An Introduction and Review on Recent Developments

Group 28

Abstract

Today, Quantum computers have high error rates compared to the classical computer. In order to make quantum computers useful, error rates have to be as low as 1 in a trillion. While a typical transistor in a microprocessor can run for about a billion years at a billion operations per second, without ever suffering a hardware fault due to any form of interference. A huge improvement in performance is needed, since the typical quantum bits become randomized in about one one-thousandth of a second. Classical error correction employ redundacy, however, it is impossible for quantum code due to no-cloning theorem. In this review, we will summarize one of the promising method – surface code. Besides, the novel improvement of the algorithm and decoder will also be provided.

Keywords: Quantum Computation, Quantum Error Correction, Surface Code, Stabilizer codes

1.0 Introduction

Quantum computing is a rapidly developing field, and its future is full of exciting possibilities. With breakthroughs from various promising teams, quantum computers are fast approaching the point where they can start to benefit our society. However, in order to fully explore the potential of quantum computers, complex design flows have to be carefully designed and employed to boost the accuracy and efficiency of quantum computing development. Among the different methodologies, error correction is a critical component in ensuring the reliable operation of quantum computers. One of the most prominent error correction schemes is the surface code, which has garnered significant attention due to its robustness and scalability. Surface codes, a class of topological quantum codes, play a vital role in protecting quantum information from decoherence and operational errors. The surface code utilizes a two-dimensional lattice of qubits with stabilizers to detect and correct errors. Recent advancements have focused on improving the efficiency and performance of surface code decoders, stabilizers, and associated algorithms, which are pivotal for practical quantum computing. Generally, implementing and improving surface codes is a complex task. Unlike classical error correction, quantum error correction deals with qubits that exist in superpositions, making the detection and correction of errors more challenging. The design and optimization of decoders and stabilizers, which are responsible for identifying and rectifying errors, are crucial for enhancing the fidelity of quantum computations.

to be modified In Section II, we provide a comprehensive review of the surface code, including its foundational principles and its importance in quantum error correction. In Section III, we delve into recent improvements in surface code decoders, examining various approaches to enhance their accuracy and efficiency. Section IV discusses advancements in stabilizer codes and their role in the implementation of the surface code. Finally, in Section V, we explore algorithmic developments that leverage the surface code, highlighting their potential impact on the future of quantum computing.

2.0 Review of the surface code

2.1 From classical to quantum error correction

2.1.1 Classical error correction

The basic principle behind classical error correction is to increase the bits used to encode information. The exact method to introduce redundancy and retrieve information is known as a correction code. A simple example is the three-bit repetition code: the bit 0 is encoded into three bits as 000, and the bit 1 is encoded into 111, the encoded bit strings are referred to as codewords.

If the message is subject to a single-bit error, we can still retrieve the correct information via a majority vote. However, it is easy to see that this code is vulnerable to two or more bit-flip errors.

The distance d of a code is the minimum number of errors to transform a codeword into another, and its relationship to the number of errors the code can correct t is given by $d = 2t + 1$. Error correction codes are described in terms of the $[n, k, d]$ notation, where n is the total number of bits per codeword, k is the number of encoded bits and d is the code distance.

2.1.2 Classical bits to qubits

Qubits have some distinct characteristics different from classical bits. Its values can be visualized as a point on a sphere, namely the Bloch sphere, and can take on infinite numbers of values.

It may seem like qubits are subject to infinite numbers of errors, however, we can decompose any error by expanding it in terms of the Pauli basis.

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$U(\delta\theta, \delta\phi) |\psi\rangle = \alpha_I \mathbb{1} |\psi\rangle + \alpha_X X |\psi\rangle + \alpha_Y Y |\psi\rangle + \alpha_Z Z |\psi\rangle$$

The error correction process involves performing projective measurements that cause the above superposition to collapse to a subset of its terms, and by checking abnormality in parties, we can find a suitable correction process to restore the original quantum information.

2.1.3 Challenges in quantum error correction

Qubits states are confined to the no-cloning theorem, thus making it impossible for us to add redundancy by simply copying states and using the tensor product. The second complication in quantum coding arises from the fact that qubits are susceptible to both bit-flips (X-errors) and phase-flips (Z-errors), and we need to design codes to detect both errors simultaneously. The last challenge in quantum error correction is the fact that quantum states collapse after measurement, thus the error correction procedure must be carefully chosen so as not to cause the wave function to collapse and erase the encoded information.

2.2 Quantum redundancy and stabilizer measurements

In quantum codes, redundancy is added by expanding the Hilbert space in which the qubits are encoded. Taking the two-qubit code as an example, the encoding process is shown below, noting that this process does not disobey the no-cloning theorem:

$$\begin{aligned} |\psi\rangle = \alpha |0\rangle + \beta |1\rangle & \xrightarrow{\text{two-qubit encoder}} |\psi\rangle_L \\ &= \alpha |00\rangle + \beta |11\rangle = \alpha |0\rangle_L + \beta |1\rangle_L , \end{aligned}$$

After encoding, the logical qubit occupies a four-dimensional Hilbert space

$$|\psi\rangle_L \in \mathcal{H}_4 = \text{span}\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$$

More specifically, the logical qubits are defined within a two-dimensional subspace called the codespace, and bit-flip errors rotate the state into another subspace called the error subspace.

$$|\psi\rangle_L \in C = \text{span}\{|00\rangle, |11\rangle\} \subset \mathcal{H}_4$$

$$X_1 |\psi\rangle_L \in \mathcal{F} \subset \mathcal{H}_4$$

To differentiate between the codespace and the error space, a projective measurement of Z_1Z_2 is performed. The Z_1Z_2 operator yields a (+1) eigenvalue when applied to the logical state, and is said to stabilize the logical qubit. On the other hand, The Z_1Z_2 operator yields a (-1) eigenvalue when applied to errored states. Note that the coefficients stay undisturbed during this parity check.

$$Z_1Z_2 |\psi\rangle_L = Z_1Z_2(\alpha |00\rangle + \beta |11\rangle) = (+1) |\psi\rangle_L$$

The figure 1 shows the circuit implementation of the two-qubit code. Following the error stage, an ancilla qubit $|0\rangle_A$ is introduced to measure the Z_1Z_2 stabilizer. The ancilla's outcome is referred to as a syndrome, and we can construct a syndrome table that shows the outcome of the syndrome corresponding to possible errors.

Table 1: The syndrome table for the two-qubit code.

Error	**Syndrome, S**
I_1I_2	0
X_1I_2	1
I_1X_2	1
X_1X_2	0

Note: The syndrome S is a bit string where each bit corresponds to the outcome of a stabiliser measurement.

In this correction scheme, we can conclude that the logical qubit is subject to a bit-flip error if we measure the ancilla and get the syndrome 1. It requires simultaneous bit-flip errors on both qubits such that we do not notice any error, thus suppressing the error rate. Note that we could not determine which qubit had been subject to the bit-flip mistake even if we acknowledge the existence of it, thus more sophisticated codes must be designed to fulfill the purpose of error detection and correction.

2.3 Stabilizer codes and operators

This section briefly discusses the operation of the general $[[n, k, d]]$ stabilizer code, discussing basic concepts sufficient for readers to understand key concepts of recent papers we have surveyed.

The figure 2 shows the basic structure of an $[[n, k, d]]$ stabilizer code. A register of k data qubits is entangled with $m = n-k$ redundancy qubits to create a logical qubit, errors can then be detected by performing m stabilizer measurements Π.

Constructing a good code involves finding stabilizers that anti-commute with the errors to be detected so that the presence of errors will be detected via syndrome.

The result of the m stabilizer measurements gives us an m-bit syndrome, we then deduce the best recovery operation to restore the logical state to the codespace using various decoding algorithms, since for large numbers of qubits it is impossible to enumerate errors for all possible syndromes.

2.3.1 Properties of the stabilizers

The stabilizers Π must satisfy some properties. Firstly, they must stabilize all logical states, which means when the stabilizers act on any logical state, they must return the eigenvalue 1. Secondly, all the stabilizers of a code must commute with one another, this is necessary so that the stabilizers can be measured simultaneously.

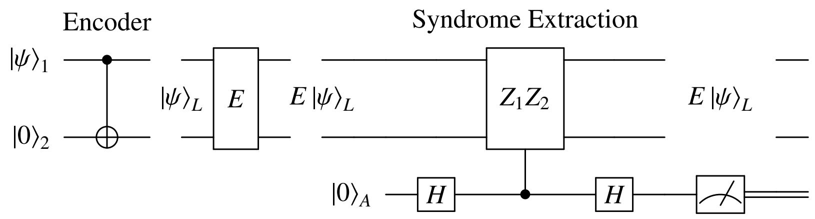

Figure 1

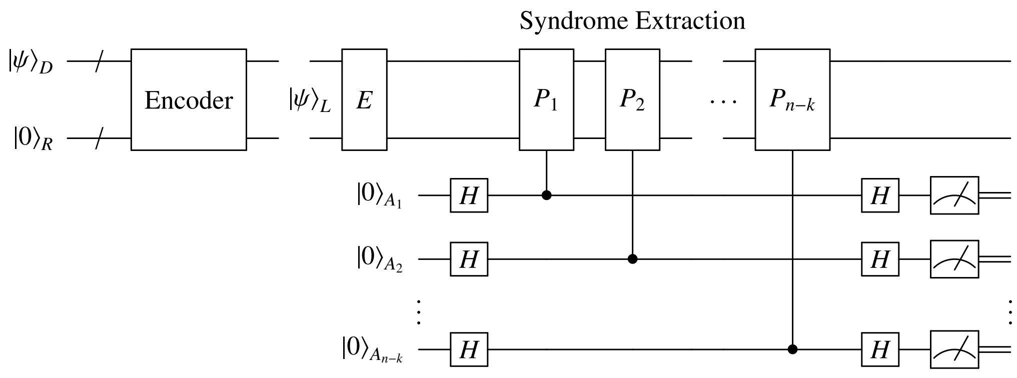

Figure 2

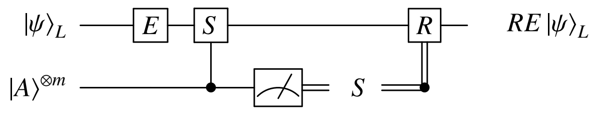

Figure 3

2.3.2 Logical operators

An $[[n, k, d]]$ stabilizer code has $2k$ logical Pauli operators that allow for logical states to be modified without having to decode and then re-encode. Each logical operator must commute with all the code stabilizers.

Finding sets of stabilizers and logical operators is not trivial, and needs to be found with effort.

2.3.3 Quantum error correction with stabilizer codes

The figure 3 shows the general error correction procedure for a single cycle of an $[[n, k, d \geq 3]]$ stabilizer code. After the logical qubit is subject to an error, stabilizer measurements are performed to produce an m-bit syndrome $\mathcal{S}$. The next step, referred to as decoding, involves processing the syndrome to determine the best unitary operation $\mathcal{R}$ to return the logical state to the codespace. The decoding step is successful if the combined action of $\mathcal{R}E$ on the code state is as follows

$$\mathcal{R}E|\psi\rangle_L = (+1)|\psi\rangle_L$$

The decoding step fails if the recovery operation maps the code state as follows

$$\mathcal{R}E|\psi\rangle_L = L|\psi\rangle_L$$

The recovery process rotates the qubit into the codespace but leads to a change in the encoded information.

2.4 Surface code

Commuting sets of stabilizers enable the ability to detect different errors simultaneously without disturbing information. Finding such sets is non-trivial, and special codes have to be designed.

The general design principle behind the surface code is that the code is built up by patching repeated elements. This approach ensures that the surface code can be straightforwardly scaled whilst ensuring stabilizer commutativity. One advantage of the surface code is that it requires only nearest-neighbor interactions, as high-fidelity long-range interactions are difficult to maintain.

2.4.1 The surface code four-cycle

The basic element in the surface code is shown in the figure 4 below, squares represent ancilla qubits, while circles represent data qubits. Black edges represent controlled X gates, each controlled on an ancilla qubit A and acting on a data qubit D. Likewise, the dashed edges represent controlled-Z operations, each controlled by an ancilla qubit and acting on a data qubit. For example, ancilla A_1 measures the stabilizer XD_1XD_2. The surface code four-cycle is not useful itself as it encodes

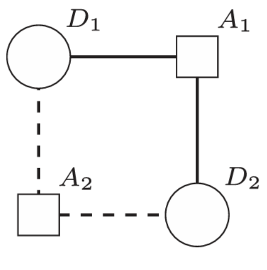

Figure 4: Pictorial representation of the four cycle. The code qubits, D_1 and D_2, are represented by the circular nodes. The ancilla qubits, A_1 and A_2, are represented by the square nodes. The black and dashed edges depict controlled-X and controlled-Z operations controlled on the ancilla qubits and acting on the code qubits.

$k = n - m = 0$ qubits, but working detection and correction codes can be formed by tiling together multiple four-cycles to form square lattices.

2.4.2 The [[5, 1, 2]] surface code

The figure 5 below shows the five-qubit surface code formed by tiling four four-cycles in a square. The stabilizers can be read off to give

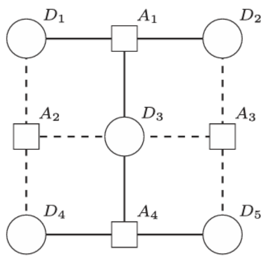

Figure 5: The $[[5, 1, 2]]$ surface code is formed by tiling together four four-cycles in a square lattice.

The logical operators of the surface code can be defined as chains of Pauli operators along the edges of the boundaries of the surface code. For example, XD_1XD_4 and ZD_1ZD_2 are both valid operators.

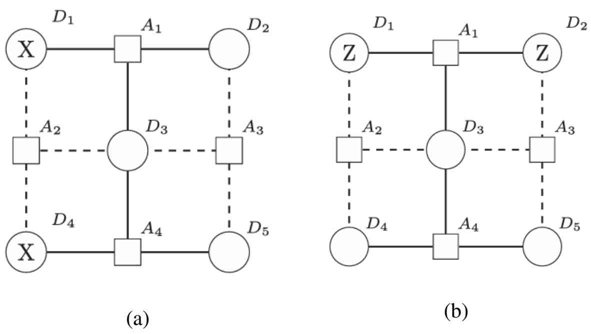

Figure 6: The logical operators of a surface code can be defined as chains of Pauli operations that act along the boundaries of the lattice.

From the above we see that the minimum weight of the logical operators is 2, meaning the $[[5, 1, 2]]$ code is a detection code with $d = 2$.

2.4.3 Scaling

The distance of a surface code can be increased simply by scaling the size of the lattice. The figure 7 shown below is the $[[13, 1, 3]]$ surface code, stabilizers and operators can be read off just by inspection.

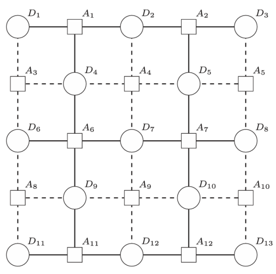

Figure 7: A distance-three surface code with parameters $[[13, 1, 3]]$.

2.5 Practical considerations

2.5.1 Decoding algorithms

Given a code syndrome, the role of the decoder is to find the best recovery operation to restore the quantum information into the codespace. For small code examples, it is possible to compute the lookup table for all combination of errors. However, as the code size increases, the number of possible syndromes grows exponentially, and it becomes impractical to use such decoding strategy.

In place of lookup tables, decoding algorithms are developed to decode syndromes. For surface codes, one technique known as minimum weight perfect matching (MWPM) can be used for decoding, which works by identifying error chains between positive syndrome measurements.

For these approximate inference techniques, the logical error rate of a quantum error correction code will depend heavily on the decoder used, and the efficiency of the decoder is also a topic that researchers opt to optimize.

2.5.2 Experimental implementation

The experimental implementation of the surface code represents a significant milestone in quantum computing, aiming to achieve fault-tolerant logical qubits. Researchers at institutions like Google, IBM, and TU Delft are at the forefront, developing superconducting devices to realize this goal. The surface code known for its high threshold for error rates under realistic noise conditions, demands sophisticated hardware capable of maintaining low error rates.

2.5.3 Fault tolerant

In the part where we introduce basic working principles of the surface code, we have assumed that errors happen in certain parts of the circuit. In realistic environments, however, this is not the case as errors could also appear at two-qubit gates or stabiliser measurements

A fault-tolerant code is a code that could handle errors up to the code distance occurring at any location in the circuit. This involves modifying quantum circuits, which increases overhead by requiring additional ancilla qubits.

3.0 Decoder

As we have mentioned in the section 2, decoders are used for performing inference of the error and then after the decoder makes a guess of the channel error, the recovery operation can be implemented. However, making the decoder to be more accuracy usually leads to increasing its computational complexity, which will make it slower in making guesses. The trade-off between the accuracy and complexity is vital for quantum error correction. In section 2, we have discussed the detail of MWPM. Here we provide a review about the paper (Optimal adaptation of surface-code decoders to local noise(arXiv:2403.08706v1)) which confirm that decoding algorithms to the specific noise characteristics of a quantum device create more efficient and effective quantum error correction.

3.1 Tensor-network decoder

The biggest different from MWPM is that MWPM only takes Pauli noise into account, and it assume uncorrelation between X error and Z error. But, the two-dimensional tensor-network method is provided with the noise map. It can be optimally adapted to arbitrary local noise, including non-Pauli noise, and can also handle correlated noise. The decoding algorithm involves constructing a square-lattice tensor network using information from the syndrome and the suspected noise map, then contracting the network using the time-evolving block decimation (TEBD) algorithm.

3.1.1 Tensor Network States

A tensor network can be visualized as a graph where each vertex represents a tensor, a multi-index array of complex numbers. The number of edges incident to a vertex corresponds to the number of tensor indices, with the dimension of an index called the bond dimension. When two vertices are connected by an edge, their corresponding indices are summed over (contracted), similar to matrix multiplication. A vertex can have incident edges not connected to another vertex, called physical indices, which are not contracted.

3.1.2 Surface Code Wave Function

The surface code wave function for any encoded state can be described as a projected entangled pair state (PEPS), represented on a square lattice where each tensor corresponds to a physical qubit, which is shown in Figure 8(a). The PEPS structure implies that the density operator can be expressed as a projected entangled pair operator (PEPO).

3.1.3 Logical Channel Calculation

For noiseless syndrome extraction, the logical channel $R_s \circ N$ can be calculated by computing all 16 elements of the Pauli transfer matrix: $C_{ij} = \mathrm{Tr}(L_i(R_s \circ N(L_j)))$ where L_i , L_j are logical Pauli operators, N is noise map. The tensor-network structure preserves the PEPS form during the application of local noise N , recovery R_s, and logical Pauli L_i , allowing to be expressed as a multi-layer PEPS. Evaluating the trace involves tracing out pairs of physical indices, resulting in a tensor network with no physical indices. This tensor network can be compressed into a single layer by contracting along the time dimension. The resulting square-lattice tensor network can then be contracted using methods like treating the left-hand column as a matrix product state (MPS), evolving it by applying columns sequentially, and using singular value decomposition to control bond dimension growth. Alternatively, the square lattice tensor network can be contracted exactly by merging tensors in the left column into a single tensor, then contracting tensors from the remaining network column by column. The process is shown in the Figure 8.

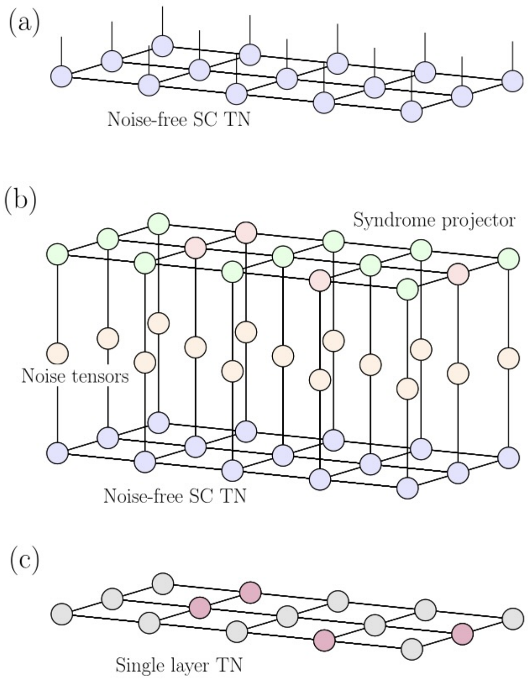

Figure 8: (a) A square-lattice PEPS depicting a surface code state, with each tensor having one physical index. (b) A tensor network representing an entry Cij of the Pauli transfer matrix of the logical channel for the surface code under a local noise map with noiseless syndrome extraction. There are three layers corresponding to the noiseless surface code state, local noise, and a syndrome projector, with physical indices traced out. (c) The same tensor network from (b) expressed as a single-layer tensor network which is obtained by vertically contracting the three layers.

3.2 Recognizing critical noise parameters using simulation-based decoding

The paper provides how to identify the most crucial noise in the noise map. If a surface code with a local physical noise map N. We can find the optimal decoder that achieves optimal performance for this noise map Then we adjust the miscalibrated decoder, with different parameter N' . If one single noise parameter is different from the optimal parameter, but the result is identical to the optimal one, then we can indicate that the noise is not crucial. The paper have done some adaption about three primary noise features: coherence, inhomogeneity, and bias.

3.2.1 Coherence

Quantum noise involves systematic, non-random errors, such as unitary over-rotations due to imperfect gate control. Theoretical analyses of quantum error correcting codes often assume a Pauli noise model, where errors are characterized by random Pauli errors drawn from a probability distribution. This simplifies calculations, especially for stabilizer codes like the surface code, as Pauli noise can be efficiently simulated within the stabilizer formalism. However, physical systems rarely adhere exactly to this assumption. Non-Pauli noise types, such as amplitude damping and systematic over rotations due to imperfect gate control, are common but often overlooked. In order to differentiate from Pauli noise, the author refers to non-Pauli noise as coherent noise. Coherent noise introduces off-diagonal elements in the χ matrix representation of quantum channels, indicating interactions between different Pauli errors. Previous studies have shown that the surface code's performance can be significantly impacted by coherent noise, sometimes even more so than by Pauli noise. The paper investigates two noise models: single-qubit unitary rotations and systematic over-rotations in CNOT gates. The author compares the performance of different decoders, including an optimal decoder, a decoder considering only incoherent noise components (Pauli' decoder), and a standard minimum-weight perfect matching (MWPM) decoder applied separately to X and Z syndrome data. The paper reveals that adapting the decoder to consider the full noise map, including coherent components, can lead to performance improvements over decoders optimized solely for incoherent noise. However, the magnitude of this improvement is relatively small.

3.2.2 Inhomogeneity

Noise in many quantum computing system is spatially inhomogeneous, that is, it varies across different qubits in the device. The paper take noise model of amplitude-phase damping into account. As the **equation (6)(7)** shows, γ represents damping probability and λ scattering probabilities. We can assign it as relaxation time T_1 and the dephazing time T_2 by $e^{-t/T_2} = \sqrt{(1-\gamma)(1-\lambda)}$ and $e^{-t/T_1} = 1-\gamma$. In experiments, T_1 and T_2 times can vary significantly over different qubits in a device. The results of our simulations with different decoders are shown in Figure 9. The first decoder is optimal, which is perfectly fit to the T_1 and T_2. The second 'uniform' one only have the average of T_1 and T_2 of each qubit. The third decoder 'ratio' only has information about ratio of T_1 and T_2 on each qubit. The fourth decoder is simply standard MWPM using a uniformly weighted syndrome graph. The result are clear that the optimal one which equipped with all the information about damping noise outperforms the standard one. The most interesting part is that even just have the ratio of T_1 and T_2 also improved significantly from the standard one.

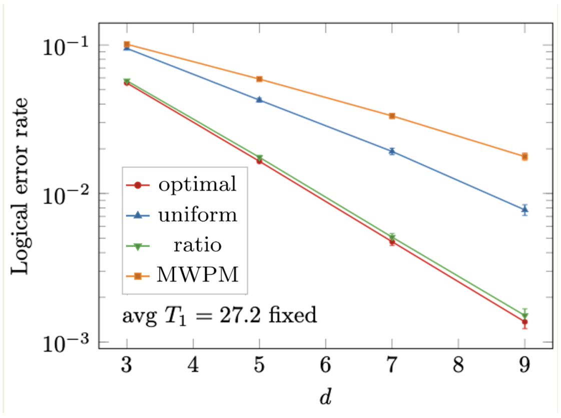

Figure 9

$$N_{APD}(\rho) = \begin{pmatrix} \rho_{00} + \gamma\rho_{11} & \rho_{01}\sqrt{(1-\gamma)(1-\lambda)} \\ \rho_{10}\sqrt{(1-\gamma)(1-\lambda)} & \rho_{11}(1-\gamma) \end{pmatrix}$$
$$= \begin{pmatrix} 1-\rho_{11}e^{-t/T_1} & \rho_{01}e^{-t/T_2} \\ \rho_{01}^{*}e^{-t/T_2} & \rho_{11}e^{-t/T_1} \end{pmatrix}$$

3.3 Bias

The paper defines a Pauli noise model as biased when one Pauli error occurs with significantly higher probability than others. As the paragraph aforementioned, we designate either X, Y, or Z as the dominant error, and the surface code's performance heavily relies on the local basis used for defining checks. The author focus is on single-qubit Y-biased Pauli noise, where $p_X = p_Z$, $p := p_X + p_Y + p_Z$, and $\eta := \frac{p_Y}{p_X+p_Z}$. The simulation results, depicted in Figure 10, illustrate the effects of depolarizing (unbiased) noise corresponding to $\eta = 0.5$ and biased noise with $\eta = 100$. For depolarizing noise, as shown in Figure 10(a), the miscalibrated decoder achieves nearly same logical error rates compared to the optimal decoder, with the threshold barely altering. On the other hand, under biased noise with fixed $\eta = 100$, as shown in Figure 10(b), there is a significant difference observed between the two decoders. The logical error rate for the largest system size with the miscalibrated decoder. These findings suggest that sensitivity to decoder miscalibration escalates with noise bias. Notably, the decoder needs a number of information about the bias η to perform optimally

By focusing on coherence, inhomogeneity, and bias, and understanding their impact on decoder performance, the paper provides a framework for optimizing surface-code decoders by leveraging detailed noise information. This targeted approach not only aids in developing better decoders but also informs the design of noise characterization techniques, ensuring that future quantum computing systems can maintain high levels of fault tolerance with reduced overhead.

4.0 Stabilizer

Stabilizers are used to detect errors in the quantum state. In the context of surface code, stabilizers are operators that mea-

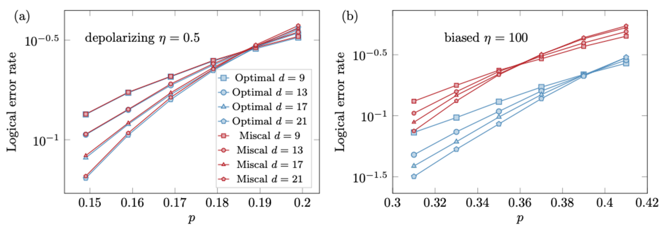

Figure 10: The effect of noise bias on surface-code decoding. 'Optimal' means that the noise model input to the decoder is exactly the physical noise model, and 'Miscal' means that the average infidelity in the decoder noise model is mischaracterized pdec = p/2. (a) physical noise model is depolarizing noise, while in (b) it is biased noise with $\eta = 100$

sure a group of qubits. When an error occurs, it changes the measurement outcomes of these stabilizers, indicating the presence and type of error. Nonetheless, as quantum computing scales, the defects in qubits and couplers due to fabrication errors become inevitable. These can significantly disrupt the surface codes error-correcting capabilities, which are essential for reliable quantum computation. **The paper (Low-Overhead Defect-Adaptive Surface Code with Bandage-Like Super-Stabilizers)** propose an automated adapter for surface codes that can handle defective lattices. This adapter includes three main subroutines: boundary deformation, internal defect disabling, and stabilizer patching. In this sector, we will briefly go through all the steps.

4.1 Constructing stabilizer steps

4.1.1 Boundary deformation

We first begin the boundary deformation process by addressing defects along the boundary, focusing on removing unsafe boundary data qubits and redundant syndrome qubits. A boundary data qubit is considered safe if it meets three conditions: the qubit itself is defect-free, its surrounding frontier—including neighboring syndrome qubits and couplers—is also defect-free, and this frontier aligns with the boundary type, as example in Figure 11(e). We can simply use a Breadth-First Search (BFS) algorithm to evaluate the safety of each boundary data qubit. If one unsafe data qubit is found, the surrounding redundant syndrome qubits—those that are defective, have no undisabled data qubit neighbors, or are of different types from the boundary—are disabled. Following these guidelines, the surface code in Figure 11(a), will turn into Figure 11(b). The BFS algorithm continues to reevaluate boundary data qubits, checking any new unsafe ones and redundant syndrome qubits, until no new unsafe boundary data qubits are detected. This iterative process results in a defect-free boundary.

4.1.2 Internal defect disabling

The process of internal defect disabling involves addressing defects within the lattice. Internal defects are classified into three types: data qubits, syndrome qubits, and couplers. According to the rules, for data qubit and coupler defects, we disable the corresponding data qubits, while for syndrome qubit defects, we disable the syndrome qubits and their neighboring data qubits, which shown in Figure **??**. This is because internal data qubits need two X and two Z syndrome qubits to detect Z and X errors. The defects must be addressed in a specific order—first defective syndrome qubits, then defective data qubits, and finally defective couplers—in order to ensure each rule is applied only once and to prevent conflicts. Additionally, we disable weight-0 syndrome qubits (a weight-n syndrome qubit has n undisabled data qubit neighbors) resulting from these actions. Internal defects may cluster, particularly at high defect rates, leading to situations like weight-1 syndrome qubits and bridge syndrome qubits, where a syndrome qubit connects to two active data qubits along the same diagonal. As shown in Figure 11(h). Disabling these types of syndrome qubits, doing so might require reapplying internal defect rules, potentially disrupting the boundary shape and causing an avalanche effect where many qubits are disabled. To avoid this, the proposed method do not disable internal weight-1 and bridge syndrome qubits. Instead, the author use a bandage-like super-stabilizer strategy to reduce the number of disabled qubits, minimize super-stabilizer weight, and prevent an avalanche effect.

4.1.3 Stabilizer patching

The proposed bandage-like super-stabilizers connect the same type of gauge syndrome qubits through disabled qubits, covering all internal disabled qubits. For unclustered internal defect qubits, these super-stabilizers are similarly to traditional super-stabilizers, which shown in Figure 11(g). While for clustered defect qubits, they stretch across weight-1 and bridge syndrome qubits, preserving more data qubits and maintaining integrity, as shown in Figure 11(g). And then logical operators (X and Z) are placed on paths containing opposing types of syndrome qubits to avoid intersecting super-stabilizers and introducing gauge qubits. Eventually, the proposed method adapts the defect lattice depicted in Figure 11(a) into surface code shown in Figure 11(d)

4.2 Stabilizer measurement circuit

Due to their anti-commuting gauge operators, constructing the stabilizer measurement circuits for adapted devices can't be directly measured like regular single-syndrome stabilizer. In the method, multiple bandage-like super-stabilizers may combine to form a group of super-stabilizers such as that in Figure 12, where a group with two X and two Z bandage-like super-stabilizers. It is essential to ensure that X and Z super-stabilizers in the same group are not measured in the same cycle, while the same type within a group is measured simultaneously. To enhance error correction capability, the method repeat the measurement of the same type of gauge operator for several consecutive cycles to gather information about each operator's value. The number of consecutive measurement cycles is referred to as the shell size, as shown in Figure 12. It is necessary to determine the appropriate shell size for each stabilizer group while considering experimental constraints. There are two strategies for determining the shell size: the global strategy

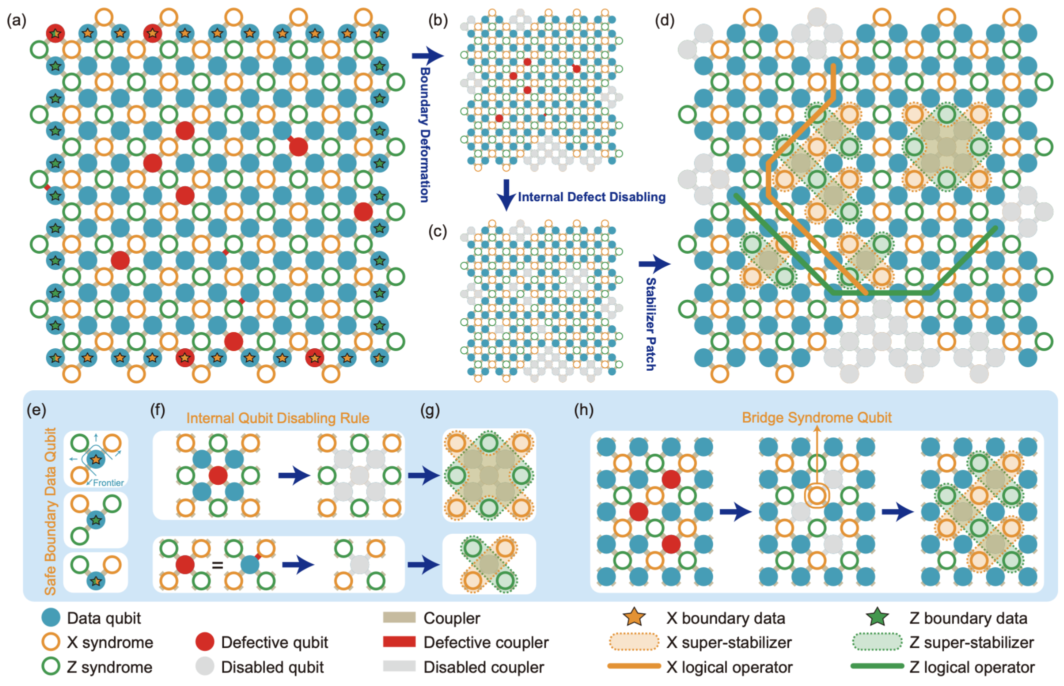

Figure 11: The Construction Steps for the Defect-Adaptive Surface Code.(a) An instance of a defective surface code lattice,the defective qubits and couplers in red, and boundary qubits are marked with star symbols. (b) code lattice after boundary deformation (c) code lattice after internal defect disabling. (d) code lattice after stabilizer patch, respectively. (e) The depiction for boundary data qubits and their frontiers, including X, Z, and corner boundary data qubits from top to bottom, along with their frontiers, encompassing couplers and syndrome qubits around data qubits. (f) Depicts the rules for internal defect disabling, showcasing the disabled qubits rules for defect syndrome qubits, data qubits, and couplers from top to bottom. (g) practice traditional super-stabilizers if internal defect qubits are not clustered. (h) in clustered situations, these super-stabilizers can stretch across weight-1 and bridge syndrome qubits. The middle of (h) highlights a bridge syndrome qubit for illustration purposes.

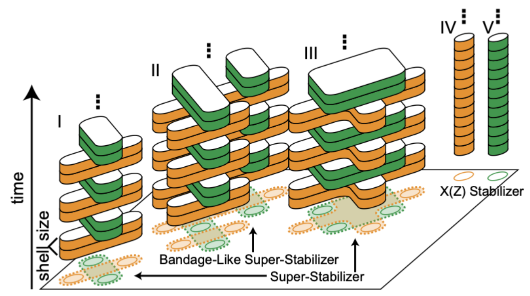

Figure 12

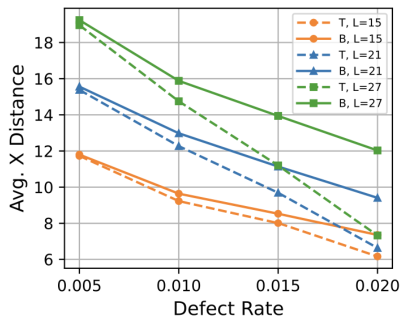

Figure 13: Average X Distance vs Defect Rate

applies the same shell size to all stabilizer groups, while the local strategy assigns each group its own shell size. The choice of shell size depends on the processor and physical system characteristics.

4.3 Conclusion

By following these steps, the method ensures a defect-adaptive surface code with enhanced efficiency, and also avoid overhead. The paper compared the traditional stabilizer with their proposed method. Naturally, code distance will decrease as defect rates gets higher. In contrast to the traditional stabilizer, the bandage stabilizer maintains a superior code distance, and this advantage grows as defect rates increase, similar to the above specific case. The result is demonstrated in Figure 13 and Figure 14. Besides, the proposed method also preserves code distance by disabling fewer qubits. The illustration is in Figure 15 which shows the average percentage of disabled qubits after adaptation. Despite that disabled qubits increase with defect rates, the proposed method demonstrates a slower increase, which indicate better qubit preservation compared to the traditional method.

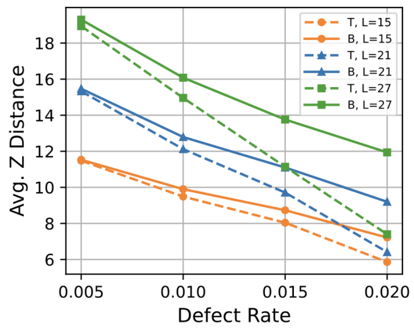

Figure 14: Average Z Distance vs Defect Rate

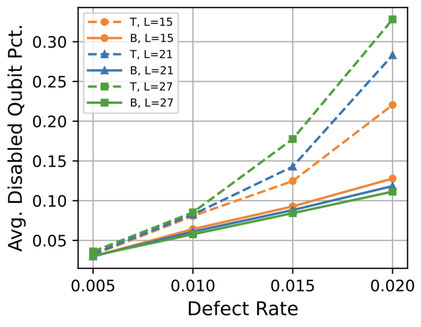

Figure 15: Average Z Distance vs Defect Rate

5.0 Practical implementation of the surface code

5.1 Suppressing quantum errors by scaling a surface code logical qubit

Scaling of the surface code is essential to building large-scale quantum computers. Google Quantum AI has demonstrated successful implementation of the scaling of surface code, showing that increasing the number of qubits can indeed improve the performance of a logical qubit, suppressing the error rate while also dealing with additional problems associated with scaling.

The research team of Google Quantum AI implemented a distance-5 surface code using 72 qubits. By comparing it to a smaller distance-3 code, they observed a reduction in logical error rates per cycle, from 3.028% to 2.914%.

This work shows that quantum error correction improves with scaling, essential for developing large quantum systems for applications in cryptography, optimization, and simulations beyond classical capabilities. Using a 72-qubit device, the team showed that larger codes can outperform smaller ones, highlighting the potential of scaling for better quantum error correction. This is a key step towards making quantum computing viable for complex computations.

5.2 Realizing repeated quantum error correction in a distance-three surface code

Successful implementation of the surface code is a milestone in quantum error correction, showing that practical realization

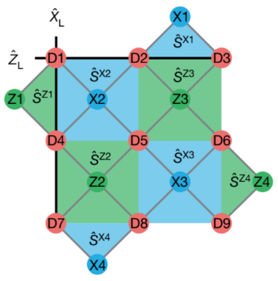

Figure 16

is possible, we are one step closer to making a large-scale quantum computer. Using 17 physical qubits in a superconducting circuit, researchers have encoded quantum information in a distance-three logical qubit. In a correction cycle of 1.1 (μs) that can correct both bit-flip and phase-flip errors, they have successfully preserved the cardinal states of the qubit. By repeatedly correcting errors, a low error rate of 3% per cycle shows that the surface code does suppress the error probability as theory suggests. This breakthrough brings us closer to large quantum computing systems that can perform computation out of reach of the capacities of modern classical computers.

This achievement demonstrates the robustness of the surface code in a practical setup, supporting its potential for fault-tolerant quantum computing. By achieving low error rates and preserving qubit states, the experiment validates theoretical predictions and advances the development of scalable quantum computers.

5.3 Repeated quantum error detection in a surface code

The paper discusses the implementation of a minimal surface code for QEC using superconducting qubits. The surface code is a promising architecture for fault-tolerant quantum computing due to its high error threshold. In this study, the authors use seven superconducting qubits—four data qubits and three ancilla qubits—to repeatedly detect and correct any single error. This is a crucial step towards practical QEC, which is necessary for the realization of reliable and scalable quantum computers.

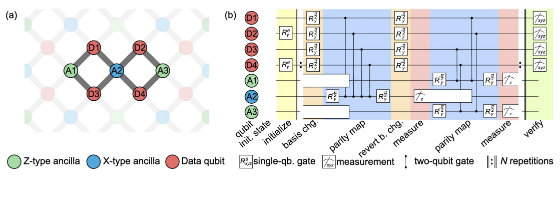

Figure 17

Numerical simulations, accounting for factors like finite qubit lifetimes and coherence times, as well as residual ZZ coupling and readout errors, show good agreement with the experimental data. The simulated logical decay times (44.2 (μs) for Z_L and 59.6 (μs) for X_L) are within the experimental error bars.

Table 2: Lifetime/Coherence Time Comparison

Error	**Lifetime (μs)**	**Coherence Time (μs)**
Logical	62.5 ± 9.4	72.5 ± 32.9
Physical	16.8	21.5

Note: For both operators, the logical error probabilities per stabilizer measurement cycle are lower than the physical error rates of the qubits involved.

5.4 Low-distance surface codes under realistic quantum noise

The paper investigates the practical implementation of distance-3 surface codes, which are a key component for scalable quantum computing. These surface codes are essential for achieving fault-tolerant quantum computation by correcting errors in quantum states. The authors focus on developing a circuit and a decoder tailored for the near-term experimental realization of these codes. They simulate the performance of the surface code under various realistic quantum noise models, such as amplitude and phase damping, and compare the results to those obtained using a Pauli-twirl approximation, finding that the approximation tends to provide a pessimistic threshold estimate.

As the figure 18 below, the study delves into three specific types of surface code layouts: the standard 25-qubit layout, and two optimized layouts with 17 and 13 qubits, respectively. Each layout has distinct configurations and resource requirements, with the 17- and 13-qubit versions designed to reduce the number of physical qubits while maintaining error correction capabilities. The paper also details the construction of a fast decoder that operates in limited-memory, low-temperature environments, crucial for practical deployment.

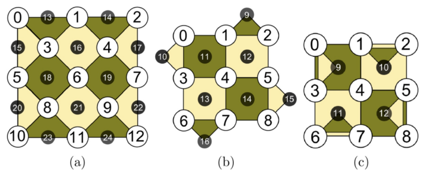

Figure 18: (a) 25,(b) 17, and (c) 13 qubits. White circles represent data qubits, and black circles represent syndrome qubits. Dark square and triangular patches represent X stabilizers; light patches represent Z stabilizers. The layered patches on surface-13 indicate use of the syndrome qubit first to measure a four-qubit stabilizer and then to measure a two-qubit stabilizer.

In their simulations, the authors consider five types of flips that can occur in the quantum bits (qubits): bit flips, phase flips, and combinations of these errors. The simulations employ realistic noise models to determine the necessary gate and measurement speeds for achieving reliable error correction. For superconducting devices, they demonstrate that a qubit encoded in a 17-qubit surface code exhibits a significantly lower error rate compared to an unencoded qubit, provided the gate times range from 5 to 40 (ns) and T_1 times (relaxation times) are at least 1 to 2 (μs). In the context of ion trap devices, they find that gate times of 1 microsecond and T_1 times of at least 40 (ms) are adequate to observe measurable differences in error rates.

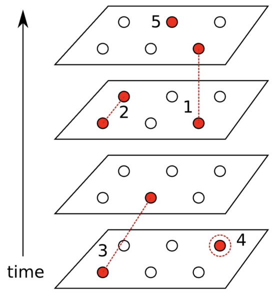

Figure 19: Lookup table decoding rules. Each circle represents a syndrome measurement. Filled (red) circles indicate "flips." Five types of flips are shown: (1) measurement error, (2,3) paired flips indicating a single-qubit error on the data qubit, (4) a single flip indicating one data-qubit error, and (5) and undetermined flip.

The paper emphasizes the importance of these findings for the experimental demonstration and verification of small surface codes acting as single-qubit quantum memory. By presenting a framework for near-term implementation, the research paves the way for significant advancements in the field of quantum error correction and the development of robust quantum computing technologies.