Fermion-to-Fermion Low-Density Parity-Check Codes

Chong-Yuan Xu[1],[*] Ze-Chuan Liu[1],[*] and Yong Xu[1],[2][†]

[1]*Center for Quantum Information, IIIS, Tsinghua University, Beijing 100084, People's Republic of China and*
[2]*Hefei National Laboratory, Hefei 230088, People's Republic of China*

Simulating fermionic systems on qubit-based quantum computers often demands significant computational resources due to the requirement to map fermions to qubits. Thus, designing a fault-tolerant quantum computer that operates directly with fermions offers an effective solution to this challenge. Here, we introduce a protocol for fault-tolerant fermionic quantum computation utilizing fermion-to-fermion low-density parity-check (LDPC) codes. Our method employs a fermionic LDPC memory, which transfers its state to fermionic color code processors, where logical operations are subsequently performed. We propose using odd-weight logical Majorana operators to form the code space, serving as memory for the fermionic LDPC code, and provide an algorithm to identify these logical operators. We present examples showing that the encoding rate of fermionic codes often matches that of qubit codes, while the logical error rate can be significantly lower than the physical error rate. Furthermore, we propose two methods for performing fermionic lattice surgery to facilitate state transfer. Finally, we simulate the dynamics of a fermionic system using our protocol, illustrating effective error suppression.

The study of strongly correlated fermionic systems is central to high-energy physics [1], material science [2], and quantum chemistry [3], promising insights into phenomena ranging from quark dynamics [4] to high-temperature superconductivity [5]. However, solving these problems using classical computers is often very challenging. For instance, quantum Monte Carlo methods usually face the sign problem when addressing fermionic problems [6]. In this context, quantum computers offer a promising alternative [7]. However, since conventional quantum computers use qubits, solving fermionic problems requires mapping fermionic operators to qubit operators, which often incurs substantial overhead [8–13], making experimental implementation significantly challenging in the near term. To address this challenge, developing programmable fermionic quantum processors is increasingly appealing [10, 14]. In light of the unavoidable presence of noise, fault-tolerant fermionic quantum computing based on fermion-to-fermion repetition and color codes have been proposed very recently [15, 16]. However, the protocol's overhead remains substantial because the encoding rate is low, with each block encoding only a single logical fermion.

Recently, protocols based on quantum LDPC codes have been proposed to reduce the overhead in qubit-based fault-tolerant quantum computation [17–40]. Compared to the paradigmatic surface code, quantum LDPC codes feature a significantly higher encoding rate. Although these codes require long-range connectivity between qubits, recent technological advances in platforms [38, 41–43] such as Rydberg atom arrays, superconducting qubits, and trapped ions have made the near-term implementation of these codes promising. Despite these significant advancements, existing studies primarily focus on constructing fault-tolerant quantum computers with qubits. It remains unclear how to construct a fermion-to-fermion LDPC code capable of encoding mul-
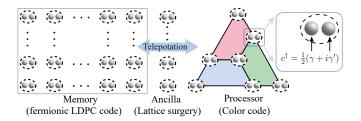

FIG. 1. Schematic illustration of fault-tolerant fermionic quantum computation framework. The architecture comprises three components: fermionic memory implemented using fermionic LDPC codes, fermionic processor implemented using fermionic color codes to execute logical operations, and fermionic interface composed of ancilla fermions serving as a critical bridge for communication between memory and processor.

tiple fermion modes in a single code block and how to perform logical operations on such codes.

Here, we introduce a protocol for fault-tolerant fermionic quantum computation based on fermion-to-fermion LDPC codes. Our approach employs a fermionic LDPC memory integrated with fermionic color code processors as illustrated in Fig. 1, which is generalized from the quantum LDPC case [37]. Logical fermionic information is initially stored in the fermionic LDPC memory and, when needed, is transferred to fermionic color code processors, where fault-tolerant logical operations are conducted. Upon completion, the information is returned to the memory. To construct the fermionic LDPC memory, we develop a systematic workflow for creating fermionic stabilizer codes based on self-dual Calderbank-Shor-Steane (CSS) codes. These codes are characterized by logical Majorana operators consisting of an odd number of physical Majorana operators. We construct three distinct classes of fermionic LDPC codes using three different LDPC codes: bicycle codes [44], finite Euclidean geometry codes [45, 46], and finite projective geome-

try codes [47]. These codes can have a significantly higher encoding rate compared to the fermionic color code. Furthermore, we propose two methods for performing fermionic lattice surgery to facilitate transfer of logical fermionic states between the fermionic memory and processor. We show that this process maintains the code distance. Finally, we simulate the dynamics of a fermionic system, demonstrating effective error suppression.

We start by introducing the construction of our fermionic LDPC codes from $2n$ physical Majorana fermion operators $\{\gamma_1, \gamma_1', \gamma_2, \gamma_2', \ldots, \gamma_n, \gamma_n'\}$ arising from n physical complex fermion's creation and annihilation operators [48]. The strings of these Majorana operators, together with a phase factor $\eta \in \{\pm 1, \pm i\}$, generate the group of Majorana operators $\mathrm{Maj}(2n) \equiv \left\{ \Gamma = \eta \prod_{j=1}^{n} \gamma_j^{\alpha_j} (\gamma_j')^{\alpha_j'} | \alpha_j, \alpha_j' \in \{0, 1\} \right\}$ [49]. The number of Majorana operators in Γ is referred to as its weight. The Majorana stabilizer code is defined by a Majorana stabilizer group $\mathcal{S}_{\mathrm{maj}}$, a subgroup of $\mathrm{Maj}(2n)$, where all elements are Hermitian, mutually commute, ensuring it is an Abelian subgroup, have even weight to preserve the parity of a physical fermion system, and $-I$ with I being the identity element in the group is excluded [49]. The logical operators are generated by an independent subset of $\mathrm{Maj}(2n)$ comprising Majorana operator strings that commute with all elements of the stabilizer group $\mathcal{S}_{\mathrm{maj}}$ but are not members of $\mathcal{S}_{\mathrm{maj}}$ [50].

Each Majorana string operator can be represented by a binary vector $(\alpha_1, \alpha_1', \ldots, \alpha_n, \alpha_n')$, where $\alpha_j, \alpha_j' \in \{0, 1\}$ and $1 \le j \le n$. The binary vectors corresponding to the m independent stabilizer generators of $\mathcal{S}_{\mathrm{maj}}$ form an $m \times n$ check matrix. For a fermion-to-fermion LDPC code, we construct a check matrix H based on a self-dual CSS code with the binary check matrices that satisfy $H_X = H_Z = A$ and $AA^T = 0$, as follows,

$$H = \begin{pmatrix} H_\gamma & 0 \\ 0 & H_{\gamma'} \end{pmatrix}, \tag{1}$$

where $H_\gamma = H_{\gamma'} = A$. H_γ and $H_{\gamma'}$ correspond to check matrices that characterize the stabilizers consisting exclusively of $\{\gamma_j\}$ and $\{\gamma_j'\}$ operators, respectively, as described for fermion-to-fermion color codes in Ref. [15]. The logical operators are represented by vectors in the kernal of H that cannot be generated by stabilizers, forming a homology group $\ker(A)/\mathrm{im}(A)$ for both γ and γ' types of logical operators [51].

In our protocol, there are multiple logical fermions within a single fermionic LDPC memory, and these logical fermions are transferred to processors for the execution of logical operations. It is essential that the logical Majorana operator in memory anticommutes with that in the processor to adhere to fermionic statistics. We thus require all logical Majorana operators in memory have odd weight and even overlap with each other [49].
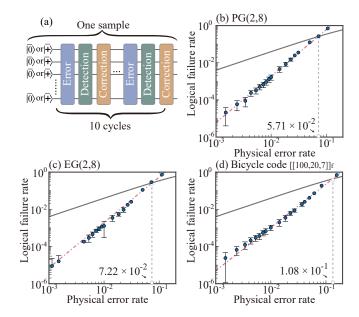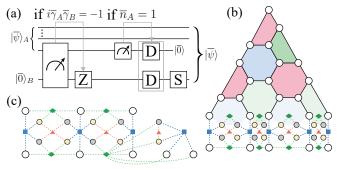

FIG. 2. (a) Error model used to benchmark the capacity of our fermionic LDPC code memory initialized in either the logical vacuum state $\overline{|0 \ldots 0\rangle}$ or the logical product state $\overline{|+ \ldots +\rangle}$, where $\overline{\gamma}\overline{|+\rangle} = \overline{|+\rangle}$. Each sample consists of 10 cycles, each incorporating three layers: random errors, syndrome measurements, and error correction. (b)-(d) The numerically calculated logical failure rate with respect to the physical error rate for three fermionic LDPC codes: PG(2,8), EG(2,8), and Bicycle code $[[100, 20, 7]]_\mathrm{f}$. The dashed red line indicates a power law fit to the data, and the gray line represents the probability that an error occurs on at least one physical site (see Supplemental Material). The vertical dashed gray line highlights the code's pseudo-threshold.

Such conditions also ensure that the fermionic statistics are maintained when concatenating different fermionic code blocks [15]. We denote the logical Majorana operators as $\overline{\gamma}_j$ and $\overline{\gamma}_j'$ with $1 \le j \le k$ (k is the number of logical complex fermions), which consist of physical Majorana operators $\{\gamma_1, \ldots, \gamma_n\}$ and $\{\gamma_1', \ldots, \gamma_n'\}$, respectively. The logical complex fermion creation operators are then defined as $\overline{c}_j^\dagger = \frac{1}{2}(\overline{\gamma}_j + i\overline{\gamma}_j')$ [48].

Previous studies have primarily focused on encoding Majorana modes into qubits [49, 52, 53]. While some results also demonstrate the existence of odd-weight logical operators in Majorana surface codes [54], there is a lack of methods for systematically identifying these operators within a fermionic LDPC code. To identify them, we generalize the Gram-Schmidt orthogonalization process to vector spaces over $\mathbb{F}_2$ [55]. Let $\{\vec{\gamma}_j | j = 1, \ldots, k', k' \ge k\}$ be a basis for $\ker A/\mathrm{im}(A)$, which is itself also a vector space; $\vec{\gamma}_j$ represents the corresponding logical $\overline{\gamma}_j$ or $\overline{\gamma}_j'$ operator, expressed as $\gamma_1^{[\vec{\gamma}_j]_1} \ldots \gamma_n^{[\vec{\gamma}_j]_n}$ or $(\gamma_1')^{[\vec{\gamma}_j]_1} \ldots (\gamma_n')^{[\vec{\gamma}_j]_n}$. If there exists a basis vector with odd number of 1's, say $\vec{\gamma}_1$, we remove it from the basis and make the remaining basis vectors orthogonal to it via $\vec{\gamma}_j \to \vec{\gamma}_j - (\vec{\gamma}_1 \cdot \vec{\gamma}_j)\vec{\gamma}_1$, i.e., $\vec{\gamma}_j$ has

: Measurement stabilizer   ◆ : Modified stabilizer   ○ : Majorana operator
▲ : Gauge stabilizer   ● ○ : A pair of ancilla Majorana operators

FIG. 3. (a) Quantum circuit to transfer a logical fermionic state $|\overline{\psi}\rangle_A$ to code block B (initialized as $|\overline{0}\rangle_B$). The protocol involves: (i) measuring $i\overline{\gamma}_A\overline{\gamma}_B$ and applying the Z gate of $\exp(i\pi\overline{n}_B)$, provided the measurement result is -1; (ii) measuring the logical particle number $\overline{n}_A$ and applying the D gate defined as $\overline{D} = \exp(i\frac{\pi}{2}\overline{\gamma})$ to both code block A and B if the measurement outcome is 1; (iii) applying a phase rotation gate to the state in the processor. The validation of this circuit is provided in Supplemental material. (b) Illustration of the first method for lattice surgery achieving fault-tolerant measurements of $i\overline{\gamma}_A\overline{\gamma}_B$ in the case with $|\overline{\gamma}_A| = |\overline{\gamma}_B|$. In this example, code B is a $d = 5$ fermionic color code, and code A is a fermionic LDPC code containing five Majorana operators in the support of $\overline{\gamma}_A$. The empty circles represent the γ-type Majorana operators in blocks A and B. Several pairs of ancilla Majorana operators are introduced with each horizontal pair represented by yellow and gray circles to denote γ-type and γ'-type Majorana operator forming a complex fermion. The γ-type stabilizer generators at the boundary, connected to ancilla fermions, are modified to incorporate the ancilla Majorana operators (green rhombuses). Measurement and gauge stabilizer generators are depicted as blue squares and red triangles, respectively. The product of these stabilizer generators yields the joint logical operator $i\overline{\gamma}_A\overline{\gamma}_B$. The γ'-type Majorana stabilizers remain unchanged. The general procedure for introducing ancilla Majorana operators and constructing the corresponding stabilizers is detailed in Supplemental Material. (d) Illustration of the second method for fermionic lattice surgery between two fermionic LDPC codes with arbitrary weights.

even overlap with all others. This procedure is repeated until no odd-weight basis vector remains. The obtained odd-weight vectors correspond to logical operators which have odd weight, and the overlap weight between any two of them is even, thus satisfying the fermionic anticommutation relation. The necessary and sufficient condition for the existence of at least one odd-weight logical is $(1, 1, \ldots, 1)^T \notin \text{im}(A)$ [49]. We find that for most cases we consider the procedure produces a linearly independent set containing k' odd-weight binary vectors (see Supplemental Material). The resulting odd-weight logical operators define a fermionic subspace code for our fermionic LDPC code.

Based on three different LDPC codes, which include finite projective geometry codes [47], finite Euclidean geometry codes [45, 46], and bicycle codes [44], we construct three classes of fermionic LDPC codes (see Supplemental Material). We find that these subspace fermionic LDPC codes yield the same encoding rate as the original ones. We consider one code from each type and benchmark their logical information resilience as a fermion memory through numerical simulations. Specifically, we randomly initialize the logical state in either the logical vacuum state $\overline{|0 \ldots 0\rangle}$ or the logical product state $\overline{|+ \ldots +\rangle}$ ($\overline{\gamma}|+\rangle = |+\rangle$), both of which can be prepared by a series of braiding operations [56, 57]. We then simulate errors occurring as physical single-fermion gate γ_j, γ'_j, and $i\gamma_j\gamma'_j$ with equal probability $p/3$, where p denotes the physical error rate. Subsequently, we perform error detection via syndrome measurements (assuming perfect syndrome measurements for simplicity), followed by error decoding using the belief propagation and ordered-statistical decoding (BP+OSD) algorithm [58, 59]. After decoding, we correct errors by applying physical single-fermion gates and then repeat the entire process for the corrected new state. For each sample, we conduct $N_c = 10$ rounds of error-correction circuit cycles (each cycle contains three layers: random errors, syndrome measurements, and error correction), as shown in Fig. 2(a). The logical error probability for each sample is defined as $P_L(N_c) = N_{\text{error}}/N_{\text{sample}}$, where N_{error} and N_{sample} denote the number of erroneous trials and total trials, respectively. The logical failure rate is then $p_L = 1 - (1 - P_L(N_c))^{1/N_c}$ [37, 38].

Figure 2 shows the logical error rate with respect to the physical error rate for the finite projective geometry $\text{PG}(2, 8)$ code, the finite Euclidean geometry $\text{EG}(2, 4)$ code, and the $[[100, 20, 7]]_f$ bicycle code. We clearly see that the logical error rate is significantly suppressed compared to the physical one according to $p_L \sim p^\alpha$ with $\alpha \approx 2$. Following the convention [38], we define the code's pseudo-threshold as the solution to the break-even equation $p_L(p) = P(p, k)$, where $P(p, k)$ is the probability of at least one error occurring on k physical fermion sites at physical error rate p. Under this error model, all three codes exhibit relatively high pseudo-thresholds.

In Supplemental Material, we also study several other self-dual codes such as Kitaev Majorana code [48] and unicycle codes [44]. We find that they are not suitable for producing well-behaved fermionic LDPC because they do not support the generation of odd-weight logical operators.

We now study how to perform logical operations on the fermionic LDPC memory. Instead of executing logical gates directly on the fermionic LDPC, we consider a method whereby the states of a target logical fermion mode in memory are transferred to an external fermionic color code processor. Logical operations are then performed on these processors, and the states are subsequently transferred back to the memory. This approach is inspired by fault-tolerant measurements of logical op-
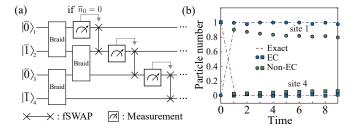
FIG. 4. (a) Quantum circuit used to benchmark our protocol. We first initialize the system in the logical state $|\overline{0101}\rangle$ and then apply three braid gates with each gate executing the operation $B_{j,j+1} = \exp(i\pi(\bar{c}_j^\dagger \bar{c}_{j+1} + \text{H.c.})/2)$. Onsite measurements of $\bar{n}_j = \bar{c}_j^\dagger \bar{c}_j$ with $j = 0, 1, 2$ are then performed, followed by a fermionic swap (fSWAP) gate if the measurement result is zero. The unitary gates and feedback process constitute a module, which is executed consecutively multiple times. (b) The expectation value of logical particle numbers at site 1 and 4 with respect to time which is characterized by the number of executed modules. We see that the simulation under noise with error correction (filled blue circles and squares) is very close to the noiseless simulation result (blue and red dotted-dashed lines), in stark contrast to the simulation results under noise without error correction (filled green circles and squares). The error bars for blue circles and green squares are hidden behind the symbols.

erators in qubit LDPC codes [31, 37, 60–62]. If the logical operation involves two logical fermion modes, we consider two color code processors. The logical information of the two fermion modes is individually transferred, logical operations are applied between these two color code blocks, and the information is subsequently returned to memory. Logical operations involving more logical fermion modes can be performed similarly. For fermionic state transfers between memory and processor, we design a measurement-based circuit as illustrated in Fig. 3(a). This circuit enables the transfer of a single logical fermion information described by logical Majorana operators $\bar{\gamma}_A$ and $\bar{\gamma}'_A$ to a processor described by logical Majorana operators $\bar{\gamma}_B$ and $\bar{\gamma}'_B$, which is initialized in the logical vacuum state $|\bar{0}\rangle_B$. In other words, an output state is the same as the initial state except that the processor logical fermion plays the role of the corresponding logical fermion in memory. During the transfer, a key step is the fault-tolerant measurement of the joint logical operator $i\bar{\gamma}_A\bar{\gamma}_B$, as shown in Fig. 3(a).

We propose two methods for fermionic lattice surgery to realize the fault-tolerant joint measurement as shown in Fig. 3(b) and (c). Let d_A be the weight of $\bar{\gamma}_A$ and $\bar{\gamma}'_A$. For simplicity, we consider the case where code B is a fermionic color code whose distance satisfies that $d_B = d_A$ and put the general case where code B is also a fermionic LDPC code in Supplemental Material. To realize the measurement, we align the support of the logical operator $\bar{\gamma}_A$ and $\bar{\gamma}_B$ and introduce a set Q_C of $4(d_A - 1)$ ancilla Majorana operators (correspond-

ing to $2(d_A - 1)$ complex fermions) between these two codes, labeled as $\gamma_{a,1}, \gamma'_{a,1}, \ldots, \gamma_{a,(d_A-1)}, \gamma'_{a,(d_A-1)}$ and $\gamma_{b,1}, \gamma'_{b,1}, \ldots, \gamma_{b,(d_A-1)}, \gamma'_{b,(d_A-1)}$. The original γ-type stabilizer generators at the boundary are modified by including two ancilla Majorana operators as shown in Fig. 3(b). Measurement stabilizer generators $M_1, M_2, \ldots, M_{d_A}$ and gauge stabilizer generators $G_1, G_2, \ldots, G_{d_A-1}$ are also introduced. The original γ'-type stabilizer generators for both block of A and B are not modified. This merging procedure creates a merged stabilizer code $\mathcal{C}_{\text{merged}}$ where $i\bar{\gamma}_A\bar{\gamma}_B$ becomes a stabilizer. Since $2(d_A - 1)$ ancilla complex fermions and $2d_A - 1$ new stabilizers are introduced, this new code only contains one logical complex fermion described by $\bar{\gamma}'_A$ and $\bar{\gamma}'_B$. In other words, one cannot find γ-type logical operators in the merged code. In this case, we show that the merging procedure does not decrease the code distance based on the formalism of the subsystem code [63, 64] (see Supplemental Material). The number of the Majorana modes introduced during the whole process is small compared to the fermionic LDPC memory, resulting in a small resource overhead (see Supplemental Material). The Supplemental Material also provides the guidelines for introducing ancilla Majorana operators and constructing stabilizer generators for this method. Moreover, we propose a more comprehensive scheme for performing lattice surgery between two fermionic stabilizer codes (see Fig. 3(c) for an example) with details in Supplemental Material.

To achieve the fault-tolerant measurement of $i\bar{\gamma}_A\bar{\gamma}_B$, we first initialize the physical ancilla complex fermions in the $|0\rangle$ state and then measure all the stabilizer generators in the merged code and perform $\min\{d_A, d_B\}$ round of error corrections in the presence of noisy measurements to ensure fault-tolerance. Finally, each physical ancilla fermion is measured in the particle number basis to return the state to the original code space.

We now demonstrate the capability of our codes by simulating the quantum circuit of fermions shown in Fig. 4(a). The circuit incorporates braid operations that enable the tunneling of fermions between neighboring sites, and onsite measurements, followed by fermionic swap gates, provided the measurement outcome is zero. For an initial state $|\overline{0101}\rangle$, we expect that the steady state of this dynamical process is a skin state where fermionic particles mainly reside in the upper half part due to feedback effects [65–67]. To simulate the dynamic behavior, we utilize the PG(2, 8) code as our fermionic LDPC code and select four encoded complex fermion modes from it. Additionally, we introduce two copies of the fermionic Steane code to function as processors. The calculated time evolution of the particle number at the first and fourth site is shown in Fig. 4(b). The value of $\langle\bar{n}_1\rangle$ suddenly increases to one while the value of $\langle\bar{n}_4\rangle$ decreases to zero, which aligns with the expected characteristics of a skin state. Notably, with our fault-tolerant logical operations and error correction techniques, the error-corrected

data demonstrate significantly higher fidelity compared to the uncorrected data, as shown in Fig. 4(b). The time evolution results using the second method for lattice surgery is provided in Supplemental Material.

In summary, we have developed a systematic workflow for constructing fermionic LDPC code memory based on self-dual CSS codes and proposed methods for executing logical operations based on lattice surgery. We demonstrate that our fermionic LDPC code can be used to simulate dynamics of fermionic systems fault-tolerantly with effective error detection and correction. Given that our protocol employs gates like the D gate, which does not conserve the parity of fermions, one can consider using referenced fermionic modes as our physical modes, which can be realized with the assistance of auxiliary reference modes [16]. Such fermion-to-fermion quantum computation has the potential to significantly reduce the computational complexity associated with simulating fermionic systems. Our work encourages the pursuit of high-performance fermionic LDPC codes with higher encoding rates and larger code distances, as well as the advancement of more efficient methods for executing logical operations with reduced overhead.

Note added. During the final stage of this work, we became aware of a related work [68], which uses Majorana LDPC codes to encode logical qubits.

---

* These authors contribute equally to this work.
† yongxuphy@tsinghua.edu.cn

[1] C. W. Bauer, Z. Davoudi, A. B. Balantekin, T. Bhattacharya, M. Carena, W. A. De Jong, P. Draper, A. El-Khadra, N. Gemelke, M. Hanada, *et al.*, Quantum simulation for high-energy physics, PRX quantum **4**, 027001 (2023).

[2] B. Bauer, S. Bravyi, M. Motta, and G. K.-L. Chan, Quantum algorithms for quantum chemistry and quantum materials science, Chem. Rev. **120**, 12685 (2020).

[3] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Quantum computational chemistry, Rev. Mod. Phys. **92**, 015003 (2020).

[4] J. Berges, M. P. Heller, A. Mazeliauskas, and R. Venugopalan, Qcd thermalization: Ab initio approaches and interdisciplinary connections, Rev. Mod. Phys. **93**, 035003 (2021).

[5] C. M. Varma, Colloquium: Linear in temperature resistivity and associated mysteries including high temperature superconductivity, Rev. Mod. Phys. **92**, 031001 (2020).

[6] M. Troyer and U.-J. Wiese, Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations, Phys. Rev. Lett. **94**, 170201 (2005).

[7] R. P. Feynman, Simulating physics with computers, Int. J. Theor. Phys. **21**, 467 (1982).

[8] D. S. Abrams and S. Lloyd, Simulation of many-body fermi systems on a universal quantum computer, Phys. Rev. Lett. **79**, 2586 (1997).

[9] G. Ortiz, J. E. Gubernatis, E. Knill, and R. Laflamme, Quantum algorithms for fermionic simulations, Phys. Rev. A **64**, 022319 (2001).

[10] S. B. Bravyi and A. Y. Kitaev, Fermionic quantum compuation, Ann. Phys. **298**, 210 (2002).

[11] R. C. Ball, Fermions without fermion fields, Phys. Rev. Lett. **95**, 176407 (2005).

[12] F. Verstraete and J. I. Cirac, Mapping local hamiltonians of fermions to local hamiltonians of spins, J. Stat. Mech.: Theory Exp. **2005** (09), P09012.

[13] J. D. Whitfield, V. c. v. Havlíček, and M. Troyer, Local spin operators for fermion simulations, Phys. Rev. A **94**, 030301 (2016).

[14] D. González-Cuadra, D. Bluvstein, M. Kalinowski, R. Kaubruegger, N. Maskara, P. Naldesi, T. V. Zache, A. M. Kaufman, M. D. Lukin, H. Pichler, *et al.*, Fermionic quantum processing with programmable neutral atom arrays, Proc. Natl. Acad. Sci. **120**, e2304294120 (2023).

[15] A. Schuckert, E. Crane, A. V. Gorshkov, M. Hafezi, and M. J. Gullans, Fermion-qubit fault-tolerant quantum computing, arXiv:2411.08955 (2024).

[16] R. Ott, D. González-Cuadra, T. V. Zache, P. Zoller, A. M. Kaufman, and H. Pichler, Error-corrected fermionic quantum processors with neutral atoms, Phys. Rev. Lett. , (2025).

[17] J.-P. Tillich and G. Zémor, Quantum ldpc codes with positive rate and minimum distance proportional to the square root of the blocklength, IEEE Transactions on Information Theory **60**, 1193 (2013).

[18] D. Gottesman, Fault-tolerant quantum computation with constant overhead, Quantum Information and Computation **14**, 1338 (2013).

[19] A. A. Kovalev and L. P. Pryadko, Quantum kronecker sum-product low-density parity-check codes with finite rate, Phys. Rev. A **88**, 012311 (2013).

[20] N. P. Breuckmann and B. M. Terhal, Constructions and noise threshold of hyperbolic surface codes, IEEE transactions on Information Theory **62**, 3731 (2016).

[21] P. Panteleev and G. Kalachev, Degenerate quantum ldpc codes with good finite length performance, Quantum **5**, 585 (2021).

[22] N. P. Breuckmann and J. N. Eberhardt, Balanced product quantum codes, IEEE Transactions on Information Theory **67**, 6653 (2021).

[23] A. Krishna and D. Poulin, Fault-tolerant gates on hypergraph product codes, Phys. Rev. X **11**, 011023 (2021).

[24] O. Higgott and N. P. Breuckmann, Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead, Phys. Rev. X **11**, 031039 (2021).

[25] N. P. Breuckmann and J. N. Eberhardt, Quantum low-density parity-check codes, PRX quantum **2**, 040101 (2021).

[26] P. Panteleev and G. Kalachev, Quantum ldpc codes with almost linear minimum distance, IEEE Transactions on Information Theory **68**, 213 (2021).

[27] N. Delfosse, M. E. Beverland, and M. A. Tremblay,

Bounds on stabilizer measurement circuits and obstructions to local implementations of quantum ldpc codes, arXiv:2109.14599 (2021).

[28] N. Baspin and A. Krishna, Connectivity constrains quantum codes, Quantum 6, 711 (2022).

[29] N. Baspin and A. Krishna, Quantifying nonlocality: How outperforming local quantum codes is expensive, Phys. Rev. Lett. 129, 050505 (2022).

[30] M. A. Tremblay, N. Delfosse, and M. E. Beverland, Constant-overhead quantum error correction with thin planar connectivity, Phys. Rev. Lett. 129, 050504 (2022).

[31] L. Z. Cohen, I. H. Kim, S. D. Bartlett, and B. J. Brown, Low-overhead fault-tolerant quantum computing using long-range connectivity, Sci. Adv. 8, eabn1717 (2022).

[32] P. Panteleev and G. Kalachev, Asymptotically good quantum and locally testable classical ldpc codes, in Proceedings of the 54th annual ACM SIGACT symposium on theory of computing (2022) pp. 375–388.

[33] A. Leverrier and G. Zémor, Quantum tanner codes, in 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS) (IEEE, 2022) pp. 872–883.

[34] A. Strikis and L. Berent, Quantum low-density parity-check codes for modular architectures, PRX Quantum 4, 020321 (2023).

[35] A. O. Quintavalle, P. Webster, and M. Vasmer, Partitioning qubits in hypergraph product codes to implement logical gates, Quantum 7, 1153 (2023).

[36] H.-K. Lin and L. P. Pryadko, Quantum two-block group algebra codes, Phys. Rev. A 109, 022407 (2024).

[37] Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays, Nat. Phys. 20, 1084 (2024).

[38] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, High-threshold and low-overhead fault-tolerant quantum memory, Nature 627, 778 (2024).

[39] G. Zhang and Y. Li, Time-efficient logical operations on quantum low-density parity check codes, Phys. Rev. Lett. 134, 070602 (2025).

[40] Y. Li, Low-density parity-check representation of fault-tolerant quantum circuits, Phys. Rev. Res. 7, 013115 (2025).

[41] D. Bluvstein, H. Levine, G. Semeghini, T. T. Wang, S. Ebadi, M. Kalinowski, A. Keesling, N. Maskara, H. Pichler, M. Greiner, et al., A quantum processor based on coherent transport of entangled atom arrays, Nature 604, 451 (2022).

[42] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, et al., Logical quantum processor based on reconfigurable atom arrays, Nature 626, 58 (2024).

[43] J.-S. Chen, E. Nielsen, M. Ebert, V. Inlek, K. Wright, V. Chaplin, A. Maksymov, E. Páez, A. Poudel, P. Maunz, et al., Benchmarking a trapped-ion quantum computer with 30 qubits, Quantum 8, 1516 (2024).

[44] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, Sparse-graph codes for quantum error correction, IEEE Transactions on Information Theory 50, 2315 (2004).

[45] D. Cao, Y. L. Song, and S. M. Zhao, A novel construction of quantum ldpc codes based on cyclic classes of lines in euclidean geometries, J. Electronics 29, 1 (2012).

[46] S. A. Aly, A class of quantum ldpc codes constructed from finite geometries, in IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference (IEEE, 2008) pp. 1–5.

[47] J. Farinholt, Quantum ldpc codes constructed from point-line subsets of the finite projective plane, arXiv:1207.0732 (2012).

[48] A. Kitaev, Anyons in an exactly solved model and beyond, Ann. Phys. 321, 2 (2006).

[49] S. Bravyi, B. M. Terhal, and B. Leemhuis, Majorana fermion codes, New J. Phys. 12, 083039 (2010).

[50] M. A. Nielsen and I. L. Chuang, Quantum computation and quantum information (Cambridge university press, 2010).

[51] H. Bombin and M. A. Martin-Delgado, Homological error correction: Classical and quantum codes, J. Math. Phys. 48, 052105 (2007).

[52] D. Litinski and F. von Oppen, Quantum computing with majorana fermion codes, Phys. Rev. B 97, 205404 (2018).

[53] S. Vijay, T. H. Hsieh, and L. Fu, Majorana fermion surface code for universal quantum computation, Phys. Rev. X 5, 041038 (2015).

[54] C. McLauchlan and B. Béri, A new twist on the majorana surface code: Bosonic and fermionic defects for fault-tolerant quantum computation, Quantum 8, 1400 (2024).

[55] Y. Prasetia, R. Kurnia, A. Andriko, and P. Astuti, Extended gram-schmidt process on sesquilinear spaces over finite fields, International Electronic Journal of Algebra , 1 (2025).

[56] C. K. McLauchlan and B. Béri, Fermion-parity-based computation and its majorana-zero-mode implementation, Phys. Rev. Lett. 128, 180504 (2022).

[57] M. Mudassar, R. W. Chien, and D. Gottesman, Encoding majorana codes, Phys. Rev. A 110, 032430 (2024).

[58] J. Liang, Q. Wang, L. Li, L. Song, and X. Ma, A low-complexity bp-osd algorithm for quantum ldpc codes, The European Physical Journal Special Topics , 1 (2025).

[59] J. Roffe, D. R. White, S. Burton, and E. Campbell, Decoding across the quantum low-density parity-check code landscape, Phys. Rev. Res. 2, 043423 (2020).

[60] D. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, Surface code quantum computing by lattice surgery, New J. Phys. 14, 123011 (2012).

[61] A. G. Fowler and C. Gidney, Low overhead quantum computation using lattice surgery, arXiv:1808.06709 (2018).

[62] H. Poulsen Nautrup, N. Friis, and H. J. Briegel, Fault-tolerant interface between quantum memories and quantum processors, Nat. Commun. 8, 1321 (2017).

[63] C. Vuillot, L. Lao, B. Criger, C. G. Almudéver, K. Bertels, and B. M. Terhal, Code deformation and lattice surgery are gauge fixing, New J. Phys. 21, 033028 (2019).

[64] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli, Subsystem codes, arXiv preprint quant-ph/0610153 (2006).

[65] Y.-P. Wang, C. Fang, and J. Ren, Absence of measurement-induced entanglement transition due to feedback-induced skin effect, Phys. Rev. B 110, 035113 (2024).

[66] X. Feng, S. Liu, S. Chen, and W. Guo, Absence of logarithmic and algebraic scaling entanglement phases due to the skin effect, Phys. Rev. B 107, 094309 (2023).

[67] Z.-C. Liu, K. Li, and Y. Xu, Dynamical transition due to feedback-induced skin effect, Phys. Rev. Lett. 133, 090401 (2024).

[68] M. Mudassar, A. Schuckert, and D. Gottesman, Fault tolerant operations in majorana-based quantum codes:

Gates, measurements and high rate constructions, arXiv:2508.09928 (2025).

[69] D. Gottesman, The heisenberg representation of quantum computers, arXiv preprint quant-ph/9807006 (1998).

[70] B. Segre, Ovals in a finite projective plane, Canadian Journal of Mathematics **7**, 414 (1955).

[71] L. Storme, Finite geometry, in *Handbook of combinatorial designs* (Chapman and Hall/CRC, 2006) pp. 728–754.

[72] W. E. Ryan *et al.*, An introduction to ldpc codes, CRC Handbook for Coding and Signal Processing for Recording Systems **5**, 1 (2004).

[73] A. J. Malcolm, A. N. Glaudell, P. Fuentes, D. Chandra, A. Schotte, C. DeLisle, R. Haenel, A. Ebrahimi, J. Roffe, A. O. Quintavalle, *et al.*, Computing efficiently in qldpc codes, arXiv:2502.07150 (2025).

[74] Y. Li, Fault-tolerant fermionic quantum computation based on color code, Phys. Rev. A **98**, 012336 (2018).

[75] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, arXiv:1108.5738 https://doi.org/10.48550/arXiv.1108.5738 (2011).

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

[77] G. Fan and C.-Q. Zhang, Circuit decompositions of eulerian graphs, Journal of Combinatorial Theory, Series B **78**, 1 (2000).

[78] P. O. Pinheiro, A. O. Alexandrino, A. R. Oliveira, C. C. de Souza, and Z. Dias, Algorithms for the maximum eulerian cycle decomposition problem, arXiv:2203.05446 https://doi.org/10.48550/arXiv.2203.05446 (2022).

[79] A. A. Albert and R. Sandler, *An introduction to finite projective planes* (Courier Corporation, 2015).

[80] J. L. Gross, J. Yellen, and M. Anderson, *Graph theory and its applications* (Chapman and Hall/CRC, 2018).

[81] J. Roffe, Quantum error correction: an introductory guide, Contemporary Physics **60**, 226 (2019).

In the Supplemental Material, we will review Majorana stabilizer codes and detail our method for identifying odd-weight logical operators in a fermionic LDPC code in Section S-1, validate the quantum state teleportation circuit in Section S-2, present a detailed implementation of fermionic lattice surgery, analyze its overhead, and prove the fault-tolerance of this procedure in Section S-3, and finally provide more details regarding calculations of the logical failure rate of the fermionic LDPC memory and simulations of fermionic circuits in Section S-4.

S-1. CONSTRUCTION OF FERMIONIC LDPC CODES

In this section, we will follow Ref. [49] to review Majorana stabilizer codes and detail our method for identifying odd-weight logical operators in a fermionic LDPC code.

Majorana stabilizer codes

In this subsection, we will review the Majorana stabilizer codes (also see Ref. [49]). Consider the group of Majorana operators $\mathrm{Maj}(2n) \equiv \left\{ \Gamma = \eta \prod_{j=1}^{n} \gamma_j^{\alpha_j}(\gamma_j')^{\alpha_j'} | \alpha_j, \alpha_j' \in \{0,1\}, \eta \in \{\pm 1, \pm i\} \right\}$, where γ_j and γ_j' with $j = 1, \ldots, n$ are Majorana operators satisfying $\{\gamma_i, \gamma_j\} = 2\delta_{ij}$, $\{\gamma_i', \gamma_j'\} = 2\delta_{ij}$, $\{\gamma_i, \gamma_j'\} = 0$. These Majorana operators arise from complex fermion's creation (denoted $c_j^\dagger$) and annihilation operators (denoted c_j) via $\gamma_j = c_j + c_j^\dagger$ and $\gamma_j' = \mathrm{i}(c_j - c_j^\dagger)$. Analogous to Pauli stabilizer codes being defined based on the Pauli group [50], Majorana stabilizer codes are defined based on the group of Majorana operators [49].

To handle the group multiplication structure more systematically, we consider a $2n$-dimensional vector space $\mathbb{F}_2^{2n}$, where $\mathbb{F}_2$ is the binary field $\{0,1\}$. We then construct an isomorphism $\phi : \mathrm{Maj}(2n) \to \mathbb{F}_2^{2n} \otimes \{\pm 1, \pm i\}$, mapping each element Γ to a binary vector $\vec{\gamma} \in \mathbb{F}_2^{2N}$ and a phase η,

$$\phi(\Gamma) = (\vec{\gamma}, \eta) \quad \text{where} \quad \vec{\gamma} = (\alpha_1, \alpha_1', \alpha_2, \alpha_2', \ldots, \alpha_n, \alpha_n')^T. \tag{S1}$$

To ensure that $\mathrm{Maj}(2n) \cong \mathbb{F}_2^{2n} \otimes \{\pm 1, \pm i\}$, we define $(\vec{\gamma}_1, \eta_1) \cdot (\vec{\gamma}_2, \eta_2) \equiv (\vec{\gamma}_1 \oplus \vec{\gamma}_2, \eta_1 \eta_2 f(\vec{\gamma}_1, \vec{\gamma}_2))$, where $\vec{\gamma}_1 \oplus \vec{\gamma}_2$ is the standard component-wise addition modulo 2 on $\mathbb{F}_2^{2n}$, equivalent to the XOR operation, and the factor $f(\vec{\gamma}_1, \vec{\gamma}_2) \in \{1, -1\}$, termed the sign function, encodes a sign arising from the inherent anticommutation relations of the Majorana operators. Based on a common operator ordering convention, we define the sign function as $f \equiv (-1)^{\sum_k [\vec{\gamma}_2]_k m_k}$, where m_k counts the number of nonzero elements in $\{[\vec{\gamma}_1]_k, [\vec{\gamma}_1]_{k+1}, \ldots, [\vec{\gamma}_1]_{2n}\}$.

For any elements $\Gamma_1, \Gamma_2 \in \mathrm{Maj}(2n)$, they either commute or anticommute. Specifically, we have $\Gamma_1 \Gamma_2 = \Gamma_2 \Gamma_1 (-1)^{\mathrm{p}(\vec{\gamma}_1)\mathrm{p}(\vec{\gamma}_2) \oplus \vec{\gamma}_1 \cdot \vec{\gamma}_2}$ [49], where $\vec{\gamma}_j$ represents the vector mapped from Γ_j, and $\mathrm{p}(\vec{\gamma})$ is a parity function denoting the parity of the number of nonzero entries in the vector $\vec{\gamma}$: 0 indicates an even parity and 1 indicates an odd parity. We see that two even-weight Majorana strings commute if their overlap has even weight and anticommute if their overlap has odd weight. One thus can encode logical Majorana operators as even-weight physical Majorana

operator strings. However, if two logical Majorana operators are located in different blocks, which have no overlap, then odd-weight strings must be considered, as they anticommute with each other.

The Majorana stabilizer code is defined by a Majorana stabilizer group $\mathcal{S}_{\mathrm{maj}} = \langle g_1, \ldots, g_m \rangle$ generated by independent Majorana operator strings $g_j \in \mathrm{Maj}(2n)$ with $j = 1, \ldots, m$. Each element in the group commutes with any other elements in it. The code space $\mathbb{H}_L$ is a subspace spanned by all state vectors that are simultaneous eigenstates of all stabilizer operators corresponding to eigenvalue of 1 [50]. The check matrix is defined as

$$H = \begin{pmatrix} \vec{g}_1^T \\ \vdots \\ \vec{g}_m^T \end{pmatrix}, \tag{S2}$$

where the vector $\vec{g}_j$ with $j = 1, \ldots, m$ corresponds to a stabilizer generator g_j. To preserve the parity of a physical fermion system, we suppose that all stabilizers have even weight. In this case, H is a parity-check matrix of a self-orthogonal linear code, satisfying $HH^T = 0$. Logical operators correspond to operators that commute with all stabilizers but are not in the stabilizer group. Mathematically, a set of all logical operators $\mathcal{L} = \mathcal{C}(\mathcal{S}) \setminus \mathcal{S}$, where $\mathcal{C}(\mathcal{S})$ denotes the centralizer whose elements commute with all elements in $\mathcal{S}$. The code distance d is determined by the minimum weight of all logical operators [49, 50].

We can use self-dual CSS codes on qubits to construct Majorana stabilizer codes [15, 49]. Let A be the parity-check matrix of a self-dual CSS code with parameters $[[n, k, d]]$, satisfying $AA^T = 0$ and ensuring the code space contains its dual space, which means dual-containing [44]. The direct sum $H = A \oplus A$ then forms a Majorana stabilizer code's check matrix. As shown in Fig. S1(a), we use Tanner graph which has mirror reflection symmetry to represent a Majorana stabilizer code, where the blue and green squares correspond to γ-type stabilizers and γ'-type stabilizers, respectively. When a self-dual CSS code has parameters $[[n, k, d]]$, the resulting Majorana stabilizer code has parameters $[[n, k, d]]_\mathrm{f}$ [49]. If the self-dual CSS code is a quantum LDPC code, the corresponding Majorana stabilizer code is also an Majorana LDPC code. Consequently, the problem of finding Majorana LDPC codes can be reduced to the problem of finding self-dual CSS codes. The odd-weight logical operators from an Majorana LDPC code can be used to form logical Majorana modes in our fermionic LDPC code.

Examples of self-dual CSS codes

In this subsection, we will review four types of self-dual CSS codes: Kitaev Majorana code [48], bicycle code [44] (Fig. S1(b)), Euclidean geometry codes on finite fields [45, 46] (Fig. S1(c)), and projective geometry codes on finite fields [47] (Fig. S1(d)). We will use the latter three to construct fermionic LDPC code.
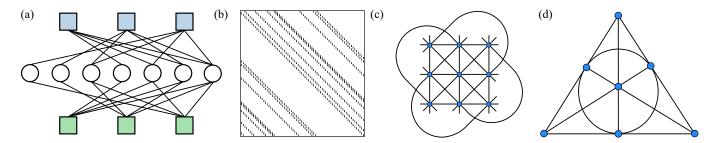

FIG. S1. (a) The Tanner graph description of the Steane code. The circles, blue and green squares represent the fermions, the γ-type stabilizers, and the γ'-type stabilizers, respectively. (b) Illustration of a circulant matrix used to construct bicycle codes. (c) Illustration of the Eucildean geometry when $(m, q) = (2, 3)$. In this special case, the lines are the straight lines on a torus with different directions. (d) The Fano plane, which is the simplest example of a finite projective plane. We will use many planes like this in code construction from finite projective geometry.

Kitaev Majorana code

According to Ref. [48], any $[[n, k, d]]$ qubit stabilizer code defined by a qubit stabilizer group $\mathcal{S}$ can be mapped to a $[[2n, k, 2d]]_\mathrm{f}$ Majorana stabilizer code defined by $\mathcal{S}_{\mathrm{maj}}$. For each qubit j, four Majorana operators γ_j^g, $\gamma_j^{x,y,z}$ are

introduced so that the Majorana stabilizer code is defined by $4n$ Majorana operators. The Pauli operators of each qubit j in the generators of $\mathcal{S}$ are mapped to the product of these Majorana operators by

$$
\begin{cases}
\sigma_j^x \to \mathrm{i}\gamma_j^x \gamma_j^g \\
\sigma_j^y \to \mathrm{i}\gamma_j^y \gamma_j^g \\
\sigma_j^z \to \mathrm{i}\gamma_j^z \gamma_j^g
\end{cases} , \tag{S3}
$$

forming generators in $\mathcal{S}_{\mathrm{maj}}$. In addition, n gauge stabilizers $G_j = \gamma_j^x \gamma_j^y \gamma_j^z \gamma_j^g$ are added into $\mathcal{S}_{\mathrm{maj}}$. The k logical operators of the Majorana stabilizer code are given by replacing the Pauli operators in the original qubit code with the Majorana modes. Specifically, consider a $[[n, k, d]]$ qubit CSS code with an X check matrix H_X and Z check matrix H_Z satisfying $H_X H_Z^T = 0$. The corresponding Majorana stabilizer code has the check matrix

$$
H = \begin{pmatrix} H_X & H_X & 0 & 0 \\ H_Z & 0 & 0 & H_Z \\ I_n & I_n & I_n & I_n \end{pmatrix}, \tag{S4}
$$

where I_n is an $n \times n$ identity matrix. In this case, the group of Majorana operator reads $\mathrm{Maj}(2n) \equiv \{\Gamma = \eta \prod_{j=1}^{n} (\gamma_j^g)^{\alpha_{j,1}} (\gamma_j^x)^{\alpha_{j,2}} (\gamma_j^y)^{\alpha_{j,3}} (\gamma_j^z)^{\alpha_{j,4}} | \alpha_{j,1}, \alpha_{j,2}, \alpha_{j,3}, \alpha_{j,4} \in \{0, 1\}, \eta \in \{\pm 1, \pm i\}\}$. The exponent factors of a Majorana stabilizer generator in $\mathrm{Maj}(2n)$ correspond to each row in H. Since there exists a stabilizer corresponding to a vector of $(1, 1, \ldots, 1)$, all logical operaotors must have even weight so as to commute with this stabilizer. In other words, one cannot find odd-weight logical operators in the Kitaev Majorana stabilizer code.

Bicycle code

In 2004, Mackay et al. proposed a so-called bicycle code in Ref. [44], which is a self-dual CSS code with two check matrices H_X and H_Z that satisfy $H_X = H_Z$ and $H_X H_X^T = 0$, thereby enabling the construction of Majorana LDPC codes. The check matrix of the bicycle code on $2n$ qubits is defined by

$$
H_X = H_Y = \begin{bmatrix} C, C^T \end{bmatrix}, \tag{S5}
$$

where C is an $n \times n$ cyclic matrix; each row of a cyclic matrix is obtained by right-cyclic shifting of the previous row, as illustrated in Fig. S1(b). Formally, let S be an $n \times n$ matrix defined by $S_{i,j} = 1$ $(1 \le i, j \le n)$ if $j = (i + 1) \bmod n$ and $S_{i,j} = 1$ otherwise, which reads

$$
S = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 1 \\ 1 & 0 & 0 & \ldots & 0 \end{pmatrix}. \tag{S6}
$$

The cyclic matrix C with weight r is defined by $C = \sum_{i=1}^{r} S^{a_i}$, where $1 \le a_i \le n$. The code is a self-dual CSS code since $H_X H_X^T = CC^T + C^T C = 0$.

As a demonstration, we construct a $[[100, 20, 7]]_{\mathrm{f}}$ bicycle Majorana code via a blind search. The same approach can be applied to search for other codes accordingly. In addition, we find that the other three types of codes proposed by Mackay et al. in Ref. [44] are not suitable for the fermionic LDPC code since one cannot find odd-weight logical Majorana operators similar to the Kitaev Majorana code.

Finite Euclidean geometry codes

In 2008, Aly proposed a class of self-dual qubit CSS codes based on the incidence matrix of finite Euclidean geometry [46]. However, we find that Aly's finite Euclidean geometry codes fail to produce odd-weight logical Majorana operators. We adopt the revised verion of finite Euclidean geometry codes proposed by Cao et al. in Ref. [45] to construct our fermionic LDPC code that contains odd-weight logical Majorana operators.

Here, we will follow Ref. [45] to briefly review the generalized finite Euclidean geometry code. For positive integers (m, q), a Euclidean geometry $\mathrm{EG}(m, q)$ consists of q^m points, each represented by an m-tuple. Here, m denotes the

spatial dimension, while q represents the number of points per dimension. As shown in Fig. S1(c), EG(2, 3) has two dimensions with three points per dimension, forming the Euclidean space through their Cartesian product. Consider $q = p^s$ for some prime p and positive integer s, with tuple elements drawn from the Galois field $\mathbb{F}_q$. First, we describe point representation before defining lines. For any prime p and positive integer s, there exists α satisfying $\alpha^{p^s} - \alpha = 0$, where $\mathbb{F}_{p^s} = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{p^s-2}\}$ [71]. For any positive integer m, $\mathbb{F}_{q^m}$ constitutes an extension field over $\mathbb{F}_q$, with elements expressible as $e = a_0 + a_1\alpha + \cdots + a_{m-1}\alpha^{m-1}$ ($a_i \in \mathbb{F}_q$). As illustrated in Fig. S1(c), each line in EG(2, 3) contains exactly three points. Thus, a bijection exists between $\mathbb{F}_{q^m}$ and m-tuples in EG(m, q). A line in EG(m, q) (or $\mathbb{F}_{q^m}$) is defined as the set containing exactly q points: $\{e_j + \beta e_k \,|\, \beta \in \mathbb{F}_q\}$, where $e_j, e_k \in \mathbb{F}_{q^m}$ and $e_k \neq 0$ [71]. We now enumerate and classify these lines. There are q^m choices for e_j and $q^m - 1$ for e_k. Since each line contains exactly q points, q distinct e_k choices yield identical lines. Additionally, $\{e_j + \alpha\beta e_k\}$ with $\beta \in \mathbb{F}_q \setminus \{0\}$ produces identical line sets. Consequently, the total number of lines is $\frac{q^m(q^m-1)}{q(q-1)}$, while the number of lines excluding the origin point is $\frac{(q^{m-1}-1)(q^m-1)}{q-1}$ [45, 71].

For a line $L = \{e_{l_0}, e_{l_1}, \ldots, e_{l_{q-1}}\}$, translation via multiplication by a nonzero $e \in \mathbb{F}_{q^m}$ yields $eL = \{ee_{l_0}, ee_{l_1}, \ldots, ee_{l_{q-1}}\}$. The $q^m - 1$ lines $\{L, eL, e^2L, \ldots, e^{q^m-2}L\}$ form an equivalence class. There exist $J = \frac{q^{m-1}-1}{q-1}$ such equivalence classes [46]. $J(q^m - 1) \times (q^m - 1)$ binary check matrices H_j ($1 \leq j \leq J$) are then constructed with rows corresponding to lines in the j-th equivalence class (each row is a binary vector indicating inclusion of points from the origin-excluding line) [45]. The full check matrix is then defined as

$$H_X = H_Z = \left(H_1^T, \ldots, H_J^T, H_1, \ldots, H_J\right). \tag{S7}$$

Since H_j with $j = 1, \ldots, J$ are cyclic matrices, H_X satisfies $H_X H_X^T = 0$ and thus gives rise to a self-dual CSS code.

Finite projective plane code

The finite projective plane code is a class of self-dual CSS codes constructed by Farinholt, with the help of the finite projective plane [79]. A projective plane PG(2, q) with $q = 2^s$ consists of a set of points and lines that satisfy: 1. Any two distinct points determine a unique line; 2. Any two distinct lines intersect at a unique point; 3. There exist four points, no three of which are collinear; 4. Each line contains $q + 1$ points; 5. Each point lies on $q + 1$ lines; 6. There are exactly $q^2 + q + 1$ points and lines [79].

We depict PG(2, 2) as an example in Fig S1(d). The points can be represented by the equivalence classes of $[x, y, z] \equiv \{[cx, cy, cz] | [x, y, z] \neq [0, 0, 0], \; c \in \mathbb{F} \setminus \{0\}\}$. We note that there are $(q^3 - 1)/(q - 1) = q^2 + q + 1$ such equivalence classes, which correspond to the points in the projective plane. The lines can be represented by the equivalence classes $(a, b, c) \equiv \{(\lambda a, \lambda b, \lambda c) | (a, b, c) \neq [0, 0, 0], \; \lambda \in \mathbb{F} \setminus \{0\}\}$, such that $ax + by + cz = 0$. Each line contains $q + 1$ points, and there are $q^2 + q + 1$ such lines.

The incidence matrix of the projective plane is a binary matrix B with its entry $B_{i,j} = 1$ if point i lies on line j, and 0 otherwise. The check matrix of this self-dual CSS code is then defined as

$$H_X = H_Z = \begin{pmatrix} B & \mathbf{1} \end{pmatrix}, \tag{S8}$$

where $\mathbf{1}$ is a column vector of ones. We note that $HH^T = BB^T + \mathbf{1}\mathbf{1}^T = 0$, because the overlap between any two rows of B is odd ($q + 1$ or 1), and thus BB^T is a $(q^2 + q + 1) \times (q^2 + q + 1)$ binary matrix with all entries being one.

Extracting odd-weight logical Majorana operators

We now describe how to identify odd-weight fermionic logical operators from the Majorana check matrix H constructed from check matrices $H_X = H_Z$ of a self-dual $[[n, k_q, d]]$ CSS code, that is,

$$H = \begin{pmatrix} H_\gamma & 0 \\ 0 & H_{\gamma'} \end{pmatrix} \tag{S9}$$

where $H_\gamma = H_{\gamma'} = H_X = H_Z$. H is an $(n - k_q) \times 2n$ binary matrix with all row vectors linearly independent such that $\text{rank}(H) = n - k_q$. All logical operators correspond to vectors in $\ker(H)/\text{im}(H)$. Let $S_L = \{\vec{v}_j | j = 1, \ldots, 2k_q\}$ be a basis for $\ker A/\text{im}(A)$, where the weight of $\vec{v}_j$ can be either even or odd [81]. An orthonormal set $\{\vec{\gamma}_j\}_{j=1}^{2k_f}$ with $k_f \leq k_q$ consisting of the odd-weight vectors will be constructed from S_L. We show the procedure to construct the

set constituting a basis for our code space of fermions in the following, generalizing the method in Ref. [55] to the $\mathbb{F}_2$ case.

If vectors in S_{L} all have even weight, then there do not exist odd-weight logical operators, since the addition of any two even-weight binary vectors yields an even-weight vector. Otherwise, there exists at least one odd-weight vector, say, $\vec{v}_1$. We choose $\vec{\gamma}_1 = \vec{v}_1$ as the first element in the odd-weight vector set and update the remaining $2k_{\mathrm{q}} - 1$ vectors in S_{L} as $\vec{v}_j' \to \vec{v}_j - (\vec{v}_j \cdot \vec{v}_1)\vec{v}_1$. As a result, $\vec{v}_1$ is orthogonal to all the remaining vectors $\vec{v}_j'$ ($j \geq 2$). We repeat this process until the remaining vectors are all of even weight, leading to an orthonormal set consisting of odd-weight vectors. The algorithm is summarized in Algorithm 1.

Algorithm 1: Orthonormalization of $\{\vec{v}_j\}_{j=1}^{2k_{\mathrm{q}}}$ over $\mathbb{F}_2$

Input: A set of binary vectors $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_{2k_{\mathrm{q}}}\}$ over $\mathbb{F}_2$

Output: The orthonormal basis of odd weight operators $\{\vec{\gamma}_j\}_{j=1}^{2k_{\mathrm{f}}}$

1 **Function** Gram-Schmidt orthonormalization over $\mathbb{F}_2(\{\vec{v}_1, \ldots, \vec{v}_{2k_{\mathrm{q}}}\})$:

2     $\mathcal{B} \leftarrow \{\vec{v}_1, \ldots, \vec{v}_{2k_{\mathrm{q}}}\}$;   // Copy input vectors

3     $\mathcal{O} \leftarrow \emptyset$;   // Initialize orthogonal basis set

4   **while** $\exists \vec{b}_i \in \mathcal{B}$ such that $\vec{b}_i \cdot \vec{b}_i = 1$ **do**

5        $\mathcal{O} \leftarrow \mathcal{O} \cup \{\vec{b}_i\}$;

6        $\mathcal{B} \leftarrow \mathcal{B} \setminus \{\vec{b}_i\}$;

7   **for** E **do**

8   a

9   **end**

10   ch$\vec{b} \in \mathcal{B}\ \vec{b} \leftarrow \vec{b} - (\vec{b} \cdot \vec{b}_i)\vec{b}_i$;

11   **end**

12   **return** $\mathcal{O}$;

The following theorem provides a necessary and sufficient condition for the existence of an orthonormal basis for $\mathrm{span}(S_{\mathrm{L}})$, i.e., an orthonormal basis consisting of odd-weight vectors.

Theorem .1. *Let G be a $2k_q \times 2k_q$ binary matrix defined by $G_{ij} = \vec{v}_i \cdot \vec{v}_j$ with $1 \leq i, j \leq 2k_q$. An orthonormal basis for $\mathrm{span}(S_L)$ exists if and only if there exists an invertible matrix P such that*

$$PGP^T = I_{2k_q}. \tag{S10}$$

Proof. If there exists an orthonormal basis $\{\vec{\gamma}_1, \ldots, \vec{\gamma}_{2k_q}\}$ for the span, then we must can write each vector in the basis as a linear combination of the vectors $\vec{v}_1, \ldots, \vec{v}_{2k_q}$, that is, there exists a $2k_q \times 2k_q$ matrix P such that

$$\begin{pmatrix} \vec{\gamma}_1 \\ \vdots \\ \vec{\gamma}_{2k_q} \end{pmatrix} = P \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_{2k_q} \end{pmatrix}. \tag{S11}$$

Since $\{\vec{\gamma}_1, \ldots, \vec{\gamma}_{2k_q}\}$ is orthonormal, we have $\vec{\gamma}_i \cdot \vec{\gamma}_j = \delta_{i,j} = \sum_{i',j'} P_{ii'} P_{jj'} \vec{v}_{i'} \cdot \vec{v}_{j'} = [PGP^T]_{ij}$, that is, $PGP^T = I_{2k_q}$.

Conversely, if there exists an invertible matrix P such that $PGP^T = I_{2k_q}$, we define a new basis $\{\vec{\gamma}_1, \ldots, \vec{\gamma}_{2k_q}\}$ by

$$\begin{pmatrix} \vec{\gamma}_1 \\ \vdots \\ \vec{\gamma}_{2k_q} \end{pmatrix} = P \begin{pmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_{2k_q} \end{pmatrix}, \tag{S12}$$

which clearly gives $\vec{\gamma}_i \cdot \vec{\gamma}_j = \delta_{i,j}$. Thus, $\{\vec{\gamma}_1, \ldots, \vec{\gamma}_{2k_q}\}$ forms an orthonormal basis. $\qquad\square$

We numerically calculate the number of odd-weight logical Majorana operators forming k_{f} logical complex fermions for the check matrix H constructed based on bicycle codes, finite Euclidean geometry codes, and finite projective plane codes, and compare them with the number of logical qubits k_{q} for the corresponding self-dual CSS codes in Table I. We see that for most cases, the algorithm can efficiently find the same number of logical complex fermions as that of logical qubits, thus providing a high encoding rate. In Table I, only for the EG$(3, 4)$ code, k_{f} is smaller than k_{q} by two. In fact, when we slightly modify the algorithm, we can find the same number of logical complex fermions, that is, $k_{\mathrm{f}} = k_{\mathrm{q}}$.

TABLE I. Comparison between the number of logical qubits and the number of logical complex fermions for different codes. d denotes the distance of the self-dual CSS codes.

name	Bicycle code	EG(2,4)	EG(2,8)	EG(3,2)	EG(3,4)	PG(2,4)	PG(2,8)	PG(2,16)	PG(2,32)
n	100	30	126	42	630	16	64	256	1024
k_q	20	2	38	30	506	10	44	190	812
k_f	20	2	38	30	504	10	44	190	812
d	7	5	9	2	5	3	5	9	17

S-2. FERMIONIC STATE TELEPORTATION

We now validate the quantum state teleportation circuit shown in Fig. 3(a). Suppose that our fermionic LDPC code memory A encodes k_f logical complex fermion modes, and we want to transfer the logical information of the lth logical mode to a processor B, which is initialized in $|\overline{0}\rangle_B$. Let the logical state of A be $|\overline{\psi}\rangle_A$, and the state of the entire system $|\overline{\Psi}\rangle$ can be written as

$$|\overline{\Psi}\rangle = |\overline{\psi}\rangle_A |\overline{0}\rangle_B = \left(\overline{f} + \overline{c}_l^\dagger \overline{g}\right) |\mathrm{vac}\rangle, \tag{S13}$$

where $\overline{c}_l^\dagger$ denotes the logical fermionic creation operator of the lth logical mode of code A, $\overline{h} = h(\overline{c}_1^\dagger, \ldots, \overline{c}_{l-1}^\dagger, \overline{c}_{l+1}^\dagger, \ldots, \overline{c}_{k_\mathrm{f}}^\dagger)$ with $h = f, g$ represents a polynomial of the logical operators except the jth operator, and $|\mathrm{vac}\rangle$ is the vacuum state. We prove that the logical state of the system will be $|\overline{\Psi}'\rangle = \left(\overline{f} + \overline{c}_B^\dagger \overline{g}\right) |\mathrm{vac}\rangle$ after implementing the quantum state teleportation circuit, where $\overline{c}_B^\dagger$ is the encoded logical creation operator of code B.

Let $\overline{c}_l^\dagger \equiv \frac{1}{2}\left(\overline{\gamma}_l + \mathrm{i}\overline{\gamma}_l'\right)$ and $\overline{c}_B^\dagger \equiv \frac{1}{2}\left(\overline{\gamma}_B + \mathrm{i}\overline{\gamma}_B'\right)$, the first step of the circuit is to perform the projective measurement of the joint parity operator $\mathrm{i}\overline{\gamma}_j \overline{\gamma}_B$, resulting in the state proportional to

$$\begin{aligned}
|\overline{\Psi}_1\rangle &\sim \frac{1 \pm \mathrm{i}\overline{\gamma}_j \overline{\gamma}_B}{2} |\overline{\Psi}\rangle \\
&\sim \left(\overline{f} + \overline{c}_j^\dagger \overline{g} \pm \mathrm{i}\overline{\gamma}_j \overline{\gamma}_B \overline{f} \pm \mathrm{i}\overline{\gamma}_j \overline{\gamma}_B \overline{c}_j^\dagger \overline{g}\right) |\mathrm{vac}\rangle \\
&\sim \left(\overline{f} + \overline{c}_j^\dagger \overline{g} \pm \mathrm{i}\overline{\gamma}_j \overline{\gamma}_B \overline{f} \mp \mathrm{i}\overline{\gamma}_B \overline{g}\right) |\mathrm{vac}\rangle,
\end{aligned} \tag{S14}$$

where the last equation is due to the fact that $\overline{\gamma}_j = \overline{c}_j^\dagger + \overline{c}_j$. If the measurement result is 1, we proceed; otherwise, a Z gate $\hat{Z} = \exp\left(\mathrm{i}\pi \overline{c}_B^\dagger \overline{c}_B\right)$ is applied to the processor to derive the state

$$\begin{aligned}
|\overline{\Psi}_2\rangle &= \exp\left(\mathrm{i}\pi \overline{c}_B^\dagger \overline{c}_B\right) |\overline{\Psi}_1\rangle \\
&= \left(1 - 2\overline{c}_B^\dagger \overline{c}_B\right) |\overline{\Psi}_1\rangle \\
&\sim \left(\overline{f} + \overline{c}_j^\dagger \overline{g} + \mathrm{i}\overline{\gamma}_j \overline{\gamma}_B \overline{f} - \mathrm{i}\overline{\gamma}_B \overline{g}\right) |\mathrm{vac}\rangle.
\end{aligned} \tag{S15}$$

We now perform the projective measurement of the particle number operator $\overline{c}_j^\dagger \overline{c}_j$, and the state becomes

$$|\overline{\Psi}_3\rangle \sim \begin{cases} \left(1 - \overline{c}_j^\dagger \overline{c}_j\right) |\overline{\Psi}_2\rangle \\ \overline{c}_j^\dagger \overline{c}_j |\overline{\Psi}_2\rangle \end{cases} = \begin{cases} \left(\overline{f} - \mathrm{i}\overline{\gamma}_B \overline{g}\right) |\mathrm{vac}\rangle \\ \left(\mathrm{i}\overline{\gamma}_j \overline{\gamma}_B \overline{f} + \overline{c}_j^\dagger \overline{g}\right) |\mathrm{vac}\rangle \end{cases},$$

for the measurement result of 0 and 1, respectively. If the measurement result is 0, we simply proceed; otherwise, we apply the joint operator $\mathrm{i}\overline{\gamma}_j \overline{\gamma}_B$ to get the state $|\overline{\Psi}_4\rangle = \left(\overline{f} - \mathrm{i}\overline{\gamma}_B \overline{g}\right) |\mathrm{vac}\rangle = \left(\overline{f} - \mathrm{i}\overline{c}_B^\dagger \overline{g}\right) |\mathrm{vac}\rangle$. After that, we apply a phase rotation gate $\hat{S} = \exp\left(\mathrm{i}\frac{\pi}{2}\overline{c}_B^\dagger \overline{c}_B\right)$ to achieve the target state $|\overline{\Psi}'\rangle = \left(\overline{f} + \overline{c}_B^\dagger \overline{g}\right) |\mathrm{vac}\rangle$.

For two fermion gates, we need to transfer the information from two logical fermionic sites in the memory to two processors, and apply corresponding logical gates on the processors.

S-3. FERMIONIC LATTICE SURGERY

In this section, we will present a detailed implementation of fermionic lattice surgery, analyze its overhead, and demonstrate the fault-tolerance of this procedure. Fermionic lattice surgery provides a fault-tolerant method for measuring the joint operator $i\overline{\gamma}_A\overline{\gamma}_B$, where $\overline{\gamma}_A$ and $\overline{\gamma}_B$ are the fermionic logical operators of memory A and processor B, respectively.

The procedure proceeds by introducing ancilla Majorana operators between the two codes, modifying the stabilizers associated with $\overline{\gamma}_A$ and $\overline{\gamma}_B$, and adding new stabilizers that merge A and B into a larger fermionic LDPC code $\mathcal{C}_{\text{merged}}$. We prove that the joint operator $i\overline{\gamma}_A\overline{\gamma}_B$ belongs to the stabilizer group $\mathcal{S}_{\text{merged}}$ of $\mathcal{C}_{\text{merged}}$. Consequently, it can be measured fault-tolerantly by collecting the outcomes of the stabilizer measurements in $\mathcal{C}_{\text{merged}}$.

For clarity, we use the following terminology: additional Majorana operators are termed ancilla Majorana operators, stabilizers inherited from the original codes and modified during the merging are called modified stabilizers, newly introduced stabilizers acting solely on ancilla Majorana operators are termed gauge stabilizers, and stabilizers involving Majorana operators from both codes A and B are referred to as measurement stabilizers.

We first examine the first method, where the two logical operators have equal weight, i.e., $|\overline{\gamma}_A| = |\overline{\gamma}_B|$ with $|\ldots|$ denoting the weight of a Majorana operator string. We then present the second method which can handle both cases: $|\overline{\gamma}_A| = |\overline{\gamma}_B|$ and $|\overline{\gamma}_A| \neq |\overline{\gamma}_B|$.

Method 1

Since logical operators in our fermionic LDPC code all have odd weight, for any $\overline{\gamma}_A$ in a fermionic LDPC code memory A, we can always select a triangular color code B [15, 49] encoding a single pair of logicals $(\overline{\gamma}_B, \overline{\gamma}'_B)$, such that $|\overline{\gamma}_A| = |\overline{\gamma}_B| = |\overline{\gamma}'_B| = d$, as depicted in Fig. S2. Method 1 as detailed in the following is applicable to the lattice surgery between a fermionic LDPC code and a fermionic color code with $|\overline{\gamma}_A| = |\overline{\gamma}_B|$.

We align the support of $\overline{\gamma}_A$ and $\overline{\gamma}_B$ in a pair of parallel lines, each consisting of d points, as shown in Fig. 3, and indexing them as $\{\gamma_{A,i}\}_{i=1}^d$ and $\{\gamma_{B,i}\}_{i=1}^d$ from left to right, respectively. We also index the checks supported on A (B) that are associated with $\overline{\gamma}_A$ $(\overline{\gamma}_B)$ as $\{c_{A,i}\}_{i=1}^{N_A}$ $(\{c_{B,i}\}_{i=1}^{N_B})$, where N_A, $N_B = d - 1$ are the number of stabilizers associated with $\overline{\gamma}_A$ and $\overline{\gamma}_B$, respectively. As shown in Fig. S2, each c_B forms a plaquette on the boundary of the color code lattice. We now introduce ancilla Majorana modes, modify the original stabilizers, and introduce new stabilizer checks step by step.
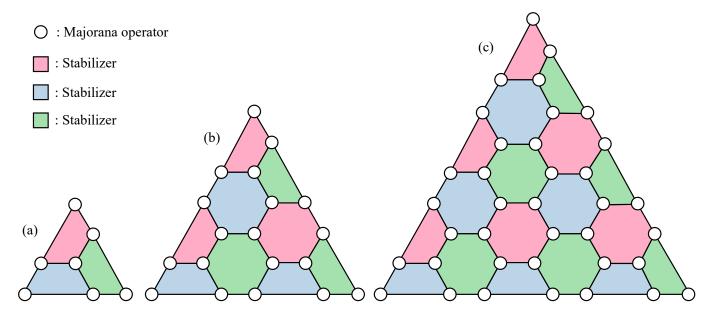

FIG. S2. Triangular fermionic color codes for code distance (a) $d = 3$ (fermionic Steane code), (b) $d = 5$, and (c) $d = 7$. The logical operator is the product of the d Majorana modes on the bottom boundary, and the stabilizers acting on this logical are represented by the $d - 1$ blue and green plaquettes. From left to the right, we index the $d - 1$ plaquettes in order. Any triangular color code with odd code distances d can be similarly constructed [49].

We first modify all the stabilizers $c_{A,i}$ and $c_{B,i}$ by introducing ancilla Majorana operators. Specifically, consider each

$$(c_{A,i})_{\overline{A}} \sim \gamma_{A,j_1}\gamma_{A,j_2}\ldots\gamma_{A,j_{m_i}}, \ 1 \leq j_1, \cdots \leq j_{m_i} \leq d, \tag{S16}$$

where $\sim$ denotes equality up to a phase, $\overline{A}$ denotes the support of $\overline{\gamma}_A$, $(\cdots)_{\overline{A}}$ is the operator constrained on $\overline{A}$, and $m_i = |(c_{A,i})_{\overline{A}}|$ is an even number, ensured by the commutativity between $\overline{\gamma}_A$ and $c_{A,i}$. We introduce m_i ancilla Majorana operators $\left\{\gamma_{a,i,j_1}, \gamma'_{a,i,j_2}, \ldots, \gamma_{a,i,j_{m_i-1}}, \gamma'_{a,i,j_{m_i}}\right\}$, where the subscript a indicate that these modes are associated with the stabilizer of A, i is the index of the stabilizer $c_{A,i}$, and $j_1, j_2, \ldots, j_{m_i}$ correspond to the positions of the Majorana operators in $c_{A,i}$ in the support of $\overline{\gamma}_A$. The last two subscripts are used to index the order of the ancilla modes, corresponding to the second subscript in $\gamma_{a,j}$ and $\gamma'_{a,j}$ with $1 \leq j \leq d_A - 1$ as described in the main text. We modify the stabilizer $c_{A,i}$ to

$$c'_{A,i} \sim c_{A,i}\gamma_{a,i,j_1}\gamma'_{a,i,j_2}\ldots\gamma_{a,i,j_{m_i-1}}\gamma'_{a,i,j_{m_i}}. \tag{S17}$$

Similarly, we introduce m_i ancilla Majorana operators $\left\{\gamma_{b,i,j_1}, \gamma'_{b,i,j_2}, \ldots, \gamma_{b,i,j_{m_i-1}}, \gamma'_{b,i,j_{m_i}}\right\}$ for code B. For each pair $(\gamma_{a,i,j_{2n-1}}, \gamma'_{a,i,j_{2n}})$, $1 \leq n \leq m_i/2$:

- If $j_{2n-1} + 1 = j_{2n}$, we introduce a modified stabilizer in code B as $c'_{B,i} \sim c_{B,i}\gamma_{b,i,j_{2n-1}}\gamma'_{b,i,j_{2n}}$, and a gauge stabilizer as $\gamma_{a,i,j_{2n-1}}\gamma'_{a,i,j_{2n}}\gamma_{b,i,j_{2n-1}}\gamma'_{b,i,j_{2n}}$.

- If $j_{2n-1} + 2 = j_{2n}$, this modified stabilizer is defined as $c'_{B,i} \sim c_{B,j_{2n-1}}c_{B,j_{2n}-1}\gamma_{b,i,j_{2n-1}}\gamma'_{b,i,j_{2n}}$, and the gauge stabilizer is $\gamma_{a,i,j_{2n-1}}\gamma'_{a,i,j_{2n}}\gamma_{b,i,j_{2n-1}}\gamma'_{b,i,j_{2n}}$.

- Otherwise, we add 4 additional stabilizers $\left\{\gamma_{b,i,j_{2n-1}+1}, \gamma'_{b,i,j_{2n}-1}, \gamma_{a,i,j_{2n-1}+1}, \gamma'_{a,i,j_{2n}-1}\right\}$, let the modified stabilizer be $c'_{B,i} \sim c_{B,j_{2n-1}}c_{B,j_{2n}-1}\gamma_{b,i,j_{2n-1}}\gamma'_{b,i,j_{2n-1}+1}\gamma_{b,i,j_{2n}-1}\gamma'_{b,i,j_{2n}}$, and introduce two gauge stabilizers as $\gamma_{a,i,j_{2n-1}}\gamma'_{a,i,j_{2n}}\gamma_{b,i,j_{2n-1}}\gamma'_{b,i,j_{2n}}$, $\gamma_{a,i,j_{2n-1}+1}\gamma'_{a,i,j_{2n}-1}\gamma_{b,i,j_{2n-1}+1}\gamma'_{b,i,j_{2n}-1}$.

We repeat this process for each $c_{A,i}$. We note that for those unused $c_{B,i}$ after iterating over all $c_{A,i}$, say the number of which is M_B. For each such $c_{B,m}$, where $1 \leq m \leq M_B$ represents the index of $c_{B,m}$, we further introduce four Majorana operators $\left\{\gamma_{b,N_A+m,i}, \gamma'_{b,N_A+m,i+1}, \gamma_{a,N_A+m,i}, \gamma'_{a,N_A+m,i+1}\right\}$, define a modified stabilizer $c'_{B,i}$ as $c_{B,i}\gamma_{b,N_A+m,i}\gamma'_{b,N_A+m,i+1}$, and add a gauge stabilizer as $\gamma_{b,N_A+m,i}\gamma'_{b,N_A+m,i+1}\gamma_{a,N_A+m,i}\gamma'_{a,N_A+m,i+1}$.

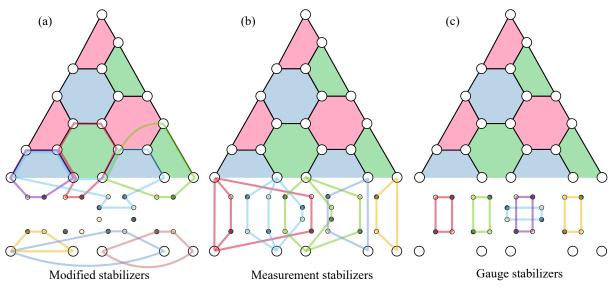After that, we define d measurement stabilizers $\{M_i\}_{i=1}^d$ as

$$M_i \sim \gamma_{A,i}\gamma_{B,i}\prod_n \gamma^{(\prime)}_{a,n,i}\gamma^{(\prime)}_{b,n,i}, \tag{S18}$$

where $\prod_n \gamma^{(\prime)}_{a,n,i}\gamma^{(\prime)}_{b,n,i}$ denotes the product of all ancilla Majorana operator (either of γ-type or γ'-type) at site i with n running over the subscripts of the ancilla modes associated with $\gamma_{A,i}$ and $\gamma_{B,i}$. In other words, the measurement stabilizer at site i consists of $\gamma_{A,i}$, $\gamma_{B,i}$ and all ancilla Majorana operators at this position. We note that $i\overline{\gamma}_A\overline{\gamma}_B$ is proportional to the product of all measurement stabilizers and the gauge stabilizers, since all $\gamma_{A,i}$ and $\gamma_{B,i}$ appear once and all ancilla Majorana operators appear twice in this product and thus cancel out. Figure S3 displays all modified, measurement, and gauge stabilizers for an illustrative case.

We show that this merged code is a well-defined fermionic LDPC code. First, all the new stabilizers have even weight by direct observation. We now analyze the weights of the stabilizers and show commutativity between them.

- The modified stabilizer $c'_{A,i}$ has weight $|c_{A,i}| + m_i$, which is even and remains bounded. It commutes with:

  - Any measurement stabilizer M_n, since their overlap is either empty or $\left\{\gamma_{A,n}, \gamma^{(\prime)}_{a,i,n}\right\}$,
  - All gauge stabilizers, since their overlap is either empty or $\left\{\gamma_{a,i,j_{2n-1}}, \gamma'_{a,i,j_{2n}}\right\}$ for some n,
  - All stabilizers and modified stabilizers in B, as their supports are disjoint.
  - All stabilizers in A, which is ensured by the construction of code A.

- The stabilizer $c'_{B,i}$ has a bounded even weight by definition, and commutes with measurement and gauge stabilizers by similar reasons for $c'_{A,i}$.

FIG. S3. Illustration of modified stabilizers in (a), measurement stabilizers in (b), and gauges stabilizers in (c) for a specific case of fermionic lattice surgery using method 1. Before merging, the original stabilizers include those of the fermionic LDPC code, the color code, and the stabilizers of ancilla Majorana operators (each stabilizer includes a pair of ancilla Majorana operators forming a complex fermion).

- The gauge stabilizers have weight of four by definition, and commute with measurement stabilizers since their overlap contains either zero or two Majorana modes.

- Each measurement stabilizer M_i has bounded weight, since the numbers of the modified stabilizers associated with $\gamma_{A,i}$ and $\gamma_{B,i}$ are bounded, and thus contribute a bounded number of ancilla modes to M_i.

Thus, code $\mathcal{C}_{\mathrm{merged}}$ is a well-defined fermionic LDPC code with mutually commuting stabilizers. Furthermore, the number of ancilla modes is bounded by $4(d-1) + \sum_{i=1}^{N_A} 4m_i$, resulting in a low overhead for the lattice surgery. We leave the analysis of capacity of fault-tolerance to the next section.

Method 2

In this subsection, we will introduce the second method for fermionic lattice surgery applicable when $|\overline{\gamma}_A| = |\overline{\gamma}_B|$ or $|\overline{\gamma}_A| \neq |\overline{\gamma}_B|$, and when code B is either a fermionic color code or a fermionic LDPC code. As an example depicted in Fig. S4(a), we align a weight-5 logical operator $\overline{\gamma}_A$ of fermionic LDPC code A and a weight-3 logical operator $\overline{\gamma}_B$ of fermionic color code B in two parallel lines, and index the Majorana modes in these two lines from top to bottom as $\{\gamma_{A,i}\}_{i=1}^{5}$ and $\{\gamma_{B,i}\}_{i=1}^{3}$, respectively.

We first modify the original stabilizers associated with $\overline{\gamma}_A$ and $\overline{\gamma}_B$. As shown in Fig. S4(a), there are $N_A = 4$ stabilizers for $\overline{\gamma}_A$ and $N_B = 2$ stabilizers for $\overline{\gamma}_B$, whose weights of overlap with $\overline{\gamma}_A$ and $\overline{\gamma}_B$ are denoted as $\{m_{A,i}\}_{i=1}^{N_A}$ and $\{m_{B,i}\}_{i=1}^{N_B}$, respectively. For each stabilizer $c_{A,i}$ and $c_{B,i}$, we introduce $m_{A,i}$ and $m_{B,i}$ ancilla Majorana operators $\{\gamma_{a,i,j_1}, \gamma'_{a,i,j_2}, \cdots, \gamma_{a,i,j_{m_{A,i}}-1}, \gamma'_{a,i,j_{m_{A,i}}}\}$ and $\{\gamma_{b,i,j_1}, \gamma'_{b,i,j_2}, \cdots, \gamma_{b,i,j_{m_{B,i}}-1}, \gamma'_{b,i,j_{m_{B,i}}}\}$, respectively, where j_k correspond to the vertical positions of the Majorana operators in $c_{A,i}$ or $c_{B,i}$. We modify $c_{A(B),i}$ to $c'_{A(B),i}$ by multiplying these ancilla Majorana operators as before, as shown in Fig. S4(b). The difference is that, in this case, the modified stabilizer $c'_{B,i}$ is constructed from a single stabilizer $c_{B,i}$, instead of the product of several stabilizers in the previous case.

Next, we define the measurement stabilizers $\{M_i\}_{i=1}^{D_{AB}}$, where $D_{AB} = (|\overline{\gamma}_A| + |\overline{\gamma}_B|)/2$. For the first $\min\{|\overline{\gamma}_A|, |\overline{\gamma}_B|\}$ measurement stabilizer M_i, we define M_i as the product of $\gamma_{A,i}$, $\gamma_{B,i}$, and all ancilla Majorana modes at site i, that
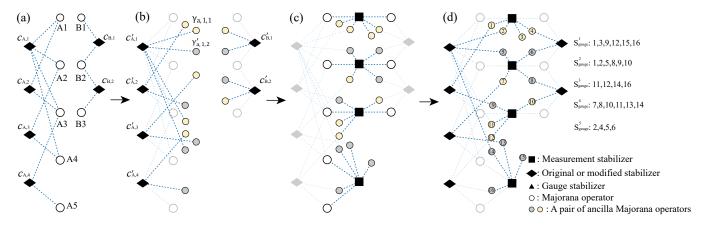
FIG. S4. Illustration of how to add ancilla Majorana modes and stabilizers for a merged code using method 2. Here we consider two logical operators in codes A and B, supported by five and three Majorana operators, respectively. Initially, we have two logical operators described by the diamond-shaped original stabilizers and circular Majorana operators. for code A in the support of $\overline{\gamma}_A$ and for code B in the support of $\overline{\gamma}_B$ (see (a)). Pairs of ancilla Majorana operators (yellow and gray circles) are added following the rules detailed in the text, and the modified stabilizer generators are introduced (see (b)). We then incorporate measurement stabilizer generators with an additional pair of ancilla Majorana operators to make their weights even (see (c)). In (d), we label all ancilla modes and construct a graph with vertices corresponding to stabilizers and ancilla Majorana modes, and edges defined between them accordingly, which corresponds to Fig. S5(a). The gauge stabilizer generators are also presented in Fig. S5(c).

is,

$$M_i \sim \gamma_{A,i}\gamma_{B,i} \prod_{n_a,n_b} \gamma^{(\prime)}_{a,n_a,i}\gamma^{(\prime)}_{b,n_b,i}, \tag{S19}$$

where $\prod_{n_a,n_b} \gamma^{(\prime)}_{a,n_a,i}\gamma^{(\prime)}_{b,n_b,i}$ denotes the product of all ancilla Majorana operators at site i (either of γ-type or γ'-type) with n_a and n_b running over the subscripts of the ancilla modes associated with $\gamma_{A,i}$ and $\gamma_{B,i}$, respectively. For the remaining $|(|\overline{\gamma}_A| - |\overline{\gamma}_B|)|/2$ measurement stabilizers $\{M_i\}_{i=\min\{|\overline{\gamma}_A|,|\overline{\gamma}_B|\}+1}^{D_{AB}}$, they are defined as the product of a pair of Majorana operators in $\{\gamma_{A,i}\}$ or $\{\gamma_{B,i}\}$, and the ancilla Majorana modes associated with them. For example, in Fig. S4(c), we define $M_4 \sim \gamma_{A,4}\gamma_{A,5}\gamma'_{a,1,4}\gamma'_{a,3,4}\gamma'_{a,4,5}$. We note that if any M_i has odd weight, we introduce an additional ancilla mode $\gamma_{p,i}$, and modify M_i to $M_i\gamma_{p,i}$ up to a phase factor to ensure even weight, as shown in the first and fourth measurement stabilizers in Fig. S4(c). The number of such additional ancilla modes is denoted as N_P, which must be even since the product of all measurement stabilizers is an even-weight operator.

Finally, we discuss how to identify gauge stabilizers from a graph theory perspective, as illustrated in Fig. S5. A graph is constructed as follows:

- The modified stabilizers (pink and cyan circles), measurement stabilizers (yellow circles) and ancilla Majorana operators (blue circles) are treated as vertices. An edge between a measurement stabilizer vertex and an ancilla mode vertex is added if the corresponding measurement stabilizer contains the corresponding ancilla Majorana operator (see Fig. S5(a)).

- If a modified stabilizer connect to more than one pair of ancilla modes, then we add more modified stabilizer vertices and adjust connections so that each modified stabilizer vertex connects to one pair of ancilla modes (see Fig. S5(b)).

- For each pair of ancilla Majorana operators that are introduced to ensure even-weight measurement stabilizers (e.g., node 3 and 15 in Fig. S5), we add a endpoint vertex between them and connect this vertex to the two ancilla mode vertices (see Fig. S5(b)).

- We refer to the vertices corresponding to measurement stabilizers as measurement vertices, the vertices corresponding to ancilla Majorana operators as ancilla vertices, and the modified stabilizers and the vertices added in the previous two steps as modified vertices.

Therefore, this graph contains D_{AB} measurement vertices, $\sum_{i=1}^{N_A} m_{A,i} + \sum_{i=1}^{N_B} m_{B,i} + N_P$ ancilla vertices, $\sum_{i=1}^{N_A} \frac{1}{2}m_{A,i} + \sum_{i=1}^{N_B} \frac{1}{2}m_{B,i} + \frac{N_P}{2}$ modified vertices, and $2(\sum_{i=1}^{N_A} m_{A,i} + \sum_{i=1}^{N_B} m_{B,i} + N_P)$ edges, since each ancilla vertex connects
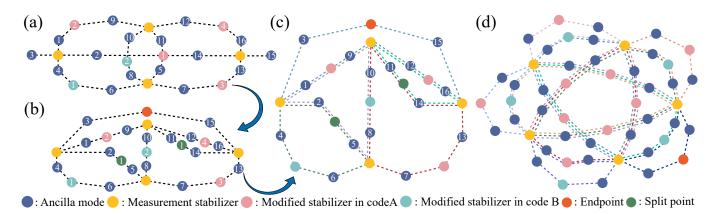
FIG. S5. Illustration of construction of gauge stabilizer generators in a general fermionic lattice surgery using method 2. (a) An initial graph. (b) A graph obtained by splitting modified stabilizers via adding more vertices, called split points, and by adding an endpoint that connects the ancilla Majorana operators introduced to ensure even-weight measurement stabilizers (node 3 and node 15). (c) Independent cycles, each represented by different colored closed paths. These figures correspond to the case shown in Fig. S4. (d) Independent cycles for the case of performing fermionic lattice surgery between PG(2, 8) and $d = 5$ fermionic color code.

two vertices. Since each measurement vertex has even degree, each modified vertex has degree of two, and any two vertices can be connected by some path, this graph is an undirected Eulerian graph with a single connected component, indicating that it contains

$$|E| - |V| + 1 = \sum_{i=1}^{N_A} m_{A,i}/2 + \sum_{i=1}^{N_B} m_{B,i}/2 + N_P/2 - D_{AB} + 1 \tag{S20}$$

independent cycles [80]. Here $|E|$ and $|V|$ denote the number of edges and vertices, respectively. The gauge stabilizer is then defined as the product of all ancilla Majorana operators on each independent cycle (see Fig. S5(c) and (d)).

The commutativity between measurement stabilizers and modified stabilizers is ensured by construction. We now verify that the gauge stabilizers are even-weight operators, and commute with measurement and modified stabilizers.

- Each path consists of a set of ancilla vertices, which are interlaced with measurement vertices and modified vertices. We note that each modified vertex connects exclusively a pair of ancilla vertices on a cycle and each ancilla vertices connect one modified vertex, as depicted in Fig. S5(c). Therefore, the number of ancilla vertices in each cycle is twice the number of modified vertices, ensuring that gauge stabilizers are even-weight operators.

- The number of overlapped ancilla vertices between a gauge stabilizer and a modified stabilizer is even, and thus each gauge stabilizer commutes with all modified stabilizers.

- The number of ancilla vertices shared by a gauge stabilizer and a measurement stabilizer is always even. This is because, within the cycle of a gauge stabilizer, traversing a measurement vertex requires entering and exiting through two distinct ancilla vertices, as shown in Fig. S5(c) and (d).

- The gauge stabilizers commute with each other. If two cycles intersect at a vertex $\gamma_{a(b),i,j_1}$, they must pass through the modified vertex associated with $\gamma_{a(b),i,j_1}$. Consequently, these cycles must also intersect at the other vertex $\gamma'_{a(b),i,j_2}$ associated with this modified vertex, resulting in an even overlap between the two gauge stabilizers.

We now demonstrate that the merged code $\mathcal{C}_{\mathrm{merged}}$ is a fermionic LDPC code with a low overhead. The number of ancilla fermions introduced is $\sum_i m_{A,i} + \sum_i m_{B,i} + N_P$, which is bounded by $\sum_i m_{A,i} + \sum_i m_{B,i} + D_{AB}$. This corresponds to a low-overhead construction, making this lattice surgery practical. The weight of measurement stabilizers is bounded due to the limited number of original stabilizers associated with each site. For the weight of gauge stabilizers, we numerically compute the maximum weight of the gauge stabilizers for 1000 randomly generated code pairs (A and B). Our results show that this weight scales as $[(|\bar{\gamma}_A| + |\bar{\gamma}_B|)/2]^{0.5}$, as illustrated in Fig. S6.

The fault-tolerant measurement of the joint logical operator $i\bar{\gamma}_A\bar{\gamma}_B$ is due to the fact that this joint operator becomes a stabilizer of the merged code. To show this, we need to demonstrate that there exists a subset of gauge
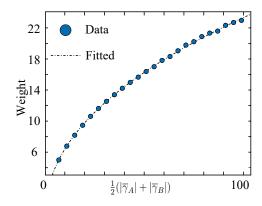
FIG. S6. Numerically computed maximum weight of the gauge stabilizers for 1000 randomly generated code pairs (A and B) with respect to the average weight of $\bar{\gamma}_A$ and $\bar{\gamma}_B$. The dotted-dashed line is power-law fit of the data.

stabilizers, such that the product of these gauge stabilizers incorporates all ancilla Majorana operators, since the product of all measurement stabilizers contains $\overline{\gamma}_A\overline{\gamma}_B$ and all ancilla Majorana operators. The product of all ancilla modes corresponds to a big cycle containing all ancilla vertices, which can always be decomposed into a subset of the independent cycles found above. As a result, the product of all measurement stabilizers and this subset of gauge stabilizers lead to $\mathrm{i}\overline{\gamma}_A\overline{\gamma}_B$. The measurement result of this operator is determined by the product of the measurement results of all measurement stabilizers and this subset of gauge stabilizers, In fact, only measurement results of measurement stabilizers are necessary since they are random due to their anticommutation with some original stabilizers in blocks A and B, while the gauge stabilizer results are always $+1$, since they belong to the stabilizer group of the original code.

In Fig. S5(d), we show an example depicting the results of independent cycle search for performing fermionic lattice surgery between $\mathrm{PG}(2,8)$ and a $d = 5$ fermionic color code. In addition, we describe the fermionic lattice surgery procedure of method 2 in Algorithm 2.

Algorithm 2: Fermionic Lattice Surgery

Input: Codes A, B with stabilizers $\mathcal{S}_A$, $\mathcal{S}_B$, Logical fermionic site supports F_A, F_B
Output: Merged code $\mathcal{C}_{\mathrm{merged}}$

1 $\mathcal{A} \leftarrow \emptyset, \mathcal{S}_{\mathrm{fix}} \leftarrow \{S \in \mathcal{S}_A \cup \mathcal{S}_B | \mathrm{supp}(S) \cap (F_A \cup F_B) \neq \emptyset\}, \mathcal{S}_{\mathrm{free}} \leftarrow (\mathcal{S}_A \cup \mathcal{S}_B) \setminus \mathcal{S}_{\mathrm{fix}}$
2 for $\hat{S} \in \mathcal{S}_{\mathit{fix}}$ **do**
3   **for** $i = 1$ **to** $\frac{1}{2}|supp(S) \cap (F_A \cup F_B)|\rfloor$ **do**
4 $a \leftarrow$ new_ancilla()$, \mathcal{A} \leftarrow \mathcal{A} \cup \{a\}, \hat{S} \leftarrow \hat{S}\hat{\gamma}_a^x\hat{\gamma}_a^z$
5   **end**
6 end
7 $\mathcal{M} \leftarrow \emptyset, p_{\mathrm{odd}} \leftarrow$ null
8 foreach $i \in 1 : \min(|F_A|, |F_B|)$ **do**
9   flag $\leftarrow$ False$, \hat{M}_i \leftarrow \hat{\gamma}_{F_A[i]}\hat{\gamma}_{F_B[i]}$, Attach relevant ancillas to M_i without reuse
10   **if** $wt(\hat{M}_i)$ *is odd* **then**
11   **if** *flag is False* **then**
12 $a \leftarrow$ new_ancilla()$, \mathcal{A} \leftarrow \mathcal{A} \cup \{a\}, \hat{M}_i \leftarrow \hat{M}_i\hat{\gamma}_a,$ flag $\leftarrow$ True
13   **else**
14 $M_i \leftarrow \hat{M}_i\hat{\gamma}_a',$ flag $\leftarrow$ False
15   **end**
16   **end**
17 $\mathcal{M} \leftarrow \mathcal{M} \cup \{\hat{M}_i\}$
18 end
19 Add pair operators for unpaired fermions
20 $G \leftarrow \mathrm{graph}(\mathcal{A})$, add edges for co-occurring ancillas in $\mathcal{S}_{\mathrm{fix}} \cup \mathcal{M}$
21 $\mathcal{G} \leftarrow \{\prod_{v \in C} \hat{\gamma}_v | C \in \text{Independent cycles}(G)\}$
22 return $\mathcal{C}(\mathcal{S}_{\mathrm{free}} \cup \mathcal{S}_{\mathrm{fix}} \cup \mathcal{M} \cup \mathcal{G})$

Fault-tolerance of fermionic lattice surgery

In this subsection, we will analyze the fault-tolerance of fermionic lattice surgery using the subsystem code formalism [63, 64]. The procedure's fault-tolerance is determined by the distance of the dressed logical operators, which will be defined shortly. For method 2 with a processor being a color code, we show that the distance of the dressed logical operators is at least the minimum weight of the original logical operators. For method 1, we provide numerical evidence demonstrating that the dressed logical distance is no smaller than the original logical distance.

A subsystem code is defined on $2n$ Majorana fermions by a gauge group $\mathcal{G}$, which is generated by a subset of Majorana string operators $\mathcal{G} \subset \text{Maj}(2n)$. The stabilizer group of $\mathcal{G}$ is given by $\mathcal{Z}(\mathcal{G}) = \mathcal{C}(\mathcal{G}) \cap \mathcal{G}$, where $\mathcal{C}(\mathcal{G})$ is the centralizer group of $\mathcal{G}$. We define the bare logical operators as $\mathcal{L}_{\text{bare}} = \mathcal{C}(\mathcal{G}) \setminus \mathcal{G}$, the gauge operators as $\mathcal{L}_g = \mathcal{G} \setminus \mathcal{Z}(\mathcal{G})$, and the dressed logical operators as $\mathcal{L}_{\text{dressed}} = \{L | L = gl, \ g \in \mathcal{L}_g, l \in \mathcal{L}_{\text{bare}}\}$. The distance of the dressed logicals is defined as the minimum weight of all dressed logicals [63].

In our lattice surgery construction for method 2, the gauge group is defined as $\mathcal{G} = \langle \mathcal{S}_{\text{old}} \cup \mathcal{S}_{\text{merged}} \cup (\prod_j i \overline{\gamma}_j \overline{\gamma}'_j) \overline{\gamma}_B \rangle$, where $\mathcal{S}_{\text{old}}$ denotes the union of stabilizer groups of codes A and B before merging, $\mathcal{S}_{\text{merged}}$ is the stabilizer group of the merged code, and $\prod_j i \overline{\gamma}_j \overline{\gamma}'_j$ is the product of all logical Majorana operators in codes A and B before merging. When $D_A = D_B$ ($D_A \equiv |\overline{\gamma}_A|$ and $D_B \equiv |\overline{\gamma}_B|$), the stabilizer group of the subsystem code is $\langle \mathcal{S}_{\text{old}} \cap \mathcal{S}_{\text{merged}} \rangle$, the gauge operators are $\langle (\prod_j i \overline{\gamma}_j \overline{\gamma}'_j) \overline{\gamma}_B, c_{B,1}, \ldots, c_{B,D_B-1}, M_1, \ldots, M_{D_B} \rangle$, and the bare logicals consist of all logical operators in codes A and B except $\overline{\gamma}_A$ and $\overline{\gamma}_B$. We note that this construction of gauge group does not apply to the fermionic lattice surgery using method 1, as more operators need to be included in the gauge group [63].

We now prove that the minimum weight of the dressed logicals is at least $\min\{d_A, d_B\}$, where d_A and d_B are the code distances of a fermionic LDPC code A and color code B, repectively. Consider an arbitrary dressed logical $L = gl$ with $g \in \mathcal{L}_g$ and $l \in \mathcal{L}_{\text{bare}}$, we analyze its weight $|L|$ according to the support of l.

If $\text{Supp}(l) \subset B$, where $\text{Supp}(\ldots)$ denotes the set of physical fermionic sites supported by the operator, then $l \sim \overline{\gamma}'_B$. Since $(g)_B$ act only on γ-type Majorana modes, where $(\ldots)_B$ denotes the operator supported on the Majorana modes of B, the weight $|l|$ cannot be reduced below D_B,

For the case with $\text{Supp}(l) \subset A$, we decompose g as $g = l' m c_B a$, where $m \in \langle M_1, \ldots, M_{D_B} \rangle$, $c_B \in \langle c_{B,1}, \ldots, c_{B,D_B-1} \rangle$, $l' \in \{\overline{\gamma}_A, I\}$, and $a \in \{(\prod_j i \overline{\gamma}_j \overline{\gamma}'_j) i \overline{\gamma}_B \overline{\gamma}_A, I\}$ (here if $l' = \overline{\gamma}_A$, then $a = (\prod_j i \overline{\gamma}_j \overline{\gamma}'_j) i \overline{\gamma}_B \overline{\gamma}_A$; otherwise, both are equal to the identity operator). Then,

$$
\begin{aligned}
|L| &= |ll' m c_B a| \\
&\geq |(ll' m c_B a)_A| + |(ll' m c_B a)_B| \\
&= |(ll' m a)_A| + |(m c_B)_B| + |(a)_B| \\
&\geq |(ll' m a)_A| + |(m)_B| + |(a)_B| \\
&= |(ll' m a)_{A \cup B}| \\
&\geq |ll' a| \\
&\geq \min\{d_A, d_B\}.
\end{aligned}
$$

The third line holds because ll' is suppoted on A, $(m c_B)_B$ is supported on γ-type modes, and $(a)_B$ is supported on γ'-type modes. For the fourth line, if $(m)_B \sim \overline{\gamma}_B$, $|(m)_B c_B| \geq |(m)_B|$ since $|(m)_B|$ is the distance of the color code. Otherwise, since $\text{Supp}((m)_B) \subseteq \text{Supp}(\overline{\gamma}_B)$, we write $(m)_B = \overline{\gamma}_B o_B$ where $\text{Supp}(o_B) \subseteq \text{Supp}(\overline{\gamma}_B)$ and $\text{Supp}(o_B) \cap \text{Supp}((m)_B) = \emptyset$. Thus, $c_B(m)_B = c_B \overline{\gamma}_B o_B = \overline{\gamma}_{B,2} o_B$, where $\overline{\gamma}_{B,2} = c_B \overline{\gamma}_B$ is still a logical operator of the color code, whose weight cannot be smaller than $|\overline{\gamma}_B|$. Let us further write $o_B \sim \gamma_{B,i_1} \ldots \gamma_{B,i_r} \gamma_{B,i_{r+1}} \ldots \gamma_{B,i_j}$ where $0 \leq r \leq j \leq D_B$, $i_1, \ldots, i_r \in \text{Supp}(\overline{\gamma}_{B,2})$, and $i_{r+1}, \ldots, i_j \notin \text{Supp}(\overline{\gamma}_{B,2})$. Thus, $|\overline{\gamma}_{B,2} o_B| = |\overline{\gamma}_{B,2}| + j - 2r \geq |\overline{\gamma}_B| + j - 2r = |(m)_B| + 2(j - r) \geq |(m)_B|$. For the sixth line, if $al' \sim I$, then $|(ll' m a)_{A \cup B}| = |(lm)_{A \cup B}|$. We decompose l as $l = l_0 \gamma_{A,j_1} \ldots \gamma_{A,j_s}$ where l_0 does not contain any operator in $\{\gamma_{A,1}, \ldots, \gamma_{A,D_A}\}$. We also have $(m)_{A \cup B} \sim \gamma_{A,i_1} \gamma_{B,i_1} \ldots \gamma_{A,i_p} \gamma_{B,i_p}$, where $0 \leq p \leq D_B$ and $1 \leq i_1, \ldots, i_p \leq D_B$. Thus, multiplying m to l only changes $\gamma_{A,i}$ in l to $\gamma_{B,i}$ if it is present in m. Thus, $|(lm)_{A \cup B}| \geq |l| \geq \min\{d_A, d_B\}$. Similarly, when $l'a \sim (\prod_j \overline{\gamma}_j \overline{\gamma}'_j) \overline{\gamma}_B$, we have $|(ll' m a)_{A \cup B}| \geq |ll'a|$. It follows that $|ll'a| \geq |(a)_B| \geq \min\{d_A, d_B\}$.

If l contains operators in both codes A and B, then $l \sim \overline{\gamma}'_B l_A$ where $\text{Supp}(l_A) \subset A$. Then, $|L| = |gl| = |\overline{\gamma}'_B| + |gl_A| \geq |\overline{\gamma}'_B| + \min\{d_A, d_B\}$.

For method 1 for fermionic lattice surgery, we numerically calculate the distance of the dressed logical operators, and the results are summarized in the table below. We find that the dressed logical distance is consistently no smaller than the original logical distance.

Index	1	2	3	4
Code A	Steane code	$d=5$-Color code	PG$(2,4)$	PG$(2,8)$
Code B	$d=5$-Color code	$d=5$-Color code	$d=5$-Color code	$d=5$-Color code
Distance	3	5	3	5

TABLE II. Distances of subsystem codes during fermionic lattice surgery using method 1.

S-4. DETAILS OF NUMERICAL SIMULATIONS

In this section, we will provide more details regarding calculations of the logical failure rate of the fermionic LDPC memory and simulations of fermionic circuits discussed in the main text.

Logical failure rate

Analogous to the Clifford gate set for qubits, we focus on the Majorana gate set $\{\gamma, \gamma', Z, S, \text{fSWAP}, \text{Braid}\}$, such that applying any gate from this set to a Majorana operator string produces another Majorana operator string [15]. This property allows us to efficiently simulate circuits composed of these gates and even-weight measurement operators on a classical computer, similar to the Gottesman-Knill theorem for qubits [69].

For the simulations shown in Fig. 2(a) in the main text, we use three different fermionic LDPC codes as memory: PG$(2,8)$, EG$(2,4)$, and $[[100,20,7]]_f$ bicycle codes. The memory is randomly initialized in either the logical vacuum state $\overline{|0\ldots0\rangle}$ or the logical product state $\overline{|+\ldots+\rangle}$, where $\overline{\gamma}\overline{|+\rangle} = \overline{|+\rangle}$, both of which can be prepared using a series of braiding operations [56, 57]. We assume no errors occur during this initialization. We then perform N_c rounds of error correction cycles on the memory (see Fig. 2(a) in the main text), and compare the final state with the initial state to determine whether an error has occurred.

At the beginning of each error-correction cycle, a noise channel is applied to each site with a probability p, which we define as the physical error rate. The noise channel applies a γ, γ', or Z gate with equal probability of $\frac{1}{3}$ each. Next, we execute a single round of perfect stabilizer measurement. We then use the Belief Propagation and Ordered Statistics Decoding (BP+OSD) algorithm to identify the most likely error and apply the corresponding gate operations for correction [59]. While BP+OSD is designed for qubit stabilizer codes, it can be directly applied to our fermionic LDPC codes, since our fermionic stabilizer codes share the same parity-check matrix representation as qubit stabilizer codes. After each error correcting gate operation, the same noise channel with probability p is applied to all sites involved in that gate.

As depicted in Fig. 2(a) in the main text, we perform $N_c = 10$ rounds of error correction cycles. We run $N_{\text{sample}} = 10000$ trials for $p < 0.01$, and $N_{\text{sample}} = 2000$ for $p > 0.01$. For each physical error rate, we repeat the above process for $N_{\text{group}} = 20$ independent simulation groups. Within each group, the logical error rate is then defined as $P_L = \frac{N_{\text{error}}}{N_{\text{sample}}}$, where N_{error} is the number of trials with a logical error. The average logical error rate and its standard deviation were calculated from the results of these groups. The logical failure rate is defined as $p_{\text{L}}(p) = 1 - (1 - P_{\text{L}})^{1/N_C}$ [37], where P_L depends on the physical error rate p. The pseudo-threshold of this code is the physical error rate p_{th}, such that $p_{\text{L}}(p_{\text{th}}) = P(p_{\text{th}}, k)$ [38], where $P(p, k)$ represents the probability that at least one of k physical fermion sites experiences an error.

Fermionic circuit simulation

We now provide the details of Fig. 4 in the main text. There, we choose the PG$(2,8)$ code as the fermionic memory, select 4 logical modes from it, and use two fermionic Steane codes as the processors. The memory is initialized in the $\overline{|0101\rangle}$ state. We then perform multiple rounds of the circuit shown in Fig. 4(a), where each round represents a time step. To implement the Braid gate or fSWAP gate between logical modes, we transfer the corresponding logical modes from the memory to the processors via fermionic lattice surgery, perform the gate on the processors, and then transfer the logical modes back to the memory. The processors are then reset to $\overline{|00\rangle}$ state, preparing them for the next Braid or fSWAP gate. We apply the same error model as described previously, with a physical error rate of $p = 0.005$. We execute the circuit with and without error correction. Error correction is performed during each Braid or fSWAP gate. We track the expectation value of logical particle numbers at site 1 and 4 at each time step to compare the simulation results of both scenarios. In the main text, we employ the first method to perform fermionic lattice surgery. Here, in

Fig. S7, we provide the simulation results achieved using the second method for lattice surgery.
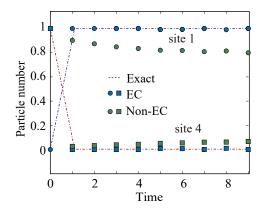

FIG. S7. The time evolution of logical particle numbers at site 1 and 4, which are the same as in Fig. 4(b) except that the lattice surgery is performed using the second method. The error bars for blue circles and green squares are hidden behind the symbols.