Error-correcting codes for fermionic quantum simulation

Yu-An Chen[1,2], Alexey V. Gorshkov[1], and Yijia Xu[1,3]

[1]Joint Quantum Institute and Joint Center for Quantum Information and Computer Science, NIST/University of Maryland, College Park, Maryland 20742, USA
[2]International Center for Quantum Materials, School of Physics, Peking University, Beijing 100871, China
[3]Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20742, USA

August 16, 2023

Abstract

Utilizing the framework of $\mathbb{Z}_2$ lattice gauge theories in the context of Pauli stabilizer codes, we present methodologies for simulating fermions via qubit systems on a two-dimensional square lattice. We investigate the symplectic automorphisms of the Pauli module over the Laurent polynomial ring. This enables us to systematically increase the code distances of stabilizer codes while fixing the rate between encoded logical fermions and physical qubits. We identify a family of stabilizer codes suitable for fermion simulation, achieving code distances of $d = 2, 3, 4, 5, 6, 7$, allowing correction of any $\lfloor \frac{d-1}{2} \rfloor$-qubit error. In contrast to the traditional code concatenation approach, our method can increase the code distances without decreasing the (fermionic) code rate. In particular, we explicitly show all stabilizers and logical operators for codes with code distances of $d = 3, 4, 5$. We provide syndromes for all Pauli errors and invent a syndrome-matching algorithm to compute code distances.

Contents

1 Introduction

Error-correcting codes were initially developed to correct quantum errors on noisy quantum devices and have found further applications in condensed matter physics and high-energy physics. The cornerstone of quantum error correction is the stabilizer formalism [1], which defines the codewords in the common eigenspace of elements in an Abelian group, referred to as the stabilizer group. A stabilizer code is labeled $[[n, k, d]]$ when it uses n physical qubits to encode k logical qubits with code distance d. The code distance is the minimum weight of an operator that commutes with all elements of $\mathcal{G}$ but is not in $\mathcal{G}$ itself. The ratios $\frac{k}{n}$ and $\frac{d}{n}$ determine the quality of codes. Recent developments show the existence of "good" quantum low-density parity-check (LDPC) codes, i.e., with the number of logical qubits k and the code distance d both scaling linearly with the number of physical qubits n [2, 3, 4, 5]. In this paper, our focus is on a different objective. Instead of encoding logical qubits, we aim to encode logical fermions using physical qubits. This motivation comes from the need to simulate fermions on quantum computers since most models of matter involve electrons, which are, in fact, fermions [6, 7, 8, 9, 10, 11, 12, 13, 14]. While fault-tolerant quantum computation [15, 16] is the ultimate goal, current devices still have limited resources and suffer from noise, so error-mitigation schemes are crucial. Therefore, we seek an effective design such that when we implement fermions with qubits on a quantum computer, certain qubit errors can be corrected directly in this protocol without having to encode the underlying qubits further. Thus, we want to systematically increase the code distance d in a fermion-to-qubit mapping with a fixed code rate (between logical fermions and physical qubits).[1]

    When a fermionic Hamiltonian consists of geometrically local terms, they can be mapped to local qubit operators by the Bravyi-Kitaev superfast encoding and its variants [8, 17], by the auxiliary methods [18, 19, 20, 21, 22, 23], or by exact bosonization [24, 25, 26]. The mappings between fermionic Hamiltonians and higher-spin Hamiltonians are also studied in Refs. [27, 28, 29, 30]. There are also proposals that utilize defects of surface codes for fermionic quantum simulation [31, 32, 33] and those defects are recently implemented by Google Quantum AI [34]. Variants of these mappings have been studied to optimize different costs [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48]. In the context of quantum many-body physics, these

---

[1]The code rate here is defined as the ratio between the number of logical fermionic modes k and the number of physical qubits n, in the $n \to \infty$ limit. Each code will be demonstrated in an infinite plane, but they can be defined on a torus or open disk (up to some boundary modifications) with linear size L, such that both n and k scale with L^2. If L is sufficiently large, the boundary effects are negligible, and the ratio k/n will converge to the code rate.

mappings also reveals the deep connections between fermion and spin systems [49, 50]. Aside from the investigation of constructing new mappings, fermion-to-qubit mappings have been studied in the context of variational quantum circuits [51, 52]. In all the above-mentioned methods, extra qubits are required, e.g., the number of qubits is twice the number of fermions on a 2d square lattice. This is the price for the locality-preserving property.[2] These methods can be thought of as stabilizer codes. Given N fermions with the Hilbert space dimension 2^N, they are mapped to $2N$ qubits with space dimension 2^{2N}, which is an enlarged space. After N gauge constraints (stabilizer conditions) are imposed, the gauge-invariant subspace (code space) has dimension 2^N, which matches the dimension of the logical fermions. It has been shown that gauge constraints can be utilized for error correction [53], and code distances can be studied for these stabilizer codes. Ref. [17] demonstrates that an improved Bravyi-Kitaev superfast encoding can correct any single-qubit error in a graph where each vertex has degree $d \geq 6$. In Ref. [54], another version of the Bravyi-Kitaev superfast encoding is proposed, called the "Majorana loop stabilizer code," which is designed to have code distance $d = 3$ such that any single-qubit error can be corrected. However, it is not known how to generalize the Bravyi-Kitaev superfast encoding to produce codes with higher and higher code distances. An alternative approach is code concatenation, where logical qubits in error-correcting codes replace physical qubits in the fermion-to-qubit mappings. However, code concatenation will decrease the code rate between logical fermions and physical qubits, increasing the overhead of fermionic simulation. In this work, we present a method that increases the code distances of fermion-to-qubit mappings while preserving the code rate.

In this paper, we conjugate an existing stabilizer code with a Clifford circuit.[3] This produces a new stabilizer code. Since the new code is obtained via conjugation by a unitary operator, the algebra of the logical operators is preserved. If we choose the circuit wisely, the new stabilizer code will have a larger code distance ($d \geq 3$) where the code distance of the original exact bosonization is $d = 2$. To study Clifford circuits systematically, we utilize the Laurent polynomial method introduced in Refs. [55, 56] and further extended in Ref. [57], which shows that any Pauli operator can be written as a vector in a symplectic space. For a system with translational symmetry, e.g., the 2d square lattice, the space of Pauli operators becomes a module over a polynomial ring. Furthermore, this polynomial method can be used to formulate the 2d bosonization concisely [58]. The commutation relations of Pauli operators are determined by the symplectic form. Ref. [59] shows that there is a one-to-one correspondence (up to a translation operator on the lattice):

$$\text{Automorphism of the symplectic form} \Longleftrightarrow \text{Clifford circuit on a 2d square lattice.}$$

Therefore, the problem of finding new codes turns into a problem of searching for "good"[4] automorphisms of the Pauli module with the symplectic form, which can be achieved efficiently by exhaustive numerical search.

In this work, we use the Laurent polynomial method to construct bosonizations on a 2d square lattice. Table 1 is the summary of our results. In Section 2, we review the original 2d bosonization method [24] in Section 2.1 and then pictorially construct 2d bosonizations

---

[2]We can apply the Jordan-Wigner transformation on the 2d lattice by choosing a path including all vertices. However, some local fermionic terms will be mapped to long string operators that are highly nonlocal.

[3]A circuit U is Clifford if and only if $UPU^\dagger$ is a product of Pauli matrices for any given Pauli matrix P.

[4]Here, "good" refers to symplectic automorphisms that generate bosonizations with higher code distances while preserving locality and code rates.

	distance	occupation	hopping	interaction	stabilizer
Bravyi-Kitaev superfast encoding	2	4	6	6	6
Majorana loop stabilizer code	3	3	3-4	4-6	4-10
Exact bosonization ($d = 3$)	3	4	3-5	6	8
Exact bosonization ($d = 4$)	4	6	5-6	10	10
Exact bosonization ($d = 5$)	5	8	5-9	12-14	12
Exact bosonization ($d = 6$)	6	12	6-13	16-20	18
Exact bosonization ($d = 7$)	7	12	7-23	16-18	26

Table 1:   A comparison of codes based on modified exact bosonization to the Bravyi-Kitaev superfast encoding [8] and to the Majorana loop stabilizer code [54] on a 2d square lattice. The $d = 2$ exact bosonization is equivalent to the Bravyi-Kitaev superfast encoding with a specific choice of the ordering of edges [48]. We list the code distance, as well as the weights (after mapping to qubits) of a fermion occupation term (local fermion parity term), of a hopping term, and of a density-density interaction between nearest neighbors. The weights of the stabilizers are also shown.

with distances of $3, 4$, and 5 in Section 2.2, 2.3, and 2.4, respectively. We review the Laurent polynomial method in Section. 3.1. In Section. 3.2, we describe all these bosonizations within the framework of the Laurent polynomial method. In addition, in Section. 3.3, we describe a computerized method to search forbosonizations. In Appendix A, we discuss the "syndrome matching" method used to compute the code distance of a given bosonization. In Appendix B, we describe 16 elementary automorphisms. In Appendix C, we show the polynomial representations of an automorphism with a distance of 6 and another with a distance of 7.

2   Results

In Section 2.1, we begin by reviewing the original 2d bosonization on a square lattice from Ref. [24]. Then we demonstrate a new way to perform bosonization with code distances of $d = 3$, $d = 4$, and $d = 5$ in Section 2.2, Section 2.3, and Section 2.4, respectively.

2.1   Review of the original bosonization

We first describe the Hilbert space in Fig. 1. The elements associated with vertices, edges, and faces will be denoted by v, e, and f, respectively. On each face f of the lattice, we place a single pair of fermionic creation-annihilation operators $c_f, c_f^\dagger$, or equivalently a pair of Majorana fermions γ_f, γ_f'. The even fermionic algebra consists of local observables with trivial fermionic parity, i.e., local observables that commute with the total fermion parity $(-1)^F \equiv \prod_f (-1)^{c_f^\dagger c_f}$ where $F = \sum_f c_f^\dagger c_f$ is the total fermion number.[5] The even algebra is generated by [24]:

---

[5]The even fermionic algebra can also be considered as the algebra of local observables containing an even number of Majorana operators.
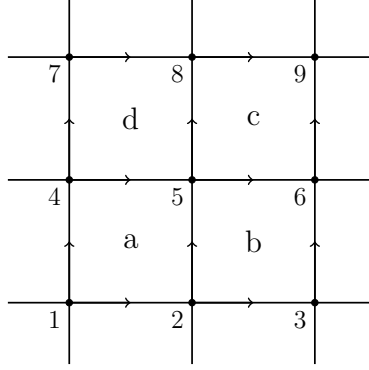
Figure 1: Bosonization on a square lattice [24]. We put Pauli matrices X_e, Y_e, and Z_e on each edge and one complex fermion $c_f, c_f^\dagger$ on each face. We will work in the Majorana basis $\gamma_f = c_f + c_f^\dagger$ and $\gamma_f' = -i(c_f - c_f^\dagger)$ for convenience.

1. On-site fermion parity:

$$P_f \equiv -i\gamma_f\gamma_f'. ^6 \tag{1}$$

2. Fermionic hopping term:

$$S_e \equiv i\gamma_{L(e)}\gamma_{R(e)}', \tag{2}$$

   where $L(e)$ and $R(e)$ are faces to the left and right of e, with respect to the orientation of e in Fig. 1.

The bosonic dual of this system involves $\mathbb{Z}_2$-valued spins on the edges of the square lattice. For every edge e, we define a unitary operator U_e that squares to 1. Labeling the faces and vertices as in Fig. 1, we define:

$$\begin{aligned} U_{56} &= X_{56}Z_{25}, \\ U_{58} &= X_{58}Z_{45}, \end{aligned} \tag{3}$$

where X_e, Z_e are Pauli matrices acting on a spin at edge e:

$$X_e = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \ Z_e = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{4}$$

Operators U_e for the other edges are defined by using translation symmetry. Pictorially, operator U_e is depicted as

$$U_e = \quad \underset{-Z-}{\overset{|}{X_e}} \quad \text{or} \quad \overset{-X_e-}{\underset{|}{Z}} \quad , \tag{5}$$

corresponding to the vertical or horizontal edge e.

---

[6] $P_f = -i\gamma_f\gamma_f' = (-1)^{c_f^\dagger c_f}$ measures the occupancy of the fermionic mode at face f.

In Ref. [24], U_e and S_e are shown to satisfy the same commutation relations. We also map the fermion parity P_f at each face f to the "flux operator" $W_f \equiv \prod_{e \subset f} Z_e$, the product of Z_e around a face f:

$$W_f = \begin{array}{c} -Z- \\ Z \quad f \quad Z \\ -Z- \end{array}. \tag{6}$$

The bosonization map is

$$\begin{aligned} S_e &\longleftrightarrow U_e, \\ P_f &\longleftrightarrow W_f, \end{aligned} \tag{7}$$

or pictorially

$$
\begin{array}{c}
\gamma_{L(e)} \\
i \times \quad \dfrac{\quad e \quad}{\quad} \quad \longleftrightarrow \quad Z \begin{array}{c} -X_e- \\ \; \end{array} \quad , \\
\gamma'_{R(e)}
\end{array}
\tag{8}
$$

$$
i \times \quad \gamma_{L(e)} \; \left| e \; \gamma'_{R(e)} \right. \quad \longleftrightarrow \quad \begin{array}{c} X_e \\ -Z- \end{array},
$$

$$
-i\gamma_f \gamma'_f \quad \longleftrightarrow \quad \begin{array}{c} -Z- \\ Z \quad f \quad Z \\ -Z- \end{array} \quad .
$$

The condition $P_a P_c S_{58} S_{56} S_{25} S_{45} = 1$ on fermionic operators gives a gauge (stabilizer) constraint $G_v = W_{f_c} \prod_{e \supset v_5} X_e = 1$ for bosonic operators, or generally

$$
G_v = \quad \begin{array}{c} -Z- \\ X Z \qquad Z \\ -X- v -XZ- \\ X \end{array} = 1.
\tag{9}
$$

The gauge constraint Eq. (9) can be considered as the stabilizer ($G_v |\Psi\rangle = |\Psi\rangle$ for $|\Psi\rangle$ in the code space), which forms the stabilizer group $\mathcal{G}$. The operators U_e and W_f generate all logical operators.[7] The weight of a Pauli string operator O is the number of Pauli matrices in O, denoted as $\mathrm{wt}(O)$. For example, we have $\mathrm{wt}(U_{56}) = \mathrm{wt}(U_{58}) = 2$, $\mathrm{wt}(W_f) = 4$, and $\mathrm{wt}(G_v) = 6$. The code distance d is defined as the minimum weight of a logical operator excluding stabilizers:

$$d = \min\{\mathrm{wt}(O) \mid [O, \mathcal{G}] = 0, \ O \notin \mathcal{G}\}. \tag{10}$$

---

[7]The logical operators consist of all operators that commute with $\mathcal{G}$. Elements of $\mathcal{G}$ are trivial logical operators as stabilizers have no effect on the code space. U_e and W_f together generate all the other logical operators.

The code distance of this original bosonization is $d = 2$ as U_e has weight 2 and any single Pauli matrix violates at least one G_v, which implies that there is no logical operator with weight 1.

There are 4 types of nearest-neighbor hopping terms ($\gamma_L \gamma_R'$, $\gamma_L \gamma_R$, $\gamma_L' \gamma_R'$, and $\gamma_L' \gamma_R$) and one type of fermion occupation term ($-i\gamma_f \gamma_f'$). When mapped to Pauli matrices, their weights wt_i are in the range $2 \leq \mathrm{wt}_i \leq 6$. The maximum weight corresponds to the worst case to simulate the fermion hopping term or the fermion occupation term. A good stabilizer code requires a balance between the minimum weight and the maximum weight. A high minimum weight guarantees the error-correcting property, while a low maximum weight implies that the cost of simulation is low. We label the minimum and maximum weights of the hopping terms as $\mathrm{wt}_{\min}$ and $\mathrm{wt}_{\max}$. In this example, $(\mathrm{wt}_{\min}, \mathrm{wt}_{\max}) = (2, 6)$.

2.2 Bosonization with code distance $d = 3$

We now introduce a new way to map the fermionic operators S_e and P_f to Pauli matrices. For simplicity, we present the mapping in a pictorial way:

$$(11)$$

The stabilizer on the bosonic side is

$$G_v^{d=3} = (-1) \times \quad = 1. \qquad (12)$$

Notice that there is a minus sign coming from $ZXZ = -X$.[8] We can manually check that the logical operators defined in Eq. (11) do commute with the stabilizer in Eq. (12). We will prove that this mapping preserves the fermionic algebra in Section. 3..

---

[8]The stabilizer is derived from the identity $P_a P_c S_{58} S_{56} S_{25} S_{45} = 1$. After we map P_f and S_e to Pauli matrices using Eq. (11), it becomes $G_v^{d=3}$.
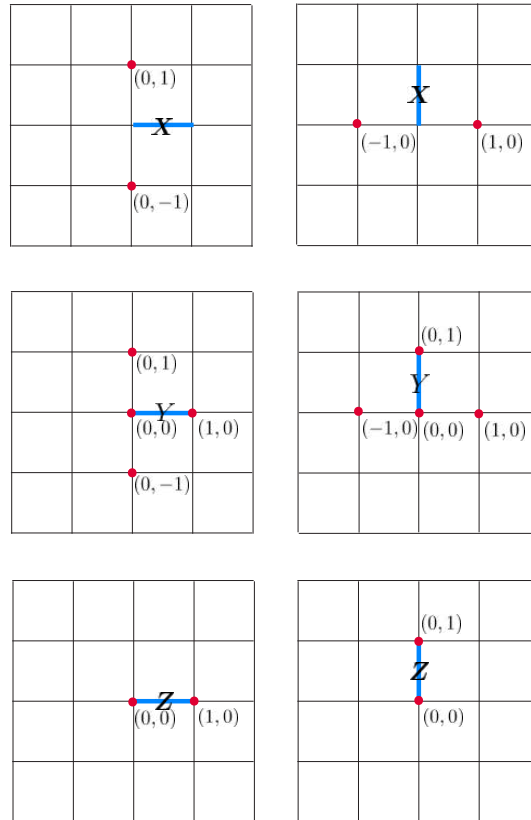
Figure 2: Syndromes of single-qubit errors for the bosonization with code distance $d = 3$. The red vertices v represent locations where the single-qubit error does not commute with the stabilizer G_v.

Given the stabilizer, we can provide the syndromes for all the single-qubit Pauli errors, as shown in Fig. 2. We see that all single-qubit Pauli matrices have different syndromes, which means that we do not have any logical operators with weight 2. This implies a code distance of $d \geq 3$. Eq. (11) shows logical operators with weight 3, so we conclude that the code distance is $d = 3$. Based on the syndrome measurements, we can always correct any single-qubit error according to Fig. 2.

The 4 types of nearest-neighbor hopping terms, $\gamma_L \gamma'_R$, $\gamma_L \gamma_R$, $\gamma'_L \gamma'_R$, and $\gamma'_L \gamma_R$ have weights wt_i in the range $3 \leq \mathrm{wt}_i \leq 5$. Therefore, the modified bosonization has $(\mathrm{wt}_{\min}, \mathrm{wt}_{\max}) = (3, 5)$ and a fermionic occupation term of weight 4. Compared to the original bosonization with $(\mathrm{wt}_{\min}, \mathrm{wt}_{\max}) = (2, 6)$, the minimum weight is increased such that error correction can be performed, while the maximum weight of the hopping terms is decreased implying a reduction in the simulation cost.

2.3   Bosonization with code distance $d = 4$

In this subsection, we provide a construction of an exact bosonization with code distance $d = 4$ as an intermediate step toward $d = 5$. Since its code distance is $d = 4$, which is even, this code (like the $d = 3$ code) can only correct the $\lfloor \frac{d-1}{2} \rfloor = 1$ Pauli error. However, for error defection, Pauli errors up to weight 3 can be observed from the stabilizer syndrome measurements.

The mapping can be described as

$$
i \times \quad
\begin{array}{c}
\gamma_{L(e)} \\
\underline{\quad e \quad} \\
\gamma'_{R(e)}
\end{array}
\quad \longleftrightarrow \quad
\begin{array}{c}
-X_e- \\
Z \quad\quad Z \\
-X- \\
Z
\end{array} ,
$$

$$
i \times \quad \gamma_{L(e)} \; \Big| e \; \gamma'_{R(e)} \quad \longleftrightarrow \quad
\begin{array}{c}
-Z- \quad -Z- \\
X \quad\quad X_e \\
-Z-
\end{array} , \tag{13}
$$

$$
-i \gamma_f \gamma'_f \quad \longleftrightarrow \quad
\begin{array}{c}
-Z- \\
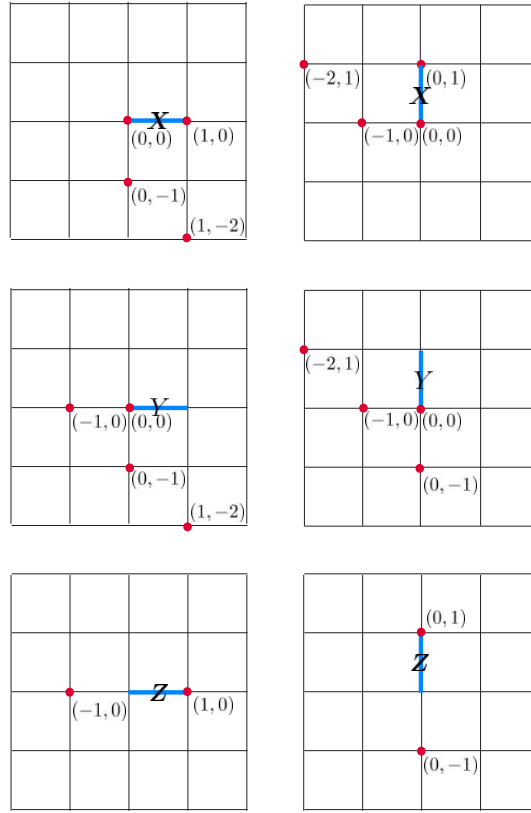X \quad f \\
-XZ- -X- \\
XZ \quad\quad Z
\end{array} .
$$

Figure 3: Syndromes of single-qubit errors for the bosonization with code distance $d = 4$. The red vertices v represent locations where the single-qubit error does not commute with the stabilizer $G_v^{d=4}$.

The stabilizer becomes

$$
G_v^{d=4} = \quad \substack{\displaystyle -Z- \\ \displaystyle X \\ \displaystyle -Z- \\ \displaystyle Z \quad Z \\ \displaystyle -XZ\text{-}v-Z-\ -X- \\ \displaystyle XZ \qquad Z} \quad = 1.
\tag{14}
$$

We can check that the logical operators in Eq. (13) commute with the stabilizer in Eq. (14). The proof of the equivalence between the even fermionic algebra and this stabilizer code will be shown in Section. 3.

The syndromes for all single-qubit Pauli errors are provided in Fig. 3. From the generators of the logical operators in Eq. (13), we may be tempted to conclude that the code distance is $d = 5$ because the minimum weight is 5. However, based on the syndromes in Fig. 3, we find

that the following operator is logical:

$$
\begin{array}{c}
\text{(15)}
\end{array}
$$

Since it does not commute with the terms in Eq. (13), this operator does not belong to the stabilizer group. Therefore, the code distance for this stabilizer code is $d \leq 4$. In Appendix A, we introduce the "syndrome matching" method to provide a lower bound for the code distance for a given stabilizer code. With this method, we check that the stabilizer code defined by Eq. (14) has a code distance of $d = 4$.

The minimum and maximum weights of the nearest-neighbor terms are $(\mathrm{wt}_{\min}, \mathrm{wt}_{\max}) = (5, 6)$ and the fermionic occupation term has weight 6, which means that all the operations are quite well-balanced. This code has an error-correcting property for any single-qubit error.

2.4 Bosonization with code distance $d = 5$

The bosonization map with code distance $d = 5$ is provided in this subsection. The generators of the even fermionic algebra are mapped to Pauli matrices as shown below:

$$
\begin{array}{c}
\text{(16)}
\end{array}
$$

$$
\begin{array}{c}
\text{(17)}
\end{array}
$$

$$
\begin{array}{c}
\text{(18)}
\end{array}
$$

The stabilizer is

$$
G_v =
\begin{array}{c}
\begin{array}{ccc}
X & X\!Z & Z \\
-Z-\!\!-Z-\!\!-XZ-\!\!-Z- \\
Z \\
v \\
\begin{array}{cc} X & X\!Z \end{array} \\
-XZ-\!\!-Z-
\end{array}
\end{array}
= 1. \tag{19}
$$

The minimum and maximum weights of the nearest-neighbor terms are $(\mathrm{wt}_{\min}, \mathrm{wt}_{\max}) = (5, 9)$ and the weight of the fermionic occupation term is 8. We use the "syndrome matching" method in Appendix A to confirm that $d = 5$. This code has an error-correcting property for any two-qubit error.
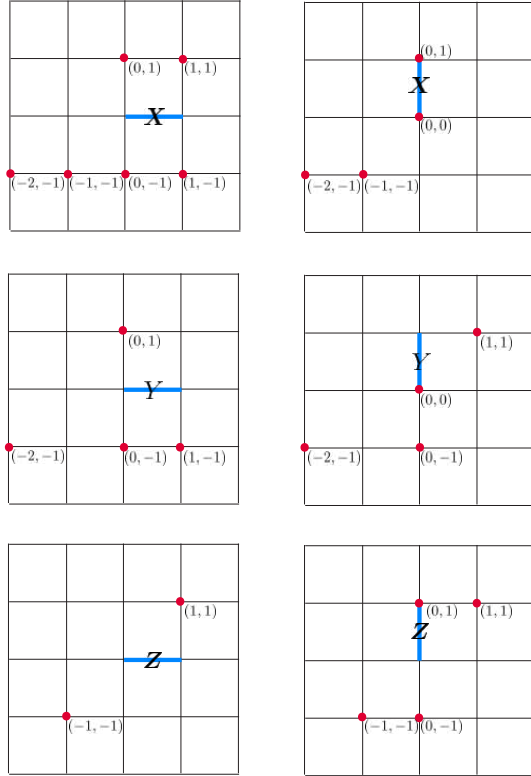

Figure 4: Syndromes of single-qubit errors for the $d = 5$ bosonization.

3 Stabilizer codes and the Pauli module

This section discusses the stabilizer code formalism and the Pauli module representation via Laurent polynomials. The Laurent polynomial method is reviewed in Section. 3.1. Then, in

Section. 3.2, we discuss bosonization with distance $d = 3, 4, 5$ and the corresponding symplectic automorphisms. The searching algorithm for automorphisms is presented in Section. 3.3.

3.1 Review of the Laurent polynomial method for the Pauli algebra

We start by reviewing how any Pauli operator can be expressed as a vector over a polynomial ring $R = \mathbb{F}_2[x, y, x^{-1}, y^{-1}]^9$ as set out in Ref. [55]. First, we define X_{12}, Z_{12}, X_{14}, and Z_{14} in Fig. 1 as column vectors:

$$X_{12} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \; Z_{12} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \; X_{14} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \; Z_{14} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{20}$$

All the other edges can be defined with the help of translation operators as follows. We use polynomials of x and y to represent translation in the x and y directions, respectively. For example,

$$Z_{78} = y^2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ y^2 \\ 0 \end{bmatrix}, \; X_{58} = xy \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ xy \\ 0 \\ 0 \end{bmatrix}. \tag{21}$$

More examples are included in Fig. 5.

Figure 5: Examples of polynomial expressions for Pauli strings. The flux term (i.e. fermionic occupation) on a plaquette and the hopping term on an edge are both shown. The factors such as x^2y^2 and x^2 represent the locations of the operators relative to the origin.

To determine whether two Pauli operators represented by vectors v_1 and v_2 commute or anti-commute, we define the dot product as

$$v_1 \cdot v_2 = \bar{v}_1 \Lambda v_2, \tag{22}$$

---

[9]This is the ring that consists of all polynomials of x, x^{-1}, y, y^{-1} with coefficients in $\mathbb{F}_2$.

where $\overline{\cdots}$ is the transpose operation on a matrix along with $x, y \to x^{-1}, y^{-1}$ and

$$\Lambda = \left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{array}\right]. \tag{23}$$

Notice that -1 is the same as 1 here because we are working over the $\mathbb{Z}_2$ field. The minus sign just reminds us of the symplectic form. The two operators v_1 and v_2 commute if and only if the constant term of $v_1 \cdot v_2$ is zero. For example, we calculate the dot products

$$X_{12} \cdot Z_{12} = 1, \quad X_{58} \cdot Z_{14} = x^{-1}y^{-1}, \tag{24}$$

and, therefore, X_{12} and Z_{12} anti-commute, whereas X_{58} and Z_{14} commute (their dot product only has a non-constant term $x^{-1}y^{-1}$). Furthermore, $X_{58} \cdot Z_{14} = x^{-1}y^{-1}$ means that the shifting of X_{58} in $-x$ and $-y$ directions by 1 step will anti-commute with Z_{14}. This dot product defined by Λ is referred to as the symplectic form. A translationally invariant stabilizer code forms an R-submodule[10] V such that

$$v_1 \cdot v_2 = \bar{v}_1 \Lambda v_2 = 0, \quad \forall v_1, v_2 \in V. \tag{25}$$

We now study the automorphisms A of the symplectic form Λ:

$$(Av_1) \cdot (Av_2) = v_1 \cdot v_2, \quad \forall v_1, v_2 \in V. \tag{26}$$

This is equivalent to $\bar{A}\Lambda A = \Lambda$, which, taking $A = \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right]$, becomes

$$\left[\begin{array}{c|c} \bar{a} & \bar{c} \\ \hline \bar{b} & \bar{d} \end{array}\right] \left[\begin{array}{c|c} 0 & I \\ \hline -I & 0 \end{array}\right] \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right] = \left[\begin{array}{c|c} 0 & I \\ \hline -I & 0 \end{array}\right] \tag{27}$$

$$\Rightarrow \quad \bar{a}d - \bar{c}b = I, \quad \bar{a}c = \bar{c}a, \quad \bar{b}d = \bar{d}b.$$

Examples of the automorphism A are

$$S = \left[\begin{array}{c|c} I & 0 \\ \hline c & I \end{array}\right], \text{where } c \in \text{Mat}_2[R] \text{ and } \bar{c} = c,$$

$$H = \left[\begin{array}{c|c} 0 & I \\ \hline -I & 0 \end{array}\right], \tag{28}$$

$$C = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ r & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & \bar{r} \\ 0 & 0 & 0 & 1 \end{array}\right], \text{where } r \in R.$$

$\text{Mat}_2[R]$ consists of all 2×2 matrices with entries in $R = \mathbb{F}_2[x, y, x^{-1}, y^{-1}]$.

---

[10]The R-submodule is similar to a subspace of a vector space, but the entries of the vector are in the ring $R = \mathbb{F}_2[x, y, x^{-1}, y^{-1}]$. In a ring, the inverse element may not exist. This is the distinction between a module and a vector space.

3.2 New stabilizer codes developed from automorphisms

First, we reformulate the original bosonization introduced in Section 2.1 and incorporate it into the Pauil module. For simplicity, we will write x^{-1} and y^{-1} as $\overline{x}$ and $\overline{y}$, respectively. The original hopping operators U_e in Eq. (5) can be written as

$$
U_1 = \begin{bmatrix} 1 \\ 0 \\ \hline 0 \\ \overline{y} \end{bmatrix}, \quad U_2 = \begin{bmatrix} 0 \\ 1 \\ \hline \overline{x} \\ 0 \end{bmatrix}, \tag{29}
$$

where U_1 represents U_e on the horizontal edge and U_2 represents U_e on the vertical edge. The flux term W_f in (6) is written as

$$
W = \begin{bmatrix} 0 \\ 0 \\ \hline 1+y \\ 1+x \end{bmatrix}. \tag{30}
$$

The stabilizer G_v in Eq. (9) corresponds to the vector

$$
G = \begin{bmatrix} 1+\overline{x} \\ 1+\overline{y} \\ \hline 1+y \\ 1+x \end{bmatrix}. \tag{31}
$$

Now, we will apply automorphisms on these vectors to generate new stabilizer codes.

3.2.1 Automorphism for code distance $d = 3$

We consider the simplest automorphism

$$
A_1 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array} \right], \tag{32}
$$

which modifies the Pauli operator X_e as

$$
A_1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad A_1 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}. \tag{33}
$$

Pictorially, this is equivalent to

$$
X_e = \begin{cases} -X_e- & \xrightarrow{A_1} & \overset{\downarrow}{Z} \\ & & {}^{|}_{|}-X_e- \\ \\ \overset{\downarrow}{\underset{|}{X_e}} & \xrightarrow{A_1} & \overset{\downarrow}{\underset{|}{X_e}} \\ & & {}_{|}-Z- \end{cases}. \tag{34}
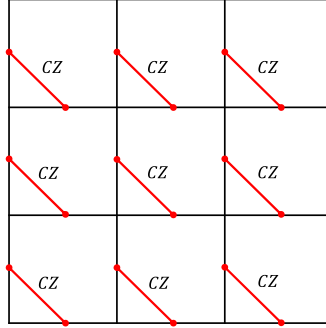$$

Figure 6: Clifford circuit corresponding to automorphism A_1

Notice that Z_e is unchanged under this automorphism. This automorphism corrsponds to the Clifford circuit shown in Fig. 6.

Now we apply A_1 on the logical operators U_1, U_2, and W and the stabilizer G:

$$
A_1 U_1 = \left[\begin{array}{c} 1 \\ 0 \\ \hline 0 \\ 1 + \overline{y} \end{array}\right], \quad A_1 U_2 = \left[\begin{array}{c} 0 \\ 1 \\ \hline 1 + \overline{x} \\ 0 \end{array}\right],
$$

$$
A_1 W = \left[\begin{array}{c} 0 \\ 0 \\ \hline 1 + y \\ 1 + x \end{array}\right], \quad A_1 G = \left[\begin{array}{c} 1 + \overline{x} \\ 1 + \overline{y} \\ \hline y + \overline{y} \\ x + \overline{x} \end{array}\right].
$$

(35)

The automorphism A_1 applied on U_e can be visualized as

$$
U_e = \left\{
\begin{array}{c}
\begin{array}{ccc}
\overset{\downarrow}{X_e} & \xrightarrow{A_1} & \overset{\downarrow}{X_e} \\
-Z-^{|} & & -Z-^{|}-Z- \\
\end{array} \\
\\
\begin{array}{ccc}
-X_e- & & \overset{\downarrow}{Z} \\
\overset{|}{Z} & \xrightarrow{A_1} & \overset{|}{-X_e-} \\
& & \overset{|}{Z} \\
\end{array}
\end{array}
\right. .
$$

(36)

The flux term Eq. (6) is unchanged. The automorphism A_1 applied on the stabilizer G_v is

$$
\begin{array}{ccc}
\begin{array}{c}
-Z- \\
X\!Z \quad Z \\
-X-\,v\,-X\,Z-^{|} \\
X
\end{array}
& \xrightarrow{A_1} &
\begin{array}{c}
-Z- \\
Z \quad X \quad Z \\
-X-\,v\,-X- \\
X \\
-Z-
\end{array}
\end{array} .
$$

(37)

This is the bosonization with code distance $d = 3$ introduced in Section 2.2. Since we applied the automorphism of the Pauli module on the original bosonization, the logical operators satisfy the same algebra. Therefore, we conclude that this new stabilizer code is a valid way to simulate fermions.

3.2.2   Automorphism for code distance $d = 4$

In this section, we consider a slightly more complicated automorphism A' [11]:

$$
A' = \left[
\begin{array}{cc|cc}
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & \overline{x}y & 1 + \overline{x}y & 0 \\
x\overline{y} & 0 & 0 & 1 + x\overline{y}
\end{array}
\right].
\tag{38}
$$

One can easily check that A' indeed satisfies the condition in Eq. (27) for being an automorphism. Applying A' on the logical operators U_1, U_2, and W and the stabilizer G, we get

$$
A'U_1 = \left[
\begin{array}{c}
1 + \overline{y} \\
0 \\
\hline
0 \\
x\overline{y}^2 + x\overline{y} + \overline{y}
\end{array}
\right], \qquad
A'U_2 = \left[
\begin{array}{c}
0 \\
1 + \overline{x} \\
\overline{x} + \overline{x}y + \overline{x}^2 y \\
0
\end{array}
\right],
$$

$$
A'W = \left[
\begin{array}{c}
1 + x \\
1 + y \\
1 + y + \overline{x} + \overline{x}y^2 \\
1 + x + \overline{y} + \overline{y}x^2
\end{array}
\right], \qquad
A'G = \left[
\begin{array}{c}
x + \overline{x} \\
y + \overline{y} \\
1 + y + \overline{x} + \overline{x}y^2 \\
1 + x + \overline{y} + \overline{y}x^2
\end{array}
\right].
\tag{39}
$$

The operators $A'(U_e)$ can be depicted as

$$
U_e = \left\{
\begin{array}{c}
\\
\\
\end{array}
\right.
\tag{40}
$$

---
[11] $A' = A_4 A_7$ in terms of the elementary automorphisms defined in Appendix B.

The stabilizer $A'(G_v)$ is

$$
\xrightarrow{A'} \tag{41}
$$

The logical operator U_e and the stabilizer G_v are mapped on the bosonization with code distance $d = 4$ introduced in Section 2.3.

Finally, the flux term under the automorphism A' becomes

$$
W_f = \xrightarrow{A'} \tag{42}
$$

This term can be simplified by multiplying it by $A'(G_v)$, which does not change the effective logical operation. The resulting flux term is

$$
\tag{43}
$$

which matches Eq. (13) in Section 2.3.

3.2.3 Automorphism for code distance $d = 5$

We introduce another automorphism A'' [12]:

$$
A'' = \left[
\begin{array}{cc|cc}
1 & \overline{x} & x & 1 \\
1 & \overline{x}+1 & 1+x & 1 \\
x & \overline{x}+1 & \overline{x}+1+x+x^2 & 1+x \\
x & 1 & x+x^2 & 1+x
\end{array}
\right]. \tag{44}
$$

---

[12] $A'' = A_9 A_3 A_7 A_{14}$ in terms of the elementary automorphisms defined in Appendix B.

Again it is easy to check that it satisfies the condition in Eq. (27) for being an automorphism. Applying A'' on the logical operators U_1, U_2, and W and the stabilizer G, we get

$$A''U_1 = \begin{bmatrix} 1+\overline{y} \\ 1+\overline{y} \\ \overline{\overline{y}+x\overline{y}+x} \\ \overline{y}+x\overline{y}+x \end{bmatrix}, \qquad A''U_2 = \begin{bmatrix} 1+\overline{x} \\ 0 \\ \overline{\overline{x}^2+x} \\ x \end{bmatrix}, \qquad (45)$$

$$A''(W+G) = \begin{bmatrix} \overline{x}\overline{y}+1 \\ \overline{x}\overline{y}+\overline{y} \\ \overline{\overline{x}\overline{y}+\overline{y}+\overline{x}+x} \\ \overline{y}+x \end{bmatrix}, A''G = \begin{bmatrix} \overline{x}\overline{y}+xy \\ \overline{x}\overline{y}+\overline{y}+y+xy \\ \overline{\overline{x}\overline{y}+\overline{y}+\overline{x}y+y+xy+x^2y} \\ \overline{y}+1+xy+x^2y \end{bmatrix}. \qquad (46)$$

The $A''(U_e)$ operators can be depicted as Eq. (16) and (17). Here, we choose $A''(W+G)$ as our flux operator shown in Eq. (18) because it has a lower weight $\mathrm{wt}[A''(W+G)] < \mathrm{wt}(A''W)$. The pictorial representation of stabilizer $A''G$ is Eq. (19).

3.3 Searching algorithm for automorphisms

In this subsection, we describe how we find automorphisms with code distances $d = 3, 4, 5, 6,$ and 7. The automorphisms A_1 in Eq. (32), A' in Eq. (38), and A'' in Eq. (44) correspond to the examples for $d = 3$, $d = 4$, and $d = 5$, respectively. We will show other examples with different code distances.

First, we consider 16 elementary automorphisms $A_1, A_2, \cdots, A_{16}$ (shown in Appendix B), which attach no more than one new Pauli matrix to the original Pauli matrix. For example, the A_1 automorphism attaches one Z to X_e, as shown in Eq. (33). Given the 16 elementary automorphisms, their product $A_{i_1} A_{i_2} A_{i_3} A_{i_4} \cdots$ with $i_n \in \{1, 2, \cdots, 16\}$ is also an automorphism. (We note that these elementary automorphisms do not generate all automorphisms.) We find that the product of five elementary automorphisms $A_{i_1} A_{i_2} A_{i_3} A_{i_4} A_{i_5}$ is sufficient to generate the code distance $d = 7$; therefore, we focus on products with five or fewer elementary automorphisms. We now describe how we search for automorphisms with large code distances:

1. We write down the bosonization of all the nearest-neighbor and on-site terms generated by S_e and P_f. They include the automorphism acting on U_1, U_2, W, $U_1 + W$, $U_1 + \overline{y}W$, $U_1 + \overline{y}W + W$, $U_2 + W$, $U_2 + \overline{x}W$, $U_2 + \overline{x}W + W$.[13]

2. We use the minimum weight of bosonization of nearest-neighbor hopping terms to roughly estimate the code distance of a given bosonization (automorphism) A.

3. We choose some candidate automorphisms with an appropriate minimum weight d, to find a bosonization with desired code distance d.

4. We apply the syndrome matching method shown in Appendix A to the candidate automorphisms. By applying syndrome matching, we find a lower bound of their code distances.

5. We apply the syndrome matching method for distance $d+1$ to an automorphism $\widetilde{A}$ with a lower bound d. If syndrome matching for distance $d + 1$ returns a logical operator with no syndrome, we conclude that $\widetilde{A}$ is a bosonization with code distance d.

---

[13] The stabilizer G can be added to any term.

Applying the syndrome-matching method, we found automorphisms to generate exact bosonization with $d = 3, 4, 5, 6, 7$ (see Table 2).

Code distance	Automorphisms
3	A_1 (3,5)
4	$A_4 A_7$ (5,6), $A_2 A_7 A_1$ (4,6)
5	$A_9 A_3 A_7 A_{14}$ (5,9)
6	$A_1 A_5 A_{14} A_1 (6, 13)$, $A_4 A_9 A_{16} A_{11}(7, 17)$
7	$A_1 A_{11} A_5 A_{14} A_9(7, 23)$

Table 2: The possible automorphisms for different code distances. The numbers inside parentheses are the minimum and maximum weights of the logical operators for the nearest-neighbor terms. For example, $A_9 A_3 A_7 A_{14}$ has the minimum logical weight 5 and the maximum logical weight 9.

4 Discussion

In this work, we introduce a method that employs Laurent polynomials and symplectic automorphisms as efficient classical computational tools in the search for fermion-to-qubit mappings with error correction capabilities. One significant advantage of this method lies in its ability to generate equivalent mappings with higher code distances while preserving the code rates. This is possible given a 2D fermion-to-qubit mapping, due to the established equivalence between various 2D fermion-to-qubit mappings, as referenced in [48].

The Laurent polynomials and symplectic automorphisms serve a dual purpose in our method: they are instrumental not only in the analytical derivation of quantum codes but are also valuable for the numerical studies of these codes. To illustrate the effectiveness of our approach, we demonstrate this method through examples of codes with distances $d = 3, 4, 5$ and further extend our constructions up to $d = 7$.

Additionally, we present general algorithms designed to systematically search for and verify fermion-to-qubit mappings with elevated code distances. It is noteworthy that our proposed method is not exclusive to fermion-to-qubit mappings; it is also applicable to regular qubit stabilizer codes, which are utilized to encode logical qubits.

Toward fault-tolerant fermionic quantum simulation, having error-correcting codes is not enough. Efficient and fault-tolerant gate sets are also important. We leave the investigations of fault-tolerant gate sets for future studies.

Acknowledgement

Note added.—As this work was being completed, we also became aware of an independent work, Ref. [60], using a similar polynomial technique to study fermionic quantum simulation from a different perspective. The other recent work Ref. [61] studied an error-mitigation scheme in fermionic encodings. The recent development of programmable neutral atom arrays in Ref. [62] can simulate fermionic modes directly, which avoids using any fermion-to-qubit mapping.

A Syndrome-matching method for finding code distances

In this appendix, we present an algorithm to find the code distance for a given stabilizer code. We call it the "syndrome matching" method. Specifically, given an integer n, we describe an algorithm to determine whether $d > n$. The algorithm is as follows.

1. We first choose once vertex to be the origin $(0,0)$. We then apply a single Pauli from X_1, Y_1, Z_1, X_2, Y_2, Z_2 (acting on qubits located on edges $(0,0) \to (1,0)$ and $(0,0) \to (0,1)$). We have 6 cases each of which has its own syndrome vertices, i.e., a set of vertices v that violate the stabilizer G_v. For a single-qubit error, the syndrome set is an ordered set $V^{(1)} = \{(x_1^{(1)}, y_1^{(1)}), (x_2^{(1)}, y_2^{(1)}), ...\}$, ordered by $x_i^{(1)}$: $x_1^{(1)} \leq x_2^{(1)} \leq \ldots$. If two vertices $i, i+1$ have the same $x_i^{(1)} = x_{i+1}^{(1)}$, they are ordered by $y_i^{(1)} < y_{i+1}^{(1)}$.

2. Ensure that the operator is logical. For this to be the case, all syndrome vertices should vanish. Therefore, we select the first syndrome vertex $(x_1^{(k)}, y_1^{(k)}) \in V^{(k)}$. Then we enumerate all choices of a Pauli matrix on an edge different from the Pauli matrix selected in the previous step(s), such that it cancels the syndrome at $(x_1^{(k)}, y_1^{(k)})$. This operation may generate other syndrome vertices. The syndrome vertices $V^{(k)}$ are updated due to this new Pauli matrix. At this stage, the operator has one more Pauli matrix, and a new ordered set for the syndrome vertices $V^{(k+1)}$. If the syndrome set $V^{(k+1)}$ is empty, this operator is logical. If the operator does not belong to the stabilizer group[14], this means that we have found the nontrivial logical operator with the minimum weight. This minimum weight is the code distance and, therefore, we stop the algorithm. Otherwise, we continue.

3. Repeat steps 1 and 2 until the algorithm stops automatically or all cases with operators containing n Pauli matrices have been considered, i.e., all $V^{(n)}$ are checked. If the algorithm stops automatically, it will return the value of the code distance. If the

---

[14]A way to check if a Pauli string operator O belongs to the stabilizer group is by computing $O \cdot (AW)$ and $O \cdot (AU_{1,2})$, where AW and $AU_{1,2}$ generate the full logical space since W and $U_{1,2}$ are the original generators in the exact bosonization and we apply an automorphism A on them. $O \cdot (AW) = O \cdot (AU_{1,2}) = 0$ if and only if the operator O commutes will all logical operators, which means $O \in \mathcal{G}$ ($\mathcal{G}$ is the stabilizer group).

algorithm stops by considering all the cases with n Pauli matrices, we conclude that code distance satisfies $d > n$.

B 16 elementary automorphisms

In this section, we demonstrate the transformation rules of Pauli matrices for 16 elementary automorphisms that we refer to as $A_1, ..., A_{16}$. These automorphism correspond to applying nearest-neighbor two-qubit Clifford gates. Here are the 16 elementary automorphisms:

$$A_1 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array}\right], \quad A_2 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & y & 1 & 0 \\ \bar{y} & 0 & 0 & 1 \end{array}\right], \tag{47}$$

$$A_3 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \bar{x} & 1 & 0 \\ x & 0 & 0 & 1 \end{array}\right], \quad A_4 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \bar{x}y & 1 & 0 \\ x\bar{y} & 0 & 0 & 1 \end{array}\right], \tag{48}$$

$$A_5 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & \bar{x}y \\ 0 & 1 & x\bar{y} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right], \quad A_6 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & y \\ 0 & 1 & \bar{y} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right], \tag{49}$$

$$A_7 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right], \quad A_8 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & \bar{x} \\ 0 & 1 & x & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right], \tag{50}$$

$$A_9 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}\right], \quad A_{10} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ x & 1 & 0 & 0 \\ 0 & 0 & 1 & \bar{x} \\ 0 & 0 & 0 & 1 \end{array}\right], \tag{51}$$

$$A_{11} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ \bar{y} & 1 & 0 & 0 \\ 0 & 0 & 1 & y \\ 0 & 0 & 0 & 1 \end{array}\right], \quad A_{12} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ x\bar{y} & 1 & 0 & 0 \\ 0 & 0 & 1 & \bar{x}y \\ 0 & 0 & 0 & 1 \end{array}\right], \tag{52}$$

$$A_{13} = \left[\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right], \quad A_{14} = \left[\begin{array}{cc|cc} 1 & \bar{x} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & x & 1 \end{array}\right], \tag{53}$$

$$A_{15} = \left[\begin{array}{cc|cc} 1 & y & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \bar{y} & 1 \end{array}\right], \quad A_{16} = \left[\begin{array}{cc|cc} 1 & \bar{x}y & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & x\bar{y} & 1 \end{array}\right]. \tag{54}$$

The diagonal blocks of A_1 are identities, and its nontrivial part is the lower left block which attaches an extra Z to X_1 and X_2. By multiplying the polynomial vectors of X_1, X_2, Z_1, and Z_2 by automorphism A_1, we get the following terms:

$$
-X_e- \ \xrightarrow{A_1} \ \underset{-X_e-}{Z} \quad , \qquad X_e \ \xrightarrow{A_1} \ \underset{-Z-}{X_e} \quad ,
$$

$$
-Z_e- \ \xrightarrow{A_1} \ -Z_e- \ , \qquad Z_e \ \xrightarrow{A_1} \ Z_e. \tag{55}
$$

Similarly, we may multiply X_1, X_2, Z_1, and Z_2 by A_2, thereby obtaining

$$
-X_e- \ \xrightarrow{A_2} \ \overset{-X_e-}{Z} \quad , \qquad X_e \ \xrightarrow{A_2} \ \overset{-Z-}{X_e} \quad ,
$$

$$
-Z_e- \ \xrightarrow{A_2} \ -Z_e- \ , \qquad Z_e \ \xrightarrow{A_2} \ Z_e. \tag{56}
$$

Following the same argument, we can obtain pictorial representations for the rest of the automorphisms:

A_3:

$$
-X_e- \ \xrightarrow{A_3} \ \underset{-X_e-}{Z}, \qquad X_e \ \xrightarrow{A_3} \ \underset{-Z-}{X_e}
$$

$$
-Z_e- \ \xrightarrow{A_3} \ -Z_e- \ , \qquad Z_e \ \xrightarrow{A_3} \ Z_e \tag{57}
$$

A_4:

$$
-X_e- \ \xrightarrow{A_4} \ \overset{-X_e-}{Z}, \qquad X_e \ \xrightarrow{A_4} \ \overset{-Z-}{X_e}
$$

$$
-Z_e- \ \xrightarrow{A_4} \ -Z_e- \ , \qquad Z_e \ \xrightarrow{A_4} \ Z_e \tag{58}
$$

A_5:

$$
-X_e- \ \xrightarrow{A_5} \ -X_e- \ , \qquad X_e \ \xrightarrow{A_5} \ X_e
$$

$$
-Z_e- \ \xrightarrow{A_5} \ \underset{-Z_e-}{X}, \qquad Z_e \ \xrightarrow{A_5} \ \overset{-X-}{Z_e} \tag{59}
$$

A_6:

$$-X_e- \quad \xrightarrow{A_6} \quad -X_e- \ , \qquad \overset{|}{\underset{|}{X}}_e \quad \xrightarrow{A_6} \quad \overset{|}{\underset{|}{X}}_e$$

$$-Z_e- \quad \xrightarrow{A_6} \quad \overset{-Z_e-}{\underset{|}{X}} \ , \qquad \overset{|}{\underset{|}{Z}}_e \quad \xrightarrow{A_6} \quad \overset{-X-}{\underset{|}{Z}}_e$$

(60)

A_7:

$$-X_e- \quad \xrightarrow{A_7} \quad -X_e- \ , \qquad \overset{|}{\underset{|}{X}}_e \quad \xrightarrow{A_7} \quad \overset{|}{\underset{|}{X}}_e$$

$$-Z_e- \quad \xrightarrow{A_7} \quad \overset{|}{\underset{-Z_e-}{X}} \ , \qquad \overset{|}{\underset{|}{Z}}_e \quad \xrightarrow{A_7} \quad \overset{|}{\underset{-X-}{Z}}_e$$

(61)

A_8:

$$-X_e- \quad \xrightarrow{A_8} \quad -X_e- \ , \qquad \overset{|}{\underset{|}{X}}_e \quad \xrightarrow{A_8} \quad \overset{|}{\underset{|}{X}}_e$$

$$-Z_e- \quad \xrightarrow{A_8} \quad \overset{}{\underset{-Z_e-}{X}} \ , \qquad \overset{|}{\underset{|}{Z}}_e \quad \xrightarrow{A_8} \quad \overset{}{\underset{-X-}{Z}}_e$$

(62)

A_9:

$$-X_e- \quad \xrightarrow{A_9} \quad \overset{|}{\underset{-X_e-}{X}} \ , \qquad \overset{|}{\underset{|}{X}}_e \quad \xrightarrow{A_9} \quad \overset{|}{\underset{|}{X}}_e$$

$$-Z_e- \quad \xrightarrow{A_9} \quad -Z_e- \ , \qquad \overset{|}{\underset{|}{Z}}_e \quad \xrightarrow{A_9} \quad \overset{|}{\underset{-Z-}{Z}}_e$$

(63)

A_{10}:

$$-X_e- \quad \xrightarrow{A_{10}} \quad \overset{|}{\underset{-X_e-}{X}} \ , \qquad \overset{|}{\underset{|}{X}}_e \quad \xrightarrow{A_{10}} \quad \overset{|}{\underset{|}{X}}_e$$

$$-Z_e- \quad \xrightarrow{A_{10}} \quad -Z_e- \ , \qquad \overset{|}{\underset{|}{Z}}_e \quad \xrightarrow{A_{10}} \quad \overset{|}{\underset{-Z-}{Z}}_e$$

(64)

A_{11}:

$$
\begin{array}{ccc}
-X_e- & \xrightarrow{A_{11}} & \overset{-X_e-}{X} \;, \qquad X_e \xrightarrow{A_{11}} X_e \\[2em]
-Z_e- & \xrightarrow{A_{11}} & -Z_e- \;, \qquad Z_e \xrightarrow{A_{11}} \overset{-Z-}{Z_e}
\end{array}
\tag{65}
$$

A_{12}:

$$
\begin{array}{ccc}
-X_e- & \xrightarrow{A_{12}} & \overset{-X_e-}{X} \;, \qquad X_e \xrightarrow{A_{12}} X_e \\[2em]
-Z_e- & \xrightarrow{A_{12}} & -Z_e- \;, \qquad Z_e \xrightarrow{A_{12}} \overset{-Z-}{Z_e}
\end{array}
\tag{66}
$$

A_{13}:

$$
\begin{array}{ccc}
-X_e- & \xrightarrow{A_{13}} & -X_e- \;, \qquad X_e \xrightarrow{A_{13}} \underset{-X-}{X_e} \\[2em]
-Z_e- & \xrightarrow{A_{13}} & \underset{-Z_e-}{Z} \;, \qquad Z_e \xrightarrow{A_{13}} Z_e
\end{array}
\tag{67}
$$

A_{14}:

$$
\begin{array}{ccc}
-X_e- & \xrightarrow{A_{14}} & -X_e- \;, \qquad X_e \xrightarrow{A_{14}} \underset{-X-}{X_e} \\[2em]
-Z_e- & \xrightarrow{A_{14}} & \underset{-Z_e-}{Z} \;, \qquad Z_e \xrightarrow{A_{14}} Z_e
\end{array}
\tag{68}
$$

A_{15}:

$$
\begin{array}{ccc}
-X_e- & \xrightarrow{A_{15}} & -X_e- \;, \qquad X_e \xrightarrow{A_{15}} \overset{-X-}{X_e} \\[2em]
-Z_e- & \xrightarrow{A_{15}} & \overset{-Z_e-}{Z} \;, \qquad Z_e \xrightarrow{A_{15}} Z_e
\end{array}
\tag{69}
$$

A_{16}:

$$-X_e- \quad \xrightarrow{A_{16}} \quad -X_e- \quad , \qquad \overset{\displaystyle |}{\underset{\displaystyle |}{X}}_e \quad \xrightarrow{A_{16}} \quad \begin{array}{c} -X- \\ | \\ X_e \\ | \end{array}$$

(70)

$$-Z_e- \quad \xrightarrow{A_{16}} \quad \begin{array}{c} -Z_e- \\ | \\ Z \\ | \end{array}, \qquad \overset{\displaystyle |}{\underset{\displaystyle |}{Z}}_e \quad \xrightarrow{A_{16}} \quad \overset{\displaystyle |}{\underset{\displaystyle |}{Z}}_e$$

C Automorphisms for code distances $d = 6$ and $d = 7$

In this section, we show the explicit form of automorphisms $A^{d=6}$ and $A^{d=7}$ found by syndrome matching. The automorphism $A^{d=6} = A_1 A_5 A_{14} A_1$ has a code distance of 6:

$$A^{d=6} = \left[\begin{array}{cc|cc} 1 + \overline{x}y & \overline{x} + y & y & \overline{x}y \\ 0 & 1 + x\overline{y} & x\overline{y} & 0 \\ \hline 0 & x\overline{y} & 1 + x\overline{y} & 0 \\ \overline{x}y & \overline{x} + x + y & x + y & 1 + \overline{x}y \end{array} \right].$$

(71)

By applying $A^{d=6}$ on logical operators U_1, U_2, W, and stabilizer G, we obtain their polynomial representations as follows:

$$A^{d=6}U_1 = \left[\begin{array}{c} \overline{x} + 1 + \overline{x}y \\ 0 \\ \hline 0 \\ \overline{y} + \overline{x} + \overline{x}y \end{array} \right],$$

(72)

$$A^{d=6}U_2 = \left[\begin{array}{c} \overline{x} + \overline{x}y + y \\ \overline{y} + x\overline{y} + 1 \\ \hline \overline{y} + x\overline{y} + \overline{x} \\ \overline{x} + 1 + x + \overline{x}y + y \end{array} \right],$$

(73)

$$A^{d=6}W = \left[\begin{array}{c} \overline{x}y + y^2 \\ x\overline{y} + x \\ \hline x\overline{y} + 1 + x + y \\ 1 + \overline{x}y + xy + y^2 \end{array} \right],$$

(74)

$$A^{d=6}G = \left[\begin{array}{c} \overline{xy} + \overline{x}^2 y + y + y^2 \\ x\overline{y}^2 + \overline{y} + 1 + x \\ \hline x\overline{y}^2 + 1 + x + y \\ \overline{xy} + x\overline{y} + \overline{x} + x + \overline{x}^2 y + y + xy + y^2 \end{array} \right].$$

(75)

The automorphism $A^{d=7} = A_1 A_{11} A_5 A_{14} A_9$ with distance $d = 7$ is

$$A^{d=7} = \left[\begin{array}{cc|cc} \overline{x} + 1 & \overline{x} & y & \overline{x}y + y \\ \overline{xy} + \overline{y} + 1 & \overline{xy} + 1 & x\overline{y} + 1 & x\overline{y} + \overline{x} + 1 \\ \hline \overline{xy} + \overline{y} + 1 & \overline{xy} + 1 & x\overline{y} + xy & x\overline{y} + \overline{x} + y + xy \\ \overline{x} + 1 & \overline{x} & x + y & 1 + x + \overline{x}y + y \end{array} \right].$$

(76)

By applying $A^{d=7}$ on logical operatrs U_1, U_2, W, and stabilizer G, we can write down their polynomial representations as follows:

$$A^{d=7}U_1 = \begin{bmatrix} 0 \\ \dfrac{x\overline{y}^2+1}{x\overline{y}^2+\overline{y}+x} \\ \overline{y}+x\overline{y} \end{bmatrix}, \quad A^{d=7}U_2 = \begin{bmatrix} \overline{x}+\overline{x}y \\ \dfrac{\overline{xy}+\overline{y}+\overline{x}+1}{\overline{xy}+\overline{y}+1+y} \\ \overline{x}+1+\overline{x}y \end{bmatrix}, \tag{77}$$

$$A^{d=7}W = \begin{bmatrix} \overline{x}y+y+xy+y^2 \\ \dfrac{x^2\overline{y}+\overline{x}+1+y}{x^2\overline{y}+\overline{x}+1+x+y+xy+x^2y+xy^2} \\ 1+x+x^2+\overline{x}y+y+y^2 \end{bmatrix}, \tag{78}$$

$$A^{d=7}G = \begin{bmatrix} \overline{xy}+\overline{x}^2+\overline{x}+1+\overline{x}y+y+xy+xy^2 \\ \dfrac{\overline{xy}^2+\overline{x}^2\overline{y}+\overline{xy}+x\overline{y}+1+y}{\overline{xy}^2+\overline{x}^2\overline{y}+\overline{xy}+x^2\overline{y}} \\ +1+x+y+xy+x^2y+xy^2 \\ \overline{xy}+\overline{x}^2+\overline{x}+x+x^2+\overline{x}y+y+y^2 \end{bmatrix}. \tag{79}$$

.

References

[1] D. Gottesman, *Stabilizer codes and quantum error correction*, California Institute of Technology (1997).

[2] P. Panteleev and G. Kalachev, *Quantum ldpc codes with almost linear minimum distance*, IEEE Transactions on Information Theory **68**(1), 213 (2022), doi:10.1109/TIT.2021.3119384.

[3] 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*, pp. 375–388 (2022).

[4] T.-C. Lin and M.-H. Hsieh, *Good quantum ldpc codes with linear time decoder from lossless expanders*, arXiv preprint arXiv:2203.03581 (2022).

[5] I. Dinur, M.-H. Hsieh, T.-C. Lin and T. Vidick, *Good quantum ldpc codes with linear time decoders*, In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pp. 905–918 (2023).

[6] P. Jordan and E. P. Wigner, *Über das paulische äquivalenzverbot*, Springer (1993).

[7] E. Fradkin, *Jordan-wigner transformation for quantum-spin systems in two dimensions and fractional statistics*, Phys. Rev. Lett. **63**, 322 (1989), doi:10.1103/PhysRevLett.63.322.

[8] S. B. Bravyi and A. Y. Kitaev, *Fermionic quantum computation*, Annals of Physics **298**(1), 210 (2002), doi:https://doi.org/10.1006/aphy.2002.6254.

[9] R. Barends, L. Lamata, J. Kelly, L. García-Álvarez, A. G. Fowler, A. Megrant, E. Jeffrey, T. C. White, D. Sank, J. Y. Mutus, B. Campbell, Y. Chen *et al.*, *Digital quantum simulation of fermionic models with a superconducting circuit*, Nature Communications **6**(1), 7654 (2015), doi:10.1038/ncomms8654.

[10] T. Hensgens, T. Fujita, L. Janssen, X. Li, C. Van Diepen, C. Reichl, W. Wegscheider, S. Das Sarma and L. M. Vandersypen, *Quantum simulation of a fermi–hubbard model using a semiconductor quantum dot array*, Nature **548**(7665), 70 (2017), doi:10.1038/nature23022.

[11] A. Mazurenko, C. S. Chiu, G. Ji, M. F. Parsons, M. Kanász-Nagy, R. Schmidt, F. Grusdt, E. Demler, D. Greif and M. Greiner, *A cold-atom fermi–hubbard antiferromagnet*, Nature **545**(7655), 462 (2017), doi:10.1038/nature22362.

[12] L. Tarruell and L. Sanchez-Palencia, *Quantum simulation of the hubbard model with ultracold fermions in optical lattices* (2019), 1809.00571.

[13] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, A. Bengtsson, S. Boixo, M. Broughton, B. B. Buckley *et al.*, *Observation of separated dynamics of charge and spin in the fermi-hubbard model*, arXiv preprint arXiv:2010.07965 (2020).

[14] B. Hebbe Madhusudhana, S. Scherg, T. Kohlert, I. Bloch and M. Aidelsburger, *Benchmarking a novel efficient numerical method for localized 1d fermi-hubbard systems on a quantum simulator*, PRX Quantum **2**, 040325 (2021), doi:10.1103/PRXQuantum.2.040325.

[15] A. Kitaev, *Fault-tolerant quantum computation by anyons*, Annals of Physics **303**(1), 2–30 (2003), doi:10.1016/s0003-4916(02)00018-0.

[16] A. G. Fowler, M. Mariantoni, J. M. Martinis and A. N. Cleland, *Surface codes: Towards practical large-scale quantum computation*, Phys. Rev. A **86**, 032324 (2012), doi:10.1103/PhysRevA.86.032324.

[17] K. Setia, S. Bravyi, A. Mezzacapo and J. D. Whitfield, *Superfast encodings for fermionic quantum simulation*, Phys. Rev. Research **1**, 033033 (2019), doi:10.1103/PhysRevResearch.1.033033.

[18] R. C. Ball, *Fermions without fermion fields*, Phys. Rev. Lett. **95**, 176407 (2005), doi:10.1103/PhysRevLett.95.176407.

[19] F. Verstraete and J. I. Cirac, *Mapping local hamiltonians of fermions to local hamiltonians of spins*, Journal of Statistical Mechanics: Theory and Experiment **2005**(09), P09012 (2005), doi:10.1088/1742-5468/2005/09/p09012.

[20] J. D. Whitfield, V. Havlíček and M. Troyer, *Local spin operators for fermion simulations*, Phys. Rev. A **94**, 030301 (2016), doi:10.1103/PhysRevA.94.030301.

[21] M. Steudtner and S. Wehner, *Quantum codes for quantum simulation of fermions on a square lattice of qubits*, Phys. Rev. A **99**, 022308 (2019), doi:10.1103/PhysRevA.99.022308.

[22] C. Derby, J. Klassen, J. Bausch and T. Cubitt, *Compact fermion to qubit mappings*, Phys. Rev. B **104**, 035118 (2021), doi:10.1103/PhysRevB.104.035118.

[23] H. C. Po, *Symmetric jordan-wigner transformation in higher dimensions*, arXiv preprint arXiv:2107.10842 (2021).

[24] Y.-A. Chen, A. Kapustin and D. Radicevic, *Exact bosonization in two spatial dimensions and a new class of lattice gauge theories*, Annals of Physics **393**, 234 (2018), doi:https://doi.org/10.1016/j.aop.2018.03.024.

[25] Y.-A. Chen and A. Kapustin, *Bosonization in three spatial dimensions and a 2-form gauge theory*, Phys. Rev. B **100**, 245127 (2019), doi:10.1103/PhysRevB.100.245127.

[26] Y.-A. Chen, *Exact bosonization in arbitrary dimensions*, Phys. Rev. Research **2**, 033527 (2020), doi:10.1103/PhysRevResearch.2.033527.

[27] J. Wosiek, *A local representation for fermions on a lattice*, Tech. rep., Univ., Physics Department (1981).

[28] A. Bochniak, B. Ruba, J. Wosiek and A. Wyrzykowski, *Constraints of kinematic bosonization in two and higher dimensions*, Physical Review D **102**(11), 114502 (2020).

[29] A. Bochniak and B. Ruba, *Bosonization based on clifford algebras and its gauge theoretic interpretation*, Journal of High Energy Physics **2020**(12), 1 (2020).

[30] A. Bochniak, B. Ruba and J. Wosiek, *Bosonization of majorana modes and edge states*, Physical Review B **105**(15), 155105 (2022).

[31] H. Bombin, *Topological order with a twist: Ising anyons from an abelian model*, Phys. Rev. Lett. **105**, 030403 (2010), doi:10.1103/PhysRevLett.105.030403.

[32] M. B. Hastings and A. Geller, *Reduced space-time and time costs using dislocation codes and arbitrary ancillas*, arXiv preprint arXiv:1408.3379 (2014).

[33] B. J. Brown, K. Laubscher, M. S. Kesselring and J. R. Wootton, *Poking holes and cutting corners to achieve clifford gates with the surface code*, Phys. Rev. X **7**, 021029 (2017), doi:10.1103/PhysRevX.7.021029.

[34] *Non-abelian braiding of graph vertices in a superconducting processor*, Nature pp. 1–6 (2023).

[35] M. Steudtner and S. Wehner, *Fermion-to-qubit mappings with varying resource requirements for quantum simulation*, New Journal of Physics **20**(6), 063010 (2018), doi:10.1088/1367-2630/aac54f.

[36] M. Steudtner and S. Wehner, *Quantum codes for quantum simulation of fermions on a square lattice of qubits*, Phys. Rev. A **99**, 022308 (2019), doi:10.1103/PhysRevA.99.022308.

[37] R. W. Chien and J. D. Whitfield, *Custom fermionic codes for quantum simulation*, arXiv preprint arXiv:2009.11860 (2020).

[38] Z. Jiang, A. Kalev, W. Mruczkiewicz and H. Neven, *Optimal fermion-to-qubit mapping via ternary trees with applications to reduced quantum states learning*, Quantum **4**, 276 (2020).

[39] J. Bausch, T. Cubitt, C. Derby and J. Klassen, *Mitigating errors in local fermionic encodings*, arXiv preprint arXiv:2003.07125 (2020).

[40] M. Chiew and S. Strelchuk, *Optimal fermion-qubit mappings*, arXiv preprint arXiv:2110.12792 (2021).

[41] C. Derby and J. Klassen, *A compact fermion to qubit mapping part 2: Alternative lattice geometries*, arXiv preprint arXiv:2101.10735 (2021).

[42] A. J. Landahl and B. C. Morrison, *Logical majorana fermions for fault-tolerant quantum simulation*, arXiv preprint arXiv:2110.10280 (2021).

[43] B. Harrison, D. Nelson, D. Adamiak and J. Whitfield, *Reducing the qubit requirement of jordan-wigner encodings of n-mode, k-fermion systems from n to $\lceil \log_2 \binom{N}{K} \rceil$* (2023), 2211.04501.

[44] O. O'Brien and S. Strelchuk, *Ultrafast hybrid fermion-to-qubit mapping*, arXiv preprint arXiv:2211.16389 (2022).

[45] W. Cao, M. Yamazaki and Y. Zheng, *Boson-fermion duality with subsystem symmetry*, Phys. Rev. B **106**, 075150 (2022), doi:10.1103/PhysRevB.106.075150.

[46] Q.-S. Li, H.-Y. Liu, Q. Wang, Y.-C. Wu and G.-P. Guo, *A unified framework of transformations based on the jordan–wigner transformation*, The Journal of Chemical Physics **157**(13), 134104 (2022).

[47] M. G. Algaba, P. V. Sriluckshmy, M. Leib and F. S. I. au2, *Low-depth simulations of fermionic systems on square-grid quantum hardware* (2023), 2302.01862.

[48] Y.-A. Chen and Y. Xu, *Equivalence between fermion-to-qubit mappings in two spatial dimensions*, PRX Quantum **4**, 010326 (2023), doi:10.1103/PRXQuantum.4.010326.

[49] A. Kitaev, *Anyons in an exactly solved model and beyond*, Annals of Physics **321**(1), 2 (2006).

[50] X.-G. Wen, *Quantum orders in an exact soluble model*, Phys. Rev. Lett. **90**, 016803 (2003), doi:10.1103/PhysRevLett.90.016803.

[51] J. Nys and G. Carleo, *Variational solutions to fermion-to-qubit mappings in two spatial dimensions*, Quantum **6**, 833 (2022).

[52] J. Nys and G. Carleo, *Quantum circuits for solving local fermion-to-qubit mappings*, Quantum **7**, 930 (2023).

[53] A. Rajput, A. Roggero and N. Wiebe, *Quantum error correction with gauge symmetries*, arXiv preprint arXiv:2112.05186 (2021).

[54] Z. Jiang, J. McClean, R. Babbush and H. Neven, *Majorana loop stabilizer codes for error mitigation in fermionic quantum simulations*, Phys. Rev. Applied **12**, 064041 (2019), doi:10.1103/PhysRevApplied.12.064041.

[55] J. Haah, *Commuting pauli hamiltonians as maps between free modules*, Communications in Mathematical Physics **324**(2), 351 (2013), doi:10.1007/s00220-013-1810-2.

[56] J. Haah, *Algebraic methods for quantum codes on lattices*, Revista Colombiana de Matemáticas **50**(2), 299 (2016), doi:10.15446/recolma.v50n2.62214.

[57] B. Ruba and B. Yang, *Homological invariants of pauli stabilizer codes*, doi:10.48550/ARXIV.2204.06023 (2022).

[58] N. Tantivasadakarn, *Jordan-wigner dualities for translation-invariant hamiltonians in any dimension: Emergent fermions in fracton topological order*, Phys. Rev. Research **2**, 023353 (2020), doi:10.1103/PhysRevResearch.2.023353.

[59] J. Haah, *Clifford quantum cellular automata: Trivial group in 2d and witt group in 3d*, Journal of Mathematical Physics **62**(9), 092202 (2021), doi:10.1063/5.0022185, https://doi.org/10.1063/5.0022185.

[60] R. W. Chien and J. Klassen, *Optimizing fermionic encodings for both hamiltonian and hardware* (2022), 2210.05652.

[61] R. W. Chien, K. Setia, X. Bonet-Monroig, M. Steudtner and J. D. Whitfield, *Simulating quantum error mitigation in fermionic encodings*, arXiv preprint arXiv:2303.02270 (2023).

[62] 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*, arXiv preprint arXiv:2303.06985 (2023).