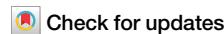




Low-overhead defect-adaptive surface code with bandage-like super-stabilizers



Zuolin Wei ^{1,2}, Tan He ^{1,2}, Yangsen Ye ^{1,2}, Dachao Wu ^{1,2}, Yiming Zhang ^{1,2}, Youwei Zhao ^{1,2}, Weiping Lin ^{1,2}, He-Liang Huang ³ , Xiaobo Zhu ^{1,2,4} & Jian-Wei Pan ^{1,2,4}

To make practical quantum algorithms work, large-scale quantum processors protected by error-correcting codes are required to resist noise and ensure reliable computational outcomes. However, a major challenge arises from defects in processor fabrication, as well as occasional losses or cosmic rays during the computing process, all of which can lead to qubit malfunctions and disrupt error-correcting codes' normal operations. In this context, we introduce an automatic adapter to implement the surface code on defective lattices. Unlike previous approaches, this adapter leverages newly proposed bandage-like super-stabilizers to save more qubits when defects are clustered, thus enhancing the code distance and reducing super-stabilizer weight. For instance, in comparison with earlier methods, with a code size of 27 and a random defect rate of 2%, the disabled qubits decrease by 1/3, and the average preserved code distance increases by 63%. This demonstrates a significant reduction in overhead when handling defects using our approach, and this advantage amplifies with increasing processor size and defect rates. Our work presents a low-overhead, automated solution to the challenge of adapting the surface code to defects, an essential step towards scaling up the construction of large-scale quantum computers for practical applications.

Overcoming the challenges posed by realistic hardware noise, quantum error correction (QEC) plays a pivotal role in protecting fragile qubits from decoherence effects, unlocking quantum computing's full potential. Among various QEC codes, the surface code¹⁻³ stands out due to its 2D nearest-neighbor coupling lattice and high error threshold, typically around 1%. While significant progress has been made in small-scale implementations of the surface code, such as increasing the code distance from two⁴⁻⁶ to three^{7,8} and then to five⁹, as well as demonstrating logical gates^{10,11}, the road to achieving large-scale, practical algorithms demands thousands of logical qubits with extremely low gate error rates, typically below 10^{-10} . This necessitates millions of physical qubits¹², a scale far beyond the capabilities of current physical devices.

As we strive to build larger quantum devices with more qubits, defects like nonfunctional qubits or failed entangling gates during fabrication are unavoidable. It is estimated that ~2% of the qubits on a transmon device would be defective with current technology¹³. Even advanced processors like *Zuchongzhi*^{7,14-17} and Google's *Sycamore*^{9,18}, with just a few dozen qubits, are susceptible to defects. Additionally,

external events such as cosmic rays impacting superconducting devices¹⁹⁻²³, or leakage and loss events in ion trap or neutral atom arrays²⁴⁻²⁶ can mimic defects. Topological codes rely on specific lattice structures to encode logical states, making them susceptible to defect errors that alter the topology and reduce code distance, necessitating an adaptive approach for error correction on defective lattices²⁷⁻³². To this end, we introduce an adapter that deforms defective lattices and identifies super-stabilizers, enabling the implementation of the surface code on defective lattices. This adapter automates the entire process, which is crucial for scalability. As chip sizes grow, manually designing the adapter based on processor defects becomes challenging, especially since we aim for programmable logical operations. Additionally, our approach introduces a new type of super-stabilizer called bandage-like super-stabilizers. These super-stabilizers ensure that our adapter operates with low overhead when dealing with defects. Compared to previous methods²⁹⁻³², our approach minimizes the number of disabled qubits caused by defects as possible, achieving higher code distances and lower-weight super-stabilizers, thus significantly reducing logical error rates. These advantages

¹Hefei National Research Center for Physical Sciences at the Microscale and School of Physical Sciences, University of Science and Technology of China, Hefei, 230026, China. ²Shanghai Research Center for Quantum Science and CAS Center for Excellence in Quantum Information and Quantum Physics, University of Science and Technology of China, Shanghai, 201315, China. ³Henan Key Laboratory of Quantum Information and Cryptography, Zhengzhou, Henan, 450000, China. ⁴Hefei National Laboratory, University of Science and Technology of China, Hefei, 230088, China. e-mail: quanhhl@ustc.edu.cn; xbzhu16@ustc.edu.cn; pan@ustc.edu.cn

highlight that our low-overhead defect-adaptive surface code approach provides a reliable and efficient path for scalable, large-scale fault-tolerant quantum computing.

Results

Defective lattice surface code adapter

Creating a surface code adapter for defective lattices demands an automated solution capable of handling diverse defect scenarios that manifest randomly across the lattice, whether along its edges or clustered closely together. Additionally, we aim to retain as many qubits as possible to mitigate the loss of error-correction capability caused by defects. In pursuit of this objective, we present a fully automated adapter customized for the surface code on a defective lattice, as depicted in Fig. 1. This adapter comprises three sequential subroutines:

Boundary deformation. We kick off by addressing defects along the boundary, removing unsafe boundary data qubits and redundant syndrome qubits. A boundary data qubit is marked as safe if it ticks off three conditions: the qubit itself and its surrounding frontier, including neighboring undisabled syndrome qubits and couplers, are defect-free, and its surrounding frontier aligns with the boundary type, as shown in Fig. 1e. This requirement stems from the surface code's need for specific syndromes to catch certain errors—an $X(Z)$ boundary data qubit, for instance, requires a $Z(X)$ syndrome to detect an $X(Z)$ error, and two $X(Z)$ syndromes to detect a $Z(X)$ error. Meanwhile, corner data qubits each require an X and a Z syndrome to catch errors in both directions.

Following these safety guidelines, we turn to a breadth-first search (BFS) algorithm to adjust the surface code lattice boundaries inward. This process, elaborated in the “boundary deformation” algorithm in the Supplementary Material, assesses the safety of every boundary data qubit. Upon detecting an unsafe data qubit, we disable it along with its surrounding redundant syndrome qubits, which may be defective, weight-0 (a weight- n syndrome qubit has n undisabled data qubit neighbors), or of different types from the boundary (e.g., Z syndrome qubits adjacent to the disabled X boundary data qubit). The BFS iteratively reassesses boundary data qubits to identify any new unsafe ones and redundant syndrome qubits until no new unsafe boundary data qubits emerge, resulting in a defect-free boundary, as depicted in Fig. 1b.

Internal defect disabling. This straightforward step involves tackling internal defects, which come in three types: data qubit defects, syndrome qubit defects, and coupler defects. We follow the rules outlined in Fig. 1f to disable these defects and their neighboring qubits. Specifically, for data qubit and coupler defects, we disable the corresponding data qubits. For syndrome qubit defects, we disable the corresponding syndrome qubits and their neighboring data qubits. The underlying reason for these rules is that internal data qubits require two X and two Z syndrome qubits to detect Z and X errors. A specific order is necessary when tackling internal defects—defective syndrome qubit first, then defective data qubit, and finally defective coupler—to ensure each rule is applied only once and prevent conflicts. Finally, we disable weight-0 syndrome qubits caused by the implementation of the above rules. As

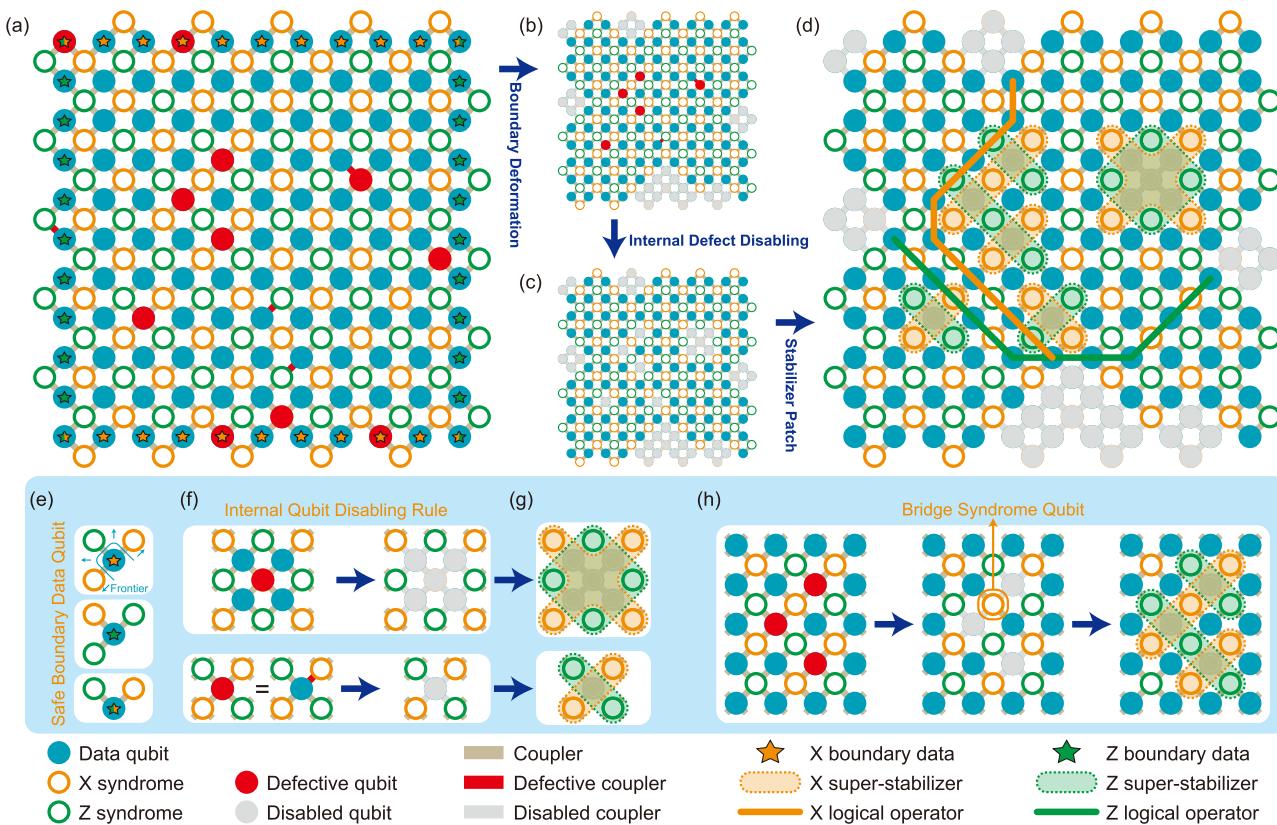


Fig. 1 | The construction steps for the defect-adaptive surface code. **a** An example of a defective surface code lattice, where defective qubits and couplers are marked in red, and boundary qubits are marked with star symbols. **b-d** Display the surface code lattice after undergoing boundary deformation, internal defect disabling, and stabilizer patch, respectively. **e** Illustrates safe 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. The rules for defect data qubits and couplers are the same. **g, h** demonstrate the rules for bandage-like super-stabilizers. In scenarios where internal defect qubits are not clustered, they behave similarly to traditional super-stabilizers, as shown in (g). However, in clustered situations, these super-stabilizers can stretch across weight-1 and bridge syndrome qubits, as illustrated in (h). Additionally, (h) highlights a bridge syndrome qubit for illustration purposes.

can be easily seen, this entire process is conducted without altering the boundary's shape.

We note that internal defects may cluster together, especially at high defect rates. This leads to two primary scenarios: weight-1 syndrome qubit, and bridge syndrome qubit (see Fig. 1h, where a syndrome qubit connects to two active data qubits along the same diagonal line. Previous research³² suggests disabling these types of syndrome qubits. However, such action may require reapplying internal defect-disabling rules, potentially disrupting the previously fixed boundary shape. In cases of high defect rates, this could trigger an avalanche effect, disabling a significant portion of qubits (refer to the Supplemental Material for an example). In our approach, we don't disable internal weight-1 and bridge syndrome qubits due to our proposed bandage-like super-stabilizer. This strategy helps reduce disabled qubits, minimizing super-stabilizer weight and preventing an avalanche effect. Additionally, retaining bridge syndrome qubits potentially maintains a greater code distance (refer to the Supplemental Material for an example).

Stabilizer patch. In this step, we utilize the proposed bandage-like super-stabilizers, which combine the same type of gauge syndrome qubits through disabled qubits, to cover all internal disabled qubits. When internal defect qubits are not clustered, they function similarly to traditional super-stabilizers^{29–32}, as depicted in Fig. 1g. However, if internal defect qubits cluster, these super-stabilizers can stretch across weight-1 and bridge syndrome qubits, maintaining the integrity of syndrome qubits and conserving more data qubits, as shown in an example in Fig. 1h (refer to Algorithm 3 of Supplementary Material for the detailed construction method for super-stabilizers). These super-stabilizers share an even number of data qubits with the opposite type of stabilizers, allowing them to commute with each other.

In the final step, we place logical operators X and Z , onto the defective lattice. These operators are positioned along paths containing opposing types of syndrome qubits to avoid intersecting super-stabilizers and introducing gauge qubits. Generally, multiple equivalent logical operators exist, and we choose the most convenient option. Note that the logical operators we aim to protect should contain no gauge operator. They may not be the shortest one and cannot be used to determine the code distance. (refer to the Fig. S12 of Supplementary Material for an example of counting the code distance). Finally, our method adapts the defect lattice depicted in Fig. 1a into the surface code shown in Fig. 1d, resulting in a greater X distance of 5 compared to the 4 achieved by the traditional method. (refer to the Supplemental Material for the performance comparison between the bandage-like method and traditional method for this defect lattice).

Building a stabilizer measurement circuit

Building stabilizer measurement circuits for adapted devices involves measuring super-stabilizers, which can't be directly measured like regular single-syndrome stabilizers because they contain anti-commuting gauge operators. We measure super-stabilizers using a common method from ref. 29, where X and Z super-stabilizers are measured in alternate cycles, and their outcomes are inferred from the gauge operators' product. We note that in our method, multiple bandage-like super-stabilizers may intertwine to form a super-stabilizer group (e.g., in Fig. 2a II, we see a group with 2 X and 2 Z bandage-like super-stabilizers). It's crucial to ensure that X and Z super-stabilizers in the same group are not measured in the same cycle, while the same type in a group are measured simultaneously.

Furthermore, to improve error correction ability, we can use the shell method outlined in ref. 30. This involves repeating the measurement of the same type of gauge operator for several consecutive cycles, allowing extracting information about each gauge operator's value. The number of consecutive measurement cycles is called the shell size, as illustrated in Fig. 2a. We must determine the appropriate shell size for each stabilizer group while ensuring it aligns with experimental constraints. Basically, there are two strategies for determining the shell size: the global strategy applies the same shell size to all stabilizer groups, while the local strategy assigns each stabilizer group its own shell size. The selection of the shell size depends

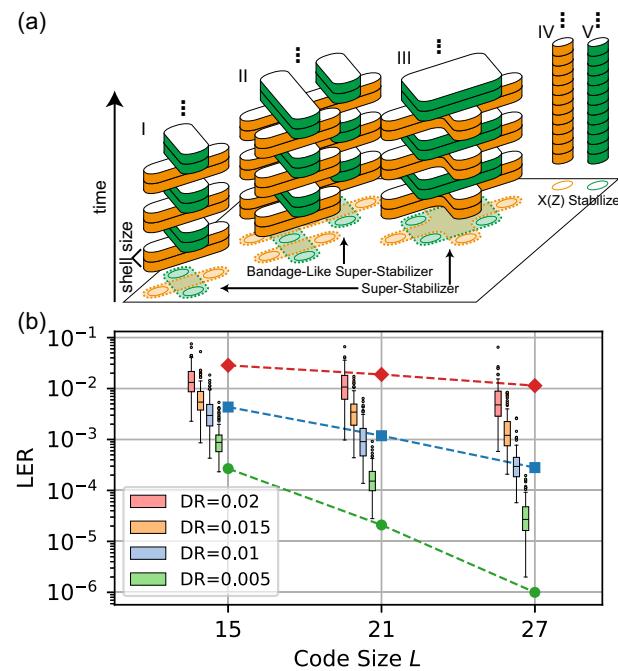


Fig. 2 | Illustration of the stabilizer measurement circuit building and related simulation results. a Space-time lattice with super-stabilizers forming shells. The columns align along the temporal direction to show the measurement of super-stabilizers in the space-time lattice. Various types of super-stabilizers are shown: I. Super-stabilizers formed by a single data qubit defect. II. Bandage-like super-stabilizers formed by the nearby defect. III. Super-stabilizers formed by a single-syndrome qubit defect. IV. X stabilizer is unaffected by defects. V. Z stabilizer unaffected by defects. The shell size indicates the consecutive measurements of the same type of super-stabilizers. For regular stabilizers, X and Z stabilizers are measured in the same cycle. However, X and Z super-stabilizers cannot be measured in the same cycle. b The logical error rate (LER) of the surface code under different code sizes L and defect rates (DR). The box plot displays the logical error rates for defect rates of 0.005, 0.01, 0.015, and 0.02 at a physical error rate of $p = 0.002$. The whiskers extend to data within 1.5 times the interquartile range (IQR) from the box. Points beyond are fliers. The outlier arises from two factors: (1) sampling defective devices at a rate of p involves independently assigning each component a defect probability of p , rather than exactly pN defects across N components, leading to variation in defect counts even at a consistent defect rate; (2) the spatial distribution of defects, such as when multiple defects align in a straight line or cluster in a specific area, can significantly impact the code distance. The green, blue, and red dashed lines represent references for a perfect surface code with physical error rates of $p = 0.002$, 0.003, and 0.004, respectively. In simulations, we generated 100 devices with randomly distributed defects for each L and defect rate.

on the characteristics of the processor and physical system, as discussed in the Supplemental Material. For simplicity, in the following numerical simulations, we use the global shell method.

Once the measurement circuit is set up, we can numerically simulate and test the performance of the defective lattice surface code adapter. In our simulations, we utilize the Stim simulator³³ and employ the SI1000 circuit-level noise model³⁴, which is well-suited for simulating superconducting experiments, as the error model. We use the minimum weight perfect matching decoder, “pymatching”, for decoding. In Fig. 2b, we observe that for a physical error rate $p = 0.002$, varying the defect rate from 0.005 to 0.02 (with consistent defect rates for qubits and couplers, independently assigning each component a defect probability of p) still allows us to exponentially suppress the logical error rate (LER) with an increase in code size L (size L device has $L \times L$ data qubits). This demonstrates that our method maintains the error correction capability of the surface code. Another observation is that lowering the defect rate can improve error suppression ability. Furthermore, we compare our results against a perfect lattice with

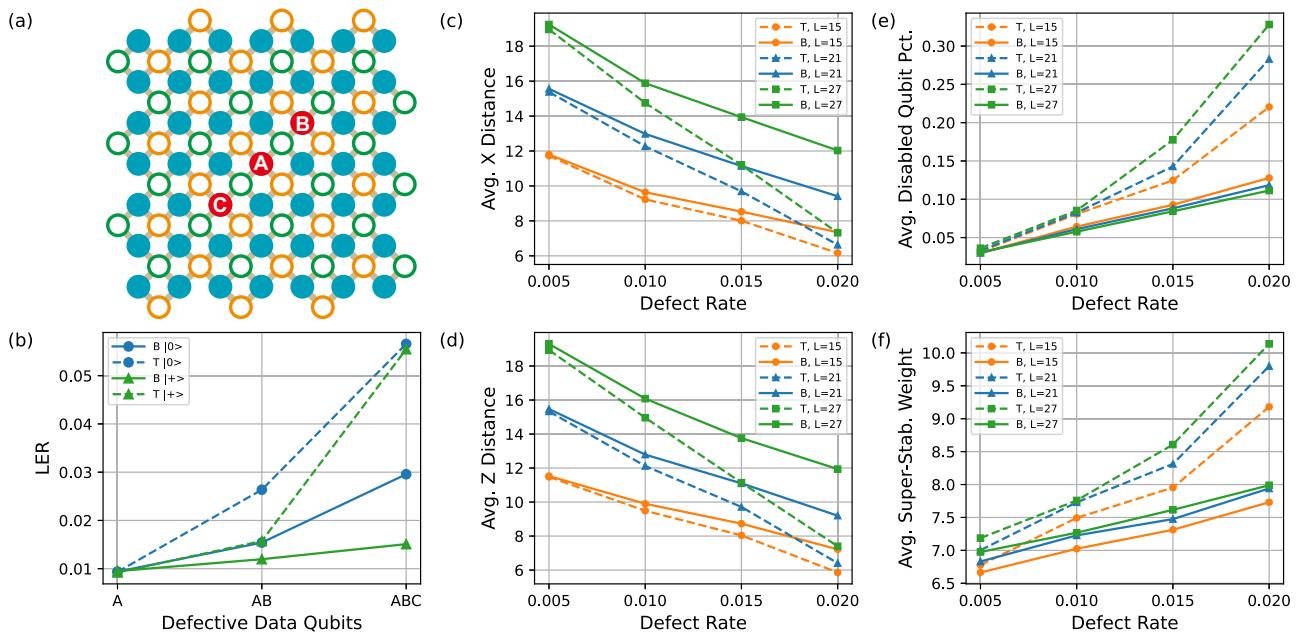


Fig. 3 | Performance comparison of the traditional method with our approach in handling defects. **a** An example with a code size of $L = 7$. Three highlighted circles represent potential defects, labeled as A, B, and C. We examine three scenarios: only defect A, defects A and B, and defects A, B, and C. **b** Comparison of logical error rate between bandage-like (B) and traditional (T) methods for the three scenarios in (a) at a physical error rate of $p = 0.002$. The bandage-like method shows significant advantages for both $|0\rangle_L$ and $|+\rangle_L$ states. **c-f** display statistics for the bandage-like

and traditional methods regarding (c) Average X distance, (d) Average Z distance, (e) Average disabled qubit percentage, and (f) Average super-stabilizer weight across different defect rates. Each data point is based on 100 generated devices with defects randomly distributed. In our simulations, defect rates are uniform for qubits and couplers. The bandage-like method consistently demonstrates substantial advantages, regardless of defect rates or code size. These advantages increase significantly with larger code sizes and defect rates.

SI1000 p ranging from 0.002 to 0.004. We find that for a 1% defect rate with $p = 0.002$, the error suppression ability of our adapter is comparable to that of defect-free devices with $p = 0.003$. This indicates that even at high defect rates, our adapter can perform equivalent to a defect-free lattice with physical error rate increasing only by 0.001, highlighting its practical utility.

Bandage-like super-stabilizer advantage

To further showcase the advantages of our approach, we compare it with traditional super-stabilizer methods^{29–32}. We start with a simple case of three scenarios with increasing defects for $L = 7$ devices, as seen in Fig. 3a. The results are then shown in Fig. 3b. When there's only one defect A, the two methods perform equally. However, as we move from A to AB and then ABC defects, the advantage of the bandage-like method becomes more pronounced. Specifically, the bandage method lowers the LER by 42% (24%) for the $|0\rangle_L$ ($|+\rangle_L$) state with 2 defects, AB, and this improves to 48% (73%) for 3 defects, ABC. This improvement happens because the bandage-like method keeps more code distance and lower-weight stabilizers. For the AB defect scenario, the traditional method provides a X (Z) distance of 5 (5), while the bandage-like method maintains 5 (6). Moreover, the average super-stabilizer weight decreases from 10 to 6.67. Similarly, in the ABC defect case, the traditional method's X (Z) distance is 4 (4), while the bandage-like method provides 4 (6). And the average super-stabilizer weight drops from 14 to 7 (refer to Supplementary Material for details).

To generalize this advantage to a broader context, we compare our bandage-like super-stabilizer approach with traditional methods^{29–32} for adapting multiple devices with randomly distributed defects. It's worth noting that we intentionally kept weight-1 and bridge syndrome qubits near boundary data qubits to prevent further boundary deformation for the traditional methods. This design choice slightly favored the traditional method, resulting in a higher baseline performance. In our simulations, we generated 100 devices with randomly distributed defects for each scenario for statistical purposes, and the defect rates are consistent across qubits and

couplers (refer to the Supplemental Material for scenarios of devices with only coupler defects).

Figure 3c, d shows the average code distance after adaptation with increasing defect rates. Naturally, code distance decreases with higher defect rates for both methods. However, the bandage-like method consistently maintains a superior code distance, and this advantage grows as defect rates increase, similar to the above specific case. For instance, at a code size of $L = 27$ and a defect rate of 0.01 (0.02), the average X distance improves from 14.8 (7.3) to 15.9 (12.0), and the Z distance from 15.0 (7.4) to 16.1 (11.9), marking a 7.6% (63%) average improvement.

The bandage-like method preserves code distance by disabling fewer qubits. To quantitatively illustrate this, Fig. 3e shows the average percentage of disabled qubits after adaptation. While disabled qubits increase with defect rates, the bandage method exhibits a slower increase, indicating better qubit preservation compared to the traditional method. For instance, at a code size of $L = 27$ and a defect rate of 0.01 (0.02), the average percentage of disabled qubits improves from 8.5% (32.8%) to 5.8% (11.1%).

Additionally, Fig. 3f shows the weighted average of the average super-stabilizer weight for each random device, calculated as: $w_{avg} = (\sum_d \sum_i w_{di}) / (\sum_d \sum_i 1)$. Here, d represents the index of the random device (100 devices with defects randomly distributed in our simulation), and i represents the index of the super-stabilizer in each device. w_{di} represents the weight of the super-stabilizer indexed by d and i . We can observe that the bandage-like method exhibits lower average super-stabilizer weights. For instance, at a code size of $L = 27$ and a defect rate of 0.01 (0.02), the average super-stabilizer weight improves from 7.8 (10.1) to 7.3 (8.0), marking a 6.3% (21%) improvement. This reduction in weight enhances the reliability of the super-stabilizer because lower-weight super-stabilizers are better at identifying errors within a more localized area, thus preventing error spread and enhancing error detection and correction capabilities.

Analyzing Fig. 3c-f collectively, we observe another significant trend: as code size increases under the same defect rates, the bandage-like method's

advantages over the traditional approach also increase in terms of code distance, disabled qubit percentage, and super-stabilizer weight. The reduction of qubit disabling is achieved through several mechanisms: retaining bridge and weight-1 qubits directly reduces qubit disabling; keeping these qubits prevents a potential avalanche effect of large-area qubit disabling; and preserving these qubits on the boundary prevents inward boundary pushing. These mechanisms ultimately provide better code distance and lower super-stabilizer weight, as detailed in the Supplementary Material. This scalability advantage is crucial for scaling quantum computing devices.

Discussion

The proposed defect-adaptive surface code has two main features: (1) Our algorithm automates the defect handling step by step, including boundary deformation, internal defect disabling, stabilizer patch, all without conflicts even in complex defect scenarios. This method allows us to obtain defect-adaptive surface codes automatically in all simulations, without manual intervention. Even with a high defect rate of 2%, our method still shows an exponential suppression in the logical error rate as the code distance increases, demonstrating the feasibility and excellent performance of our approach. These advantages are crucial for the large-scale expansion of quantum computing. (2) Unlike previous methods, our adapter utilizes a new type of bandage-like super-stabilizers, offering advantages in maintaining more qubits and code distance, and reducing super-stabilizer weight, especially in scenarios with clustered defects. This significantly reduces the logical error rate of quantum memory on defective lattices. In a simulation with three defects, our method reduces the logical error rate by 48% and 73% for the $|0\rangle_L$ and $|+\rangle_L$ states compared to previous works. Future interesting work will involve observing the performance of our experimentally-ready approach in real-world experiments and achieving error suppression by scaling surface code logical qubits under defective lattices.

Data availability

Data available at <https://github.com/DeviesVi/BandAuto>.

Code availability

Code available at <https://github.com/DeviesVi/BandAuto>.

Received: 7 August 2024; Accepted: 7 April 2025;

Published online: 12 May 2025

References

1. Kitaev, A. Fault-tolerant quantum computation by anyons. *Ann. Phys.* **303**, 2–30 (2003).
2. 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).
3. Raussendorf, R. & Harrington, J. Fault-tolerant quantum computation with high threshold in two dimensions. *Phys. Rev. Lett.* **98**, 190504 (2007).
4. Andersen, C. K. et al. Repeated quantum error detection in a surface code. *Nat. Phys.* **16**, 875–880 (2020).
5. Al, G. Q. Exponential suppression of bit or phase errors with cyclic error correction. *Nature* **595**, 383–387 (2021).
6. Marques, J. et al. Logical-qubit operations in an error-detecting surface code. *Nat. Phys.* **18**, 80–86 (2022).
7. Zhao, Y. et al. Realization of an error-correcting surface code with superconducting qubits. *Phys. Rev. Lett.* **129**, 030501 (2022).
8. Krinner, S. et al. Realizing repeated quantum error correction in a distance-three surface code. *Nature* **605**, 669–674 (2022).
9. Al, G. Q. Suppressing quantum errors by scaling a surface code logical qubit. *Nature* **614**, 676–681 (2023).
10. Bluvstein, D. et al. Logical quantum processor based on reconfigurable atom arrays. *Nature* **626**, 58–65 (2024).
11. Erhard, A. et al. Entangling logical qubits with lattice surgery. *Nature* **589**, 220–224 (2021).
12. Gidney, C. & Ekerå, M. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum* **5**, 433 (2021).
13. Smith, K. N., Ravi, G. S., Baker, J. M. & Chong, F. T. Scaling superconducting quantum computers with chiplet architectures. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* 1092–1109 (IEEE, 2022).
14. Wu, Y. et al. Strong quantum computational advantage using a superconducting quantum processor. *Phys. Rev. Lett.* **127**, 180501 (2021).
15. Zhu, Q. et al. Quantum computational advantage via 60-qubit 24-cycle random circuit sampling. *Sci. Bull.* **67**, 240–245 (2022).
16. Ye, Y. et al. Logical magic state preparation with fidelity beyond the distillation threshold on a superconducting quantum processor. *Phys. Rev. Lett.* **131**, 210603 (2023).
17. Gong, M. et al. Quantum neuronal sensing of quantum many-body states on a 61-qubit programmable superconducting processor. *Sci. Bull.* **68**, 906–912 (2023).
18. Arute, F. et al. Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505–510 (2019).
19. Martinis, J. M. Saving superconducting quantum processors from decay and correlated errors generated by gamma and cosmic rays. *npj Quantum Inf.* **7**, 90 (2021).
20. McEwen, M. et al. Resolving catastrophic error bursts from cosmic rays in large arrays of superconducting qubits. *Nat. Phys.* **18**, 107–111 (2022).
21. Wilen, C. D. et al. Correlated charge noise and relaxation errors in superconducting qubits. *Nature* **594**, 369–373 (2021).
22. Vepsäläinen, A. P. et al. Impact of ionizing radiation on superconducting qubit coherence. *Nature* **584**, 551–556 (2020).
23. Cardani, L. et al. Reducing the impact of radioactivity on quantum circuits in a deep-underground facility. *Nat. Commun.* **12**, 2733 (2021).
24. Vala, J., Whaley, K. B. & Weiss, D. S. Quantum error correction of a qubit loss in an addressable atomic system. *Phys. Rev. A* **72**, 052318 (2005).
25. Bermudez, A. et al. Assessing the progress of trapped-ion processors towards fault-tolerant quantum computation. *Phys. Rev. X* **7**, 041061 (2017).
26. Cong, I. et al. Hardware-efficient, fault-tolerant quantum computation with rydberg atoms. *Phys. Rev. X* **12**, 021049 (2022).
27. Stace, T. M., Barrett, S. D. & Doherty, A. C. Thresholds for topological codes in the presence of loss. *Phys. Rev. Lett.* **102**, 200501 (2009).
28. Nagayama, S., Fowler, A. G., Horsman, D., Devitt, S. J. & Van Meter, R. Surface code error correction on a defective lattice. *New J. Phys.* **19**, 023050 (2017).
29. Auger, J. M., Anwar, H., Gimeno-Segovia, M., Stace, T. M. & Browne, D. E. Fault-tolerance thresholds for the surface code with fabrication errors. *Phys. Rev. A* **96**, 042316 (2017).
30. Strikis, A., Benjamin, S. C. & Brown, B. J. Quantum computing is scalable on a planar array of qubits with fabrication defects. *Phys. Rev. App.* **19**, 064081 (2023).
31. Siegel, A., Strikis, A., Flatters, T. & Benjamin, S. Adaptive surface code for quantum error correction in the presence of temporary or permanent defects. *Quantum* **7**, 1065 (2023).
32. Lin, S. F. et al. Codesign of quantum error-correcting codes and modular chiplets in the presence of defects. In *Proc. 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS '24*, 216–231 (Association for Computing Machinery, 2024).
33. Gidney, C. Stim: a fast stabilizer circuit simulator. *Quantum* **5**, 497 (2021).
34. Gidney, C., Newman, M., Fowler, A. & Broughton, M. A fault-tolerant honeycomb memory. *Quantum* **5**, 605 (2021).

Acknowledgements

The authors are grateful for valuable discussions with Armands Strikis and Ke Liu. This research support by Key R & D Plan of Shandong Province (Grant No. 2024CXPT083). H.-L.H. acknowledges support from the National Natural Science Foundation of China (Grant No. 12274464), and Natural Science Foundation of Henan (Grant No. 242300421049). This research was supported by the Chinese Academy of Sciences, Anhui Initiative in Quantum Information Technologies, Shanghai Municipal Science and Technology Major Project (Grant No. 2019SHZDX01), Innovation Program for Quantum Science and Technology (Grant No. 2021ZD0300200), Special funds from Jinan science and Technology Bureau and Jinan high tech Zone Management Committee, Technology Committee of Shanghai Municipality, National Science Foundation of China (Grants No. 11905217, No. 11774326, No. 92476203), Shandong Provincial Natural Science Foundation (Grant No. ZR2022LLZ008), and Natural Science Foundation of Shanghai (Grant No. 23ZR1469600), the Shanghai Sailing Program (Grant No. 23YF1452600). X.Z. acknowledges support from the New Cornerstone Science Foundation through the XPLORE PRIZE. Y.Y. acknowledges support from the National Natural Science Foundation of China (Grant No. 12404575).

Author contributions

H.-L.H. conceived the research. H.-L.H., X.Z., and J.-W.P. provided supervision and guidance throughout the project. Z.W. designed the adapter and conducted numerical simulations. T.H. and D.W. performed calculations for the adapted code. Y.Y. made suggestions to ensure the adapter was suitable for a real superconducting device. Y.Z., Y. Zhao, and W.L. provided valuable suggestions during the development of the adapter. Z.W. and H.-L.H. wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s41534-025-01023-y>.

Correspondence and requests for materials should be addressed to He-Liang Huang, Xiaobo Zhu or Jian-Wei Pan.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025