analyzes the syndrome distribution of the surface code under a circuit-level noise model using Stim and Pymatching, as explained in the following section.

3. Proposed Method

3.1 Stim

Stim is a tool for high performance simulation and analysis of quantum stabilizer circuits. Some of Stim's key features that I took advantage of for this project include:

1. Pre-generated QEC codes: Stim provides a set of fully implemented codes like the repetition code, unrotated and rotated surface code, and the color code. There is also functionality to build custom circuits with coordinate systems, measurement blocks, and detector annotations.
2. Basic noise parameters: The pre-generated QEC codes come with a basic set of noise parameters to fine tune the simulated noise model. When simulating a circuit with noise, the user-defined noise parameters define the probability that the error takes place in each of its instances within the circuit. These parameters are summarized in Table 1, along with which noise model uses them.

Noise Parameter	Description	CC	Phen.	Circ.
`after_clifford_depolarization`	Applies single and two qubit depolarizing noise to qubits after clifford operation			☐
`before_round_data_depolarization`	Applies depolarizing noise to all data qubits at the beginning of syndrome extraction round	☐	☐	
`before_measure_flip_probability`	Applies probabilistic bit flip error before measurement gates		☐	☐
`after_reset_flip_probability`	Applies probabilistic bit flip error after reset gates (stabilizers are reset after measurement)			☐

Table 1: Noise parameters in pre-generated stim surface code, and which noise models use them (CC = code capacity, phen = phenomenological, Circ. = circuit-level)

3. Simple Decoder Configuration: Stim circuits often annotate stabilizer measurements with detectors. Detectors compare two measurements that are expected to be equal, like two stabilizer measurements separated by a single round. Stim can automatically convert these detector circuit annotations (which contain information like stabilizer coordinates, round number) to a Detector Error Model (DEM), which is a graph featuring detectors as nodes and error mechanisms as edges. Decoders utilize these graphs to detect and correct errors based on the underlying error mechanisms.
4. Realistic noise extension: Because stim is open source, many QEC enthusiasts have developed their own libraries to simulate even more realistic noise models than those provided by the pre-generated codes. In this paper, I use Jason Chadwick's implementation that includes decoherence (T1/T2) time, and user-specified means and standard deviations for error rates to model fluctuations in qubit quality.

3.2 Pymatching

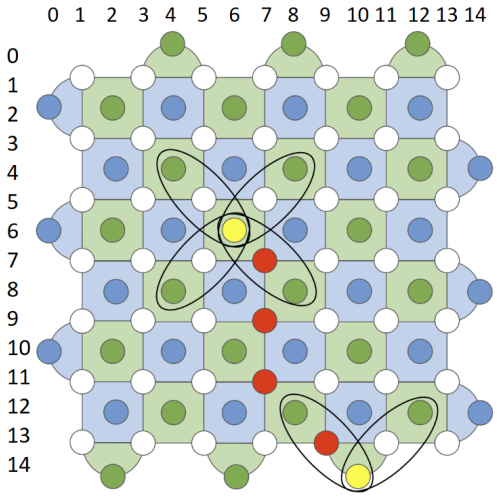
Pymatching is a fast python library for decoding QEC codes using MWPM. Given the syndrome from a quantum circuit execution, the MWPM decoder finds the most probable set of errors using techniques like the blossom algorithm, which is 100-1000x faster than previous iterations. When integrated with Stim, Pymatching can return the indices of matched detectors from a given syndrome, which can be mapped to the stabilizer coordinates within the circuit. Although Pymatching does guarantee correct detections, it is

highly accurate and optimal for the purposes of analyzing syndrome distribution under more complex noise models.

3.3 Finding Error Chain Length

An error chain is defined as a string of consecutive data errors on the surface code. Clique is designed to handle "trivial" length-1 error chains. However, hook errors will propagate as "complex" length-2 error chains. In order to understand how hook errors affect the syndrome coverage of Clique, one method is to simulate the surface code under phenomenological noise and circuit-level noise and see how the relative fraction of length-1 to length-2 error chains changes between the noise models. To do this, I initialized stim's pre-generated surface code under the two noise models, sampled the circuit's detection events for code distance number of rounds, and converted the circuit into its detector error model to feed into Pymatching. Then, I found the matched detection events using the sampled syndrome and Pymatching's `decode_to_matched_dets_array()`.

To calculate the error chain length between matched detection events, it is important to recall that the surface code has an interleaving pattern of data qubits and stabilizer qubits, and since the error chain is only concerned with data errors, the distance between matched detector coordinates must be divided by 2 to ensure data error chain length. Another important note is that data errors can chain together diagonally across stabilizers, which simplifies the distance calculation. Lastly, it is possible that a detector is matched to a boundary, in which case the error chain length is the number of data qubits between the stabilizer and the closest boundary. To find the maximum length error, I iteratively calculate the error chain length between matched detectors, comparing to the current max length error until all lengths are calculated. For example, Fig. 3 shows the error chain length calculation for detectors with coordinates (6,6) and (14,10).




```python
# Function to calculate data error chain length between two
detector coordinates (det1, det2) separated in space and time.
def calculate_length_error(det1, det2, d):
   distance = 0
   # matched to boundary, distance must be nearest boundary
    if det2 == -1:
           dx = min(det1.x//2, (2*d+1 - det1.x)//2)
           dy = min(det1.y//2, (2*d+1 - det1.y)//2)
           dt = min(det1.t, d + 1 - det1.t[2])
           distance = min(dx, dy, dt)
    else:
           # Division by 2 for interleaving data qubits
           dx = abs(det1.x[0] - det2.x[0]) / 2
           dy = abs(det1.y[1] - det2.y[1]) / 2
           dt = abs(det1.t[2] - det2.t[2])
           distance = max(dx, dy, dt)
   return distance
```

Fig. 3: Calculating error chain length using surface code grid and matched detector coordinates. The example shows matched detector coordinates in yellow ((6,6) and (14,10)), and the correctly calculated error chain between them with length 4. [1]

4. Evaluation Methodology

Equipped with Stim, Pymatching, and a method to determine the maximum length error chain in a syndrome block, we can now evaluate the error chain length distribution.

4.1 Monte Carlo Simulation

The Monte Carlo method is used for simulating random syndromes many times and obtaining the approximate probability distribution of results [5]. Specifically, 100,000 shots are taken for each circuit. Each shot contains d rounds of syndromes, where d is the code distance.

4.2 Distance and Noise Parameter Selection

To gain an understanding of the surface code's max length error chain distribution at the d-round block granularity (as opposed to single round granularity analyzed by original Clique), I use a phenomenological noise model as a control group and sweep across distances 3, 7, 15, 29, representing near-term code distances. The chosen physical error rates are 0.001, 0.005, 0.01, and 0.015, representing near-term average error rates on physical qubits. For this noise model, the pre-generated Stim noise parameters `before_round_data_depolarization` and `before_measure_flip_probability` were set to the according error rates, with all other noise parameters set to 0. When analyzing the distribution under circuit-level noise, I chose to focus on distance 15 since it is a near-term distance and showed a percentage of length-1 and length-2 error chains in the phenomenological results. I also fixed `before_measure_flip_probability` to a realistic rate of 0.005. To see how hook errors add to the distribution, I swept across the `after_clifford_depolarization` parameter with values 0, 0.0001, 0.0005, 0.001, representing realistic near-term two-qubit gate infidelities. Finally, to understand the effects of fluctuating qubit quality and decoherence on the distribution, I simulated Chadwick's surface code at distance 15, with single qubit gate error rate at 0.0001, two-qubit gate error rate at 0.0005, and measurement error rate at 0.005, and observed how enabling T1/T2 decoherence affected the distribution.

4.3 Error Chain Length Validation

To verify the calculation of error chain lengths, I created thorough debug functions to print the locations of matched detector coordinates within surface code patches. This allowed me to hand calculate error chain lengths on small code distances and compare them with results from the function in Fig. 3. Another sanity check for the error chain length calculation is by observing the fraction of length-0 (no error) syndromes within the distribution under phenomenological noise. In this noise model, each of the $2d^2 - 1$ physical qubit has an error rate p, so the probability of having no errors in d rounds of measurement is $(1 - p)^{d(2d^2-1)}$. This fact is used in the results to verify syndrome distribution correctness.

5. Experiments and Results

5.1 Syndrome Distribution under Phenomenological noise:

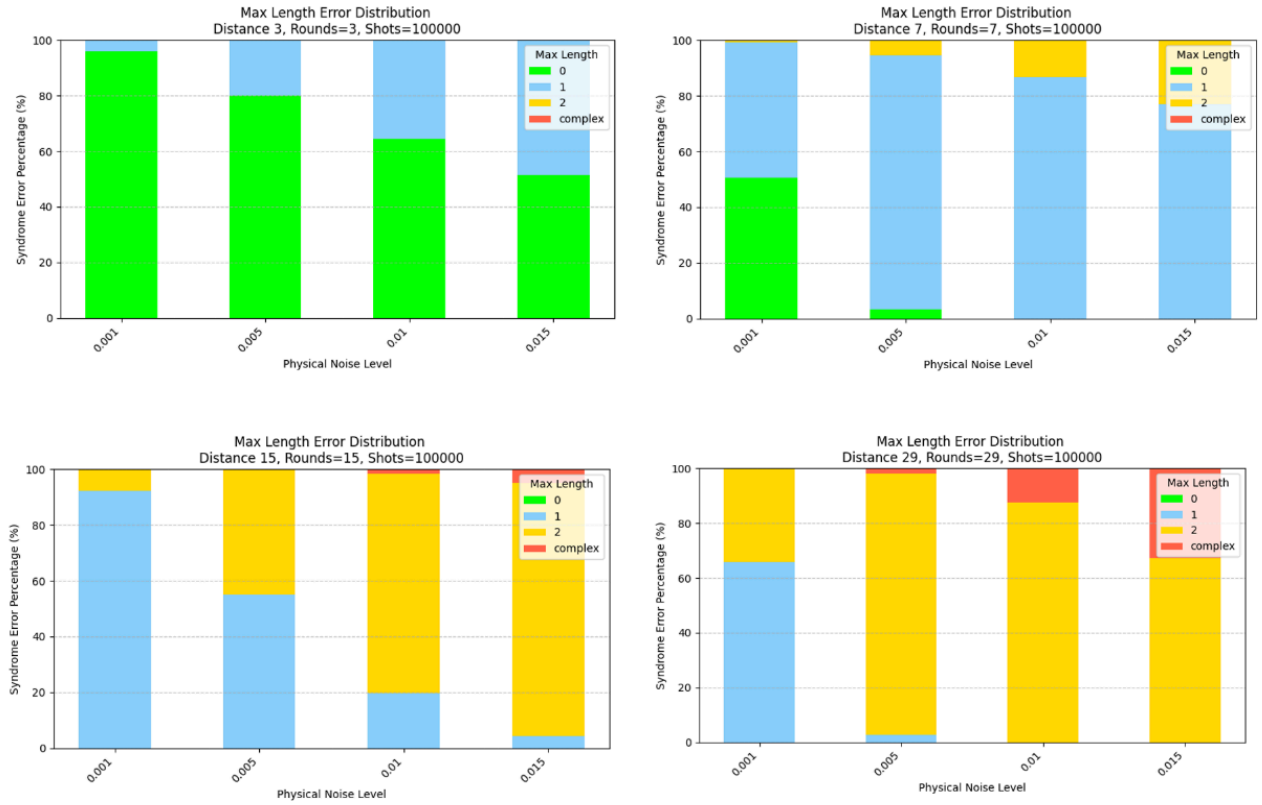


Fig. 4: Max length syndrome distribution under phenomenological noise, for distances 3, 7, 15, 29, and noise levels 0.001, 0.005, 0.01, and 0.015.

Under phenomenological noise, it is observed that near-term code distances at near-term average physical error rates experience a high fraction of length-1 max-length errors, which motivates the original Clique design. One major difference between this analysis and the motivational analysis behind Clique is that these results are for d rounds of measurement, whereas clique analyzed single syndrome extraction rounds. Although Clique corrects errors at the round granularity, any complex errors within a d-round block requires the entire code block to be sent to an off-chip decoder. This means we should be focused on the max-length error at the d-round block granularity. With more rounds of analysis, errors will line up to form complex error chains more often, leading to the results above which shows reduced Clique coverage from expectation under the phenomenological noise model. To verify the calculation of error length, see the syndrome distribution for d=7 with p=0.001. The expected fraction of max length-0 syndromes is $(1 - 0.001)^{7(2(7)^2-1)}$=0.506, which matches closely with the observed fraction.

5.2 Syndrome Distribution under Circuit-level Noise

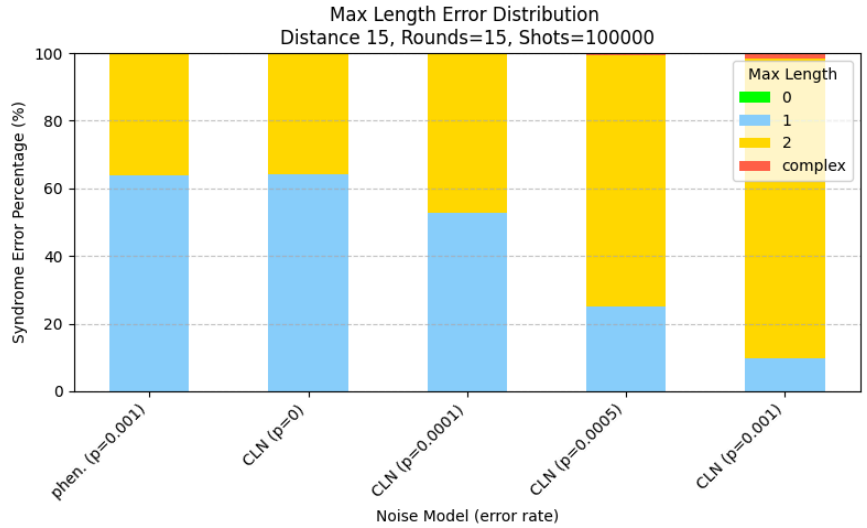


Fig. 5: Comparing syndrome distributions under phen. noise at p=0.001 and varying levels of circuit-level noise (CLN) for d=15

When noise is added to circuit-level operations, the relative fraction of length-2 errors increases dramatically. Fig. 5 demonstrates that the distributions for phenomenological and circuit-level noise models are identical when the two-qubit gates are noiseless (left two bars). However, when noise is added to the clifford operations during syndrome extraction, the fraction of length-2 errors increases dramatically. Although hook errors are not solely responsible for this increase in length-2 errors, they contribute significantly by propagating a single gate error into multiple data qubit errors, highlighting the importance of accounting for correlated error mechanisms in decoder design.

5.3 Syndrome Distribution under Realistic Noise Model

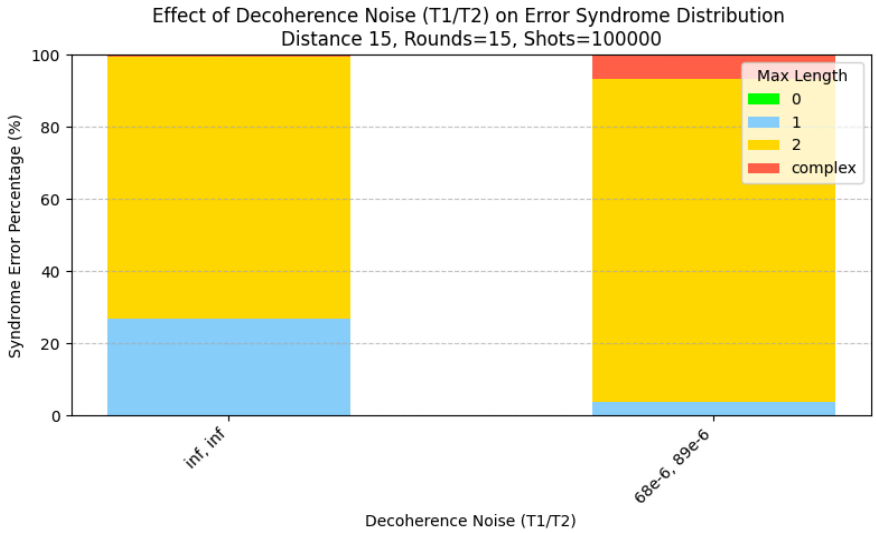


Fig. 6: Effect of decoherence and non-uniform qubit quality on distribution for d=15, p1=0.0001, p2=0.0005, p_meas=0.005,

Under a more sophisticated and hardware-realistic noise model that includes decoherence and non-uniform qubit quality, we can observe how T1/T2 times affect the syndrome distribution. First, I simulated with infinite T1/T2 times (no decoherence) to understand how different the distribution was with the pre-generated Stim implementation. We can see that the left bar in Fig. 6 closely resembles the fourth bar in Fig. 5, which makes sense since both have 2-qubit fate error rates at 0.0005. This shows how the much lower erring single-qubit gate and the varying qubit quality has a minor effect on the relative fraction of maximum error chain lengths. Next, I simulated with T1=68μs and T2=89μs, which come from Google's Willow performance metrics. We can see that length-1 error chains are much less frequent, likely because these decoherence times incur significant errors throughout the d measurement rounds, thereby increasing the fraction of length-2 and higher errors.

6. Conclusion

This work highlights the critical impact of realistic noise modeling on decoder performance for quantum error correction, specifically within the surface code architecture. Through extensive simulations with Stim and Pymatching, it was shown that circuit-level noise (particularly hook errors introduced by faulty two-qubit gates) significantly affects the syndrome distribution, leading to a much higher fraction of complex, length-2 error chains compared to the phenomenological noise model. This increase poses a serious challenge to the Clique decoder's efficiency, as these complex errors fall outside its intended correction scope, necessitating more frequent off-chip decoding. Furthermore, simulations with a realistic noise model that incorporates decoherence (T1/T2) and non-uniform qubit quality reaffirm that hardware imperfections further amplify complex error occurrences. Overall, this analysis motivates the need for hook-aware and more noise-resilient modifications to the Clique decoder. Future work should focus on developing lightweight detection schemes that can identify and handle hook errors on-chip without significant latency overhead, thus preserving the decoder's fast, resource-efficient nature even under realistic noise conditions.

7. Code

The following code repositories were used in the evaluation and analysis of this report:

1. Stim: https://github.com/quantumlib/Stim
2. Pymatching: https://github.com/oscarhiggott/PyMatching
3. Jason Chadwick's Realistic Noise Implementation of Surface Code in Stim: https://github.com/jasonchadwick/stim_surface_code
4. My repository: https://github.com/tomibruno/eecs498

8. References

[1] Ravi, G.S., Baker, J.M., Fayyazi, A., Lin, S.F., Javadi-Abhari, A., Pedram, M., Chong, F.T. (2022). Better Than Worst-Case Decoding for Quantum Error Correction. arXiv preprint arXiv:2208.08547.

[2] Fowler, A.G., Wang, D.S., Hollenberg, L.C.L. (2010). Surface code quantum error correction incorporating accurate error propagation. arXiv preprint arXiv:1004.0255.

[3] Pesah, A. (2023, May 13). An interactive introduction to the surface code. https://arthurpesah.me/blog/2023-05-13-surface-code/

[4] Gidney, C. (2023, May 31). *What's hook error in surface code?* Quantum Computing Stack Exchange. https://quantumcomputing.stackexchange.com/questions/32799/whats-hook-error-in-surface-code

[5] IBM. *Monte Carlo Simulation* IBM Think Blog. [Online]. https://www.ibm.com/think/topics/monte-carlo-simulation