Families of $d = 2$ 2D subsystem stabilizer codes for universal Hamiltonian quantum computation with two-body interactions

Phattharaporn Singkanipa[1,2], Zihan Xia[1,3], and Daniel A. Lidar[1,2,3,4]

[1]Center for Quantum Information Science & Technology
[2]Department of Physics & Astronomy
[3]Department of Electrical & Computer Engineering
[4]Department of Chemistry
University of Southern California, Los Angeles, California 90089, USA

In the absence of fault tolerant quantum error correction for analog, Hamiltonian quantum computation, error suppression via energy penalties is an effective alternative. We construct families of distance-2 stabilizer subsystem codes we call "trapezoid codes", that are tailored for energy-penalty schemes. We identify a family of codes achieving the maximum code rate, and by slightly relaxing this constraint, uncover a broader range of codes with enhanced physical locality, thus increasing their practical applicability. Additionally, we provide an algorithm to map the required qubit connectivity graph into graphs compatible with the locality constraints of quantum hardware. Finally, we provide a systematic framework to evaluate the performance of these codes in terms of code rate, physical locality, graph properties, and penalty gap, enabling an informed selection of error-suppression codes for specific quantum computing applications. We identify the $[[4k+2, 2k, g, 2]]$ family of subsystem codes as optimal in terms of code rate and penalty gap scaling.

1 Introduction

Hamiltonian quantum computation, which encompasses approaches such as adiabatic quantum computing (AQC) [1–3], quantum annealing [4–6], holonomic quantum computing [7–9], quantum simulation [10–12], and continuous-time quantum walks [13–15], operates by continuously modifying the parameters of a Hamiltonian. In the adiabatic case, for example, the goal is to maintain a quantum system in its ground state while evolving the system's Hamiltonian from a simple initial form to a complex final one. The solution to a given problem is encoded in the ground state of the final Hamiltonian by design. However, integrating fault tolerant quantum error correction into Hamiltonian-based computation poses a significant challenge due to its analog nature, which complicates the introduction of low-entropy ancilla qubits throughout the computation, as well as syndrome measurements, among other issues. Solutions have been proposed in some cases, e.g., holonomic quantum computation [16], but this involves introducing elements of the circuit model.

To protect Hamiltonian computation, in particular the adiabatic model, Ref. [17] introduced a method that employs stabilizer subspace codes [18, 19] to suppress decoherence. In this approach, the Hamiltonian is encoded by replacing each of its Pauli operators with the corresponding logical Pauli operator of the chosen code, and the code stabilizers are used to construct a penalty Hamiltonian that is added and suppresses the errors the code is designed to detect. Since the penalty Hamiltonian is a sum of commuting terms, it is automatically gapped [20]. This results in the suppression of thermal excitations out of the ground subspace, similar to the error suppression provided by topological quantum computing [21, 22]. The approach pioneered in Ref. [17] (see also Refs. [23–30]) involves 4-local interactions, where by 'n-local' we mean an

n-body (or weight-n) interaction, regardless of the geometry of the interaction graph; later we refer to the latter as 'geometric locality.' Since 4-local interactions are not naturally available (though possible to generate [31–34]), a practical implementation becomes difficult and an alternative based on naturally available 2-local interactions is desirable. Unfortunately, a purely 2-local scheme based on subspace codes that suppresses arbitrary 1-local errors is impossible [35].

Circumventing this no-go theorem, it was shown [36, 37] that one can construct non-commuting 2-local Hamiltonians capable of suppressing general 1-local errors using stabilizer *subsystem* codes [38]. However, the Hamiltonians found in Refs. [36, 37] are insufficient for universality. This obstacle was overcome in Ref. [39], which presented a 2-local construction of a subsystem code whose error-suppressing gauge operators, single-qubit logical operators, and products of two-qubit logical operators (of the same type) are all 2-local, and which showed how this code can be used to encode universal AQC while suppressing all 1-local errors. This universality result still leaves several important unanswered questions related primarily to the optimal achievable code rate, geometric locality, and the embedding of the interaction graph in physical hardware connectivity graphs.

To properly explain these issues, let us first provide some more of the technical background. We wish to protect Hamiltonian quantum computation performed by a system Hamiltonian $H_S(t)$ against the system-bath interaction. The encoded Hamiltonian, $H_S^{\text{enc}}(t)$, is constructed by replacing every operator in $H_S(t)$ with the corresponding logical operators. To protect the computational space, a penalty term $\epsilon_P H_P$ is added to the encoded Hamiltonian, where $\epsilon_P > 0$ is the energy penalty. The penalty Hamiltonian, H_P, is constructed using the elements of the gauge group of the subsystem code, $H_P = -\sum_{g_i \in \mathcal{G}} g_i$. In the large penalty limit, the system becomes decoupled from the bath and performs the desired encoded computation [37], even though in the subsystem case $[H_S^{\text{enc}}(t), H_P] \neq 0$ (unlike the subspace case, where the encoded Hamiltonian commutes with the penalty term). Since the penalty Hamiltonian is now a sum of non-commuting gauge operators, an important disadvantage of subsystem codes is that (unlike in the subspace codes case) their penalty Hamiltonians are no longer necessarily gapped, a fact that complicates their analysis.

Of particular interest is a family of $[[6k, 2k, 2]]$ stabilizer codes introduced as subspace codes in Ref. [26] and studied as subsystem codes by Marvian and Lloyd [39], which enable universal encoded computation within this energy-penalty framework. The $[[6k, 2k, 2]]$ code family studied in [39] can encode a geometrically local Hamiltonian into a new, geometrically local encoded Hamiltonian using all 2-local operators with a code rate of $2k$ logical qubit per $6k$ physical qubits (i.e., a rate of $1/3$). A natural question that arises is whether this rate can be improved while maintaining the highly desirable properties of 2-local interactions and geometric locality. Various additional tradeoffs arise that contribute to the practical implementation of subsystem codes.

Here, we study subsystem codes within the energy-penalty framework and systematically study these tradeoffs in terms of a set of criteria outlined below. A key tool in our study is Bravyi's 'A matrix' construction of subsystem codes [40]. As our central result, we generalize the $[[6k, 2k, 2]]$ code to a new family of $[[4k + 2l, 2k, g, 2]]$ codes, for $k \in \mathbb{Z}^+$, $l \in \{1, \cdots, k\}$, for which the the best rate achievable is $k/(2k+1) \geq 1/3$. We demonstrate that every code from this family can have all 2-local one-qubit logical operators and two-qubit logical operators. Additionally, we introduce an optimization algorithm to map the graph derived from logical operators and gauges to one aligned with hardware connectivity, employing the total Manhattan distance as a performance metric for the resulting graphs.

An interesting question is whether the methods we develop here can be extended to more general codes, with distance greater than 2. Unfortunately, codes with distance $d > 2$ would make it impossible to work exclusively with 2-local Hamiltonians, which is a key requirement that we impose and center our work around. Hence, we do not pursue such an extension here.

The structure of this paper is as follows: Section 2 gives an overview of our construction. To establish terminology and notation, Section 3 provides a brief introduction to subspace stabilizer codes, subsystem stabilizer codes and their construction using Bravyi's A matrix. In Sec-

tion 4, we construct the new family of codes mentioned above along with the corresponding set of optimal set of logical operators, and calculate the energy gap of the penalty Hamiltonian. Section 5 presents our optimization algorithm for alignment with hardware connectivity. Section 6 provides two detailed examples illustrating the entire pipeline of our construction and results. Section 7 discusses the various criteria and metrics guiding our code selection. Finally, Section 8 summarizes our findings. The Appendix contains technical details that complement the main text.

2 Construction and Desiderata

We first briefly describe the construction we employ in this work in very general terms, with details and full definitions provided later.

We start with a system Hamiltonian $H_S(t)$ defined in terms of qubits occupying the vertices V_S of a graph $G_S(V_S, E_S)$ whose edges E_S support the interactions between the qubits:

$$H_S(t) = \sum_i a_i X_i + \sum_i b_i Z_i + \sum_{(i,j)\in E_S^{XX}} c_{ij} X_i X_j + \sum_{(i,j)\in E_S^{ZZ}} d_{ij} Z_i Z_j, \tag{1}$$

where X_i and Z_i are the Pauli X and Z matrices acting on the i'th qubit, with identity acting on all other qubits. The sets E_S^{XX} and E_S^{ZZ} denote edges supporting the XX and ZZ interactions, respectively. Hamiltonians of the form given in Eq. (1) are universal for AQC, where the coefficients a_i, b_i, c_{ij}, and d_{ij} are all assumed to be smoothly time-dependent and fully controllable [3, 41]. A more restricted form with all $c_{ij} = 0$ (transverse field Ising model) is commonly used in quantum annealing, where the ground state at the final time is the solution to an optimization problem, typically involving an adiabatic evolution from large initial a_i and small b_i, d_{ij} to small final a_i and large b_i, d_{ij} [4, 42–47].

Second, to protect the computation, this system Hamiltonian is replaced with an encoded Hamiltonian $\hat{H}_S(t)$ where every physical Pauli operator σ_i^α is replaced with a dressed logical operator $\hat{\sigma}_i^\alpha$ while preserving the same graph G_S (we clarify the 'dressed' terminology in Section 3.2). The vertices of G_S are logical qubits.

Third, the dressed logical operators and gauge group elements are all replaced by 2-local physical operators. The latter occupy the vertices V of a new graph we call the 'induced graph' $G(V, E)$ [We use the standard notation $G(V, E)$ to denote a graph G with vertices V and edge set E.]. Unlike in the second step, the vertices of G are physical qubits. To this is added a penalty term $\epsilon_P H_P$, where the penalty Hamiltonian H_P, consisting of a sum of gauge group elements, also lives on the induced graph G.

Finally, we map the induced graph to a new graph we call the 'mapped graph' G_m, which represents the hardware graph of a physical device. The vertices of G_m are physical qubits. This step can be considered as a permutation of physical qubits on a device such that the connectivity of the device aligns more with that of G.

Next, we state five criteria for code selection: code rate, physical locality, induced graph properties, mapped graph properties, and penalty gap. These criteria motivate this work in terms of selecting subsystem codes that optimally balance the various desiderata that guide penalty-protected Hamiltonian computing.

In stating these criteria we need two technical terms: (1) the *degree* of a qubit is the number of edges from this qubit to other qubits in the same graph; (2) the *geometric locality* of a pair of qubits is 1 plus the shortest path between them, i.e., 1 plus the smallest number of edges separating them. E.g., two qubits separated by a single edge are geometrically 2-local.

1. Code rate: With a fixed code distance $d = 2$ and a 2D geometry, we aim to maximize the code rate k/n of $[[n, k, d]]$ (subsystem) stabilizer codes.

2. Physical Locality: Ideally, at most two-body interactions should be used, i.e., the Hamiltonian should have at most 2-local terms even after encoding. If this criterion turns out to be impossible to satisfy, then the physical locality should be minimized, i.e., the number of k-local terms with $k > 2$ should be as small as possible.

3. Induced graph properties: The induced graph is geometrically 2-local and should be of fixed and low degree. These requirements arise from the typical constraints of solid-state implementations of quantum comput-

ers; the degree requirement can be relaxed for some systems such as trapped atoms or ions, which can support all-to-all qubit connectivity.

The induced graph should ideally be balanced. Here, by 'balanced' we mean that each qubit has roughly the same degree.

4. Mapped graph properties: In general, the induced graph will be quite different from the qubit connectivity graphs of actual devices. To account for this, we map the induced graph to a variety of more practical graphs that correspond to feasible connectivity graphs, in some cases corresponding to actual experimental implementations. Like the induced graph, the mapped graph should be geometrically local, of low degree, and balanced.

5. Penalty Gap: The 'penalty gap' is the energy difference between the ground state and the first excited state of the penalty Hamiltonian H_P. Since the encoded Hamiltonian performs the same desired computation as the original system Hamiltonian only in the large penalty limit, a larger penalty gap is preferred. The size of the penalty gap will depend on the specific code.

We revisit these criteria in Section 7 and comment on how the different codes we study satisfy them. First, we provide the required background on subsystem codes.

3 Background

Consider a system of n physical qubits with Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$, and the Pauli group $\mathcal{P}_n$, generated by all n-fold tensor products of the 2×2 identity operator I and the Pauli matrices

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2}$$

The elements of $\mathcal{P}_n$ can be expressed as $\alpha P_1 \otimes P_2 \otimes \cdots \otimes P_n$, with $P_j \in \{I, X, Y, Z\}$ and $\alpha \in \{\pm 1, \pm i\}$. The 'weight' $|P|$ of a Pauli operator P is the number of non-identity one-qubit Pauli matrices comprising P.

An $[[n, k, d]]$ subspace code encodes k logical qubits into $n > k$ physical qubits [48]. The parameter d is the code distance: the code can correct $\lfloor \frac{d-1}{2} \rfloor$ errors and detect $d-1$ errors.

3.1 Stabilizer Subspace Codes

A stabilizer subspace code is defined by $n-k$ independent, mutually commuting stabilizers $S_j \in \mathcal{P}_n$, $j \in \{1, \cdots, n-k\}$ [18, 19]. The stabilizers generate an order 2^{n-k} Abelian stabilizer group $\mathcal{S} = \langle S_1, \cdots, S_{n-k} \rangle$, where throughout we use angular brackets to denote a generating set of a group. An important subspace is the codespace $\mathcal{C} = \{|\psi\rangle \in \mathcal{H} : S|\psi\rangle = |\psi\rangle \ \ \forall S \in \mathcal{S}\}$, consisting of all states that remain invariant under the action of all elements of the stabilizer group. Consequently, the Hilbert space can be expressed as a direct sum of $\mathcal{C}$ and its orthogonal complement, $\mathcal{C}^\perp$, i.e., $\mathcal{H} = \mathcal{C} \oplus \mathcal{C}^\perp$.

The stabilizer group $\mathcal{S}$ excludes $-I$ and forms an Abelian subgroup of $\mathcal{P}_n$. The codespace is the simultaneous $+1$ eigenspace of $\mathcal{S}$. The normalizer (also the centralizer $C_{\mathcal{P}_n}(\mathcal{S})$ in this setting[1]) $N_{\mathcal{P}_n}(\mathcal{S})$ preserves the codespace $\mathcal{C}$ and encompasses all stabilizers and logical operators. The latter enact non-identity transformations between code states. Thus, we can decompose the normalizer as $N_{\mathcal{P}_n}(\mathcal{S}) = \langle S_1, \cdots, S_{n-k}, \bar{X}_1, \bar{Z}_1, \cdots, \bar{X}_k, \bar{Z}_k \rangle$, where $\bar{X}_j$ and $\bar{Z}_j$ are, respectively, logical X and Z operators on logical qubit j.

The code distance d is the minimum weight of a logical operator, i.e., $d = \min_{P \in N_{\mathcal{P}_n}(\mathcal{S}) \setminus \mathcal{S}} |P|$. Note that logical operators are equivalent under multiplication by stabilizer elements, so the minimization includes this operation.

3.2 Stabilizer Subsystem Codes

An $[[n, k, g, d]]$ stabilizer subsystem code [38] can be understood as an $[[n, k+g, d]]$ stabilizer subspace code where g of the logical qubits and the corresponding logical operators are not used. More precisely, in the subsystem case, we select a subset of k logical qubits from a total of $k+g$ logical qubits of a subspace code, while the remaining g qubits become gauge qubits. This re-

[1] The centralizer group of a subset $V \subset G$ in a group G is $C_G(V) = \{g \in G : [g, v_i] = 0 \ \ \forall v_i \in V\}$. The normalizer is $N_G(V) = \{g \in G : [g, V] = 0\}$.

sults in a division of the codespace $\mathcal{C}$ into two distinct subsystems $\mathcal{A}$ and $\mathcal{B}$, corresponding to the logical and gauge qubits, respectively. The system Hilbert space decomposes accordingly as $\mathcal{H} = \mathcal{A} \otimes \mathcal{B} \oplus \mathcal{C}^{\perp}$.

The logical operators pertaining to the logical qubits used to encode information act as the identity on $\mathcal{B}$ (i.e., on the gauge qubits) and form a non-Abelian group $\mathcal{L} = \langle \bar{X}_1, \bar{Z}_1, \cdots, \bar{X}_k, \bar{Z}_k \rangle$, where $\bar{X}_i$ and $\bar{Z}_i$ are the logical Pauli operators on the i'th logical qubit. These are known as *bare logical operators*; they preserve the code space $\mathcal{C}$ and act trivially on the gauge qubits. Likewise, the logical operators pertaining to the gauge qubits act trivially on $\mathcal{A}$ (i.e., on the logical qubits) and form a non-Abelian group $\mathcal{K} = \langle \tilde{X}_1, \tilde{Z}_1, \cdots, \tilde{X}_g, \tilde{Z}_g \rangle \equiv \langle \bar{X}_{k+1}, \bar{Z}_{k+1}, \cdots, \bar{X}_{k+g}, \bar{Z}_{k+g} \rangle$. From here on, we use the bar and tilde decorations to denote the logical and gauge operators, respectively.

The sets $\{\mathcal{S}, \mathcal{L}, \mathcal{K}\}$ mutually commute. Subsystem codes are further characterized by their *gauge group* $\mathcal{G} = \langle S_1, \cdots, S_{n-k}, \tilde{X}_1, \tilde{Z}_1, \cdots, \tilde{X}_g, \tilde{Z}_g \rangle$, which is non-Abelian unless $g = 0$. Thus, subsystem codes with an Abelian gauge group are equivalent to subspace stabilizer codes.

Following [40], given the gauge group, we may characterize the stabilizers as the elements of $\mathcal{P}_n$ that commute with every element in $\mathcal{G}$ and are also in $\mathcal{G}$, i.e.,

$$\mathcal{S} = \mathcal{G} \cap C_{\mathcal{P}_n}(\mathcal{G}). \tag{3}$$

We may likewise characterize the bare logical operators as nontrivial logical operators that are elements of $C_{\mathcal{P}_n}(\mathcal{G})$ but not in $\mathcal{G}$, i.e., $\mathcal{L} = C_{\mathcal{P}_n}(\mathcal{G}) \backslash \mathcal{G}$. Note that the centralizer of the stabilizer group is $C_{\mathcal{P}_n}(\mathcal{S}) = \langle \mathcal{G}, \bar{X}_1, \bar{Z}_1, \cdots, \bar{X}_k, \bar{Z}_k \rangle$. The elements of $C_{\mathcal{P}_n}(\mathcal{S}) \backslash \mathcal{G}$ are called *dressed logical operators*, as they may act non-trivially on the gauge qubits: dressed logical operators can be written as a product of a bare logical operator and a gauge operator. From here on, we use the bar and hat decorations to denote the bare logical and dressed logical operators, respectively. The code distance parameter d is the minimum weight of all nontrivial dressed logical operators.

3.3 Bravyi's A matrix

In this section, we briefly review a generalization of the two-dimensional (2D) Bacon-Shor code [49, 50] by closely following the method introduced in Ref. [40]. In Bravyi's approach, a square binary A matrix is mapped to a stabilizer subsystem code, where 1-entries in the matrix represent physical qubits. Let $|A|$ denote the Hamming weight (number of 1's) in A, and let d_{row} and d_{col} denote the minimum weight of the non-zero vectors in the row space and column space of A, respectively, where linear subspaces are defined over the binary field $\mathbb{F}_2$.

Theorem 1 (Rephrased Theorem 2 in [40])**.** *Let A be an arbitrary $m \times m$ binary matrix and $\mathcal{A}$ be the subsystem code associated with A. Then $\mathcal{A}$ encodes $k = \text{rank}(A)$ qubits into $n = |A|$ qubits with minimum distance $d = \min(d_{\text{row}}, d_{\text{col}})$.*

Using Theorem 1, we can relate the parameters $[[n, k, d]]$ of an error correcting code as in Table 1.

We also need to identify logical operators and stabilizers for the code, in order to perform calculations and detect errors.

Definition 1 (Gauge group)**.** *A subsystem code $\mathcal{A}$ associated with an A matrix has a gauge group $\mathcal{G}$ generated by the following operators:*

1. *Every pair of qubits j, j' located in the same row of A contributes a generator $X_j X_{j'}$.*

2. *Every pair of qubits i, i' located in the same column of A contributes a generator $Z_i Z_{i'}$.*

In anticipation of the construction of X-type and Z-type logical operators below, we define:

Definition 2. *Row and column operators:*

1. *$R_i = \prod_{j:A_{i,j}=1} Z_{i,j}$ is a* row operator *acting via Z on every qubit in the i'th row.*

2. *$C_j = \prod_{i:A_{i,j}=1} X_{i,j}$ is a* column operator *acting via X on every qubit in the j'th column.*

Having identified the gauge group $\mathcal{G}$, the stabilizer group $\mathcal{S}$ is determined through Eq. (3), where the centralizer $C_{\mathcal{P}_n}(\mathcal{G})$ is generated by the row and column operators, as shown in [40, Prop. 1]:

$$C_{\mathcal{P}_n}(\mathcal{G}) = \langle R_1, \cdots, R_m, C_1, \cdots, C_m \rangle. \tag{4}$$

The A matrix represents the commutation relations of the row and column operators. We have

Code parameters	Properties of A	Parameter meaning
n	$\|A\|$	Number of physical qubits
k	$\mathrm{rank}(A)$	Number of logical qubits
d	$\min(d_{\mathrm{row}}, d_{\mathrm{col}})$	Distance of the code

Table 1: Properties of the A matrix and their relation to the parameters of the corresponding error correcting code $[[n, k, d]]$.

$A_{ij} = 1$ if R_i and C_j overlap on one qubit (anticommute) and $A_{ij} = 0$ if they do not overlap (commute), namely

$$R_i C_j = (-1)^{A_{ij}} C_j R_i. \tag{5}$$

With row and column operators, we can represent the X-type and Z-type operators by bitstrings of length m as follows:

Definition 3. *X-type operators and Z-type operators:*

$$P^X(\boldsymbol{x}) = \prod_{j=0}^{m} C_j^{x_j}, \quad P^Z(\boldsymbol{z}) = \prod_{j=0}^{m} R_j^{z_j}, \tag{6}$$

where $\boldsymbol{x}$ and $\boldsymbol{z}$ are bitstrings of length m.

Thus, a 1-entry in the j'th position of the bitstring means the j'th column or row is involved in the corresponding X-type or Z-type operator.

Using Eq. (5), we can obtain the commutation relations between X-type operators and Z-type operators:

$$P^X(\boldsymbol{x}) P^Z(\boldsymbol{z}) = (-1)^{\boldsymbol{z}^T A \boldsymbol{x}} P^Z(\boldsymbol{z}) P^X(\boldsymbol{x}). \tag{7}$$

Any stabilizer or logical operator can be expressed using the form in Definition 3. Specifically, using Eq. (3) we can find stabilizers by calculating the kernels of A and A^T, i.e.:

Proposition 1 ([40]). *$P^X(\boldsymbol{x})$ is an X-type stabilizer iff $\boldsymbol{x} \in \mathrm{Ker}(A)$ and $P^Z(\boldsymbol{z})$ is a Z-type stabilizer iff $\boldsymbol{z} \in \mathrm{Ker}(A^T)$. Consequently, the number of X- and Z-type stabilizers is the nullity (dimension of the kernel) of A and A^T, respectively.*

Using standard Gram-Schmidt orthogonalization one can choose k pairs $\{P_a^X, P_a^Z\}_{a=1}^k$ as bare logical operators subject to the following conditions:

1. $P_a^X P_b^Z = (-1)^{\delta_{ab}} P_b^Z P_a^X$
2. $C_{\mathcal{P}_n}(\mathcal{G}) = \langle \mathcal{S}, P_1^X, \cdots, P_k^X, P_1^Z, \cdots, P_k^Z \rangle$

Proofs can be found in Ref. [40].

This construction only works for the bare logical operators. Going beyond [40], we next show how to modify it for the dressed logical operators.

Definition 4. *Reversed row and column operators:*

1. *$\bar{R}_i = \Pi_{j:A_{i,j}=1} X_{i,j}$ is a* row operator *acting via X on every qubit in the i'th row.*
2. *$\bar{C}_j = \Pi_{i:A_{i,j}=1} Z_{i,j}$ is a* column operator *acting via Z on every qubit in the j'th column.*

These are simply the row and column operators with the roles of X and Z exchanged compared to Definition 2. These operators are gauge operators which can be generated by the gauge group given in Definition 1. Similar to Eq. (5), it follows that

$$\bar{R}_i \bar{C}_j = (-1)^{A_{ij}} \bar{C}_j \bar{R}_i. \tag{8}$$

To establish a commutation relation for dressed logical operators, we make one assumption: Each row and column of the A matrix contains an even number of qubits, which holds for the A matrix in our construction (see Definition 6). Under this assumption, we have $[\bar{R}_i, R_j] = [\bar{C}_i, C_j] = 0$, as their overlaps occur on an even number of qubits. Also, $[\bar{R}_i, C_j] = [\bar{C}_i, R_j] = 0$ because they are the same type of operators.

In analogy to Definition 3, let us define a gauge-multiplied version of these operators.

Definition 5. *X-type operators and Z-type operators with gauge multiplication:*

$$Q^X(\boldsymbol{x}, \bar{\boldsymbol{x}}) = \prod_{j=0}^{m} C_j^{x_j} \bar{R}_j^{\bar{x}_j}, Q^Z(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \prod_{j=0}^{m} R_j^{z_j} \bar{C}_j^{\bar{z}_j}, \tag{9}$$

where $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{z}}$ are bitstrings of length m.

We can obtain the commutation relations between X-type and Z-type operators with gauge multiplication.

Proposition 2. *X-type and Z-type operators with gauge multiplication obey the following commutation relations:*

$$Q^X(\boldsymbol{x},\bar{\boldsymbol{x}})Q^Z(\boldsymbol{z},\bar{\boldsymbol{z}}) = (-1)^{\boldsymbol{z}^T A\boldsymbol{x}+\bar{\boldsymbol{z}}^T A^T\bar{\boldsymbol{x}}} Q^Z(\boldsymbol{z},\bar{\boldsymbol{z}})Q^X(\boldsymbol{x},\bar{\boldsymbol{x}}). \tag{10}$$

Proof. Note that using Eq. (5) we have $C_i^{x_i}R_j^{z_j} = (-1)^{z_j A_{ji} x_i} R_j^{z_j} C_i^{x_i}$, and using Eq. (8) we have $\bar{R}_i^{\bar{x}_i}\bar{C}_j^{\bar{z}_j} = (-1)^{\bar{x}_i A_{ij} \bar{z}_j}\bar{C}_j^{\bar{z}_j}\bar{R}_i^{\bar{x}_i}$. Therefore:

$$Q^X(\boldsymbol{x},\bar{\boldsymbol{x}})Q^Z(\boldsymbol{z},\bar{\boldsymbol{z}}) = \prod_{i=0}^{m} C_i^{x_i}\bar{R}_i^{\bar{x}_i}\prod_{j=0}^{m} R_j^{z_j}\bar{C}_j^{\bar{z}_j} \tag{11a}$$

$$= \prod_{i,j=0}^{m} C_i^{x_i} R_j^{z_j}\bar{R}_i^{\bar{x}_i}\bar{C}_j^{\bar{z}_j} \tag{11b}$$

$$= \prod_{i,j=0}^{m} (-1)^{z_j A_{ji} x_i} R_j^{z_j} C_i^{x_i} (-1)^{\bar{z}_j A_{ji}^T \bar{x}_i}\bar{C}_j^{\bar{z}_j}\bar{R}_i^{\bar{x}_i} \tag{11c}$$

$$= (-1)^{\boldsymbol{z}^T A\boldsymbol{x}+\bar{\boldsymbol{z}}^T A^T\bar{\boldsymbol{x}}} \prod_{j=0}^{m} R_j^{z_j}\bar{C}_j^{\bar{z}_j}\prod_{i=0}^{m} C_i^{x_i}\bar{R}_i^{\bar{x}_i} \tag{11d}$$

$$= (-1)^{\boldsymbol{z}^T A\boldsymbol{x}+\bar{\boldsymbol{z}}^T A^T\bar{\boldsymbol{x}}} Q^Z(\boldsymbol{z},\bar{\boldsymbol{z}})Q^X(\boldsymbol{x},\bar{\boldsymbol{x}}). \tag{11e}$$

□

Thus, one can choose k pairs $\{Q_a^X, Q_a^Z\}_{a=1}^k$ as dressed logical operators subject to the following conditions:

1. $Q_a^X Q_b^Z = (-1)^{\delta_{ab}} Q_b^Z Q_a^X$
2. $C_{\mathcal{P}_n}(\mathcal{S}) = \langle \mathcal{G}, Q_1^X, \cdots, Q_k^X, Q_1^Z, \cdots, Q_k^Z\rangle$

Moreover, row and column permutations of the A matrix leave the subsystem code $\mathcal{A}$ invariant. Specifically, we show in Appendix A that while the gauge generators are modified under permutations, the gauge group and the stabilizers remain invariant.

4 Trapezoid subsystem codes

In this section, we introduce a family of $[[4k+2l, 2k, g, 2]]$ and $[[4k+2l-2, 2k-1, g, 2]]$ codes for even and odd numbers of logical qubits, respectively. We call these 'trapezoid' codes due to the structure of their A matrices, which are defined as follows:

Definition 6 (A matrix for trapezoid codes)**.** *The $m \times m$ A matrix of the family of trapezoid codes is parameterized by the integers m and l, where $l \leq \lceil \frac{m-1}{2} \rceil$. Its entries are zero unless specified otherwise below. Its matrix elements are given by:*

- *first column:* $A_{i,1} = 1$ *for* $1 \leq i \leq 2l$,
- *last row:* $A_{m,i} = 1$ *for* $m - 2l + 1 \leq i \leq m$,
- *superdiagonal:* $A_{i,i+1} = 1$ *for* $1 \leq i \leq m-1$,
- *lower diagonal:* $A_{i,i-2l+1} = 1$ *for* $2l+1 \leq i \leq m-1$,

where $A_{i,j}$ is the element in the i'th row and j'th column, $i, j \in \{1, \cdots, m\}$.

This results in the rotated isosceles trapezoidal structure illustrated in Eq. (12) (the trapezoidal shape outlined by 1-entries), in which we have indicated the row and column indices where the top of the trapezoid forms. The bottom side of the trapezoid aligns with the superdiagonal of the A matrix and its legs are on the left and the bottom side of the A matrix with length $2l$.

$$A = \begin{array}{c|ccccccccccc}
 & 1 & 1 & & & & & & & & & \\
 & 1 & & 1 & & & & & & & & \\
 & \vdots & & & \ddots & & & & & & & \\
2l & 1 & & & & 1 & & & & & & \\
2l{+}1 & & 1 & & & & 1 & & & & & \\
 & & & \ddots & & & & & \ddots & & & \\
 & & & & 1 & & & & & & 1 & \\
m{-}1 & & & & & 1 & & & & & & 1 \\
m & & & & & & 1 & \cdots & & 1 & 1 & \\
\hline
 & & & & & m{-}2l{+}1 & & & & & m &
\end{array} \tag{12}$$

To identify parameters of the trapezoid codes, we use Table 1 together with the following observations:

- The last row is the sum of all the previous rows and all other rows are linearly independent, thus we have $\text{rank}(A) = m - 1$.
- $|A| = 2 * (m-1) + m - (m - 2l + 1) + 1 = 2m + 2l - 2$.
- Let $d_i^{(r)}$ and $d_i^{(c)}$ denote the Hamming weights of the i'th row and column, respectively. Definition 6 implies that $d_i^c = d_i^r = 2$

for every i except $d_1^c = d_m^r = 2l$, hence, $\min(d_{\text{row}}, d_{\text{col}}) = 2$.

Thus, the $m \times m$ A matrix corresponds to codes with parameters $[[4k+2l, 2k, g, 2]]$ for $m = 2k+1$ and $[[4k+2l-2, 2k-1, g, 2]]$ for $m = 2k$.

We refer to the $m = 2k+1$ and $m = 2k$ cases as the odd-m and even-m codes, respectively. This is not to be confused with the even and odd numbers of logical qubits, which is $m-1$. We show below that

$$g = 2k + 2l - 2 \quad \text{for odd } m \tag{13a}$$
$$g = 2k + 2l - 3 \quad \text{for even } m. \tag{13b}$$

We note that our construction generalizes the $[[6k, 2k, 2]]$ code introduced in Ref. [39], which is the special case $l = k$ of the odd-m trapezoid code, with $g = 4k - 2$.

4.1 Properties of trapezoid codes

4.1.1 Rate

The odd-m and even-m codes have different code rates, i.e.,

$$r_{\text{odd}} = \frac{k}{2k+l}, \quad r_{\text{even}} = \frac{2k-1}{4k+2l-2}. \tag{14}$$

The code rate is maximized for $l = 1$ in both cases. Since the $[[6k, 2k, 2]]$ code corresponds to $l = k$, its rate is always surpassed by any trapezoid code with $l < k$.

For the same l, $r_{\text{odd}} > r_{\text{even}}$. For the same number of physical qubits, also, $r_{\text{odd}} > r_{\text{even}}$. Thus, the odd-$m$ codes are usually desirable since they have a higher code rate. In the following discussion, we focus our attention on the odd-m codes unless explicitly noted otherwise. The proofs for the even-m cases follow the same reasoning.

4.1.2 Stabilizers

From the rank-nullity theorem, $\text{rank}(A) + \text{nullity}(A) = m$. Hence, using Proposition 1 and $\text{rank}(A) = m - 1 = 2k$, the number of X and Z stabilizers is $\text{nullity}(A) = m - (m-1) = 1$. This shows that each of the codes in the trapezoid family has $|\mathcal{S}| = 2$ stabilizers (one S_X and one S_Z).

Lemma 1. *$S_X = \prod_{i=1}^m C_i$ and $S_Z = \prod_{i=1}^m R_i$ are valid stabilizers of the A matrix of the trapezoid codes family.*

Proof. Using Proposition 1, the stabilizers of X-type (Z-type) can be identified with the right (left) null vectors of A. Since S_X (S_Z) is a product of all columns (rows), it can be parameterized using a bitstring $\boldsymbol{x} = 1^m$ ($\boldsymbol{z} = 1^m$), as in Eq. (6). Since every row of A has an even number of 1-entries, $A\boldsymbol{x} = \mathbf{0}$. Since every column of A has an even number of entries of 1, $\boldsymbol{z}^T A = \mathbf{0}^T$. □

Recall that the 1-entries in an A matrix represent all the physical qubits. It follows that the two stabilizers are the all-X and all-Z Pauli operators. This holds for both odd-m and even-m codes. Lemma 1 shows that the stabilizers are the same as those of the family of $[[n, n-2, 3]]$ subspace stabilizer codes, whose two stabilizers are also the all-X and all-Z Pauli operators. Our family of subsystem trapezoid codes generalizes this family of subspace codes by discarding some gauge qubits and imposing a 2D geometry.

Note that the weight of the stabilizers grows linearly as the number of physical qubits increases. Therefore, using the stabilizers as penalty terms [17] would require at least n-body interactions in this case, which is impractical for $n > 2$. One of the major advantages of using subsystem codes is that we can replace these high weight stabilizers by 2-local gauge operators as penalty terms.

4.1.3 Gauge group

For a subspace code with parameters $[[n, k'+g, d]]$ we have $|\mathcal{S}| = n-(k'+g)$ [18]. The same equality holds for the corresponding subsystem code with parameters $[[n, k', g, d]]$ (where g is the number of unused logical qubits or gauge qubits). Thus, for the odd-m trapezoid family, for which $n = 4k+2l$ and $k' = 2k$, we have $g = 2k + 2l - 2$. Likewise, for even-m codes we have $n = 4k + 2l - 2$ and $k' = 2k - 1$, so $g = 2k + 2l - 3$. This confirms Eq. (13).

The gauge group $\mathcal{G}$ is generated by the logical operators of the gauge qubits (i.e., $2g$ generators) and the two stabilizers. Hence, $|\mathcal{G}| = 2g + 2 = 4(k+l)-2$ (odd-m) or $|\mathcal{G}| = 4(k+l)-4$ (even-m).

4.2 Example: the $k = 3$ family

For the odd-m codes, we have three $m \times m$ A matrices where $m = 2k + 1 = 7$ for $1 \le l \le 3$ as shown in Eq. (15).

Notation	Meaning	Example
A matrix	$m \times m$ Bravyi's A matrix in [40]	Eq. (12)
$\bar{X}_i, \bar{Z}_i$	Bare logical operators X, Z for logical qubit i	
$\hat{X}_i, \hat{Z}_i$	Dressed logical operators X, Z for logical qubit i	
$\boldsymbol{X}, \boldsymbol{Z}$	$(m-1) \times m$ operator matrix parameterizing bare logical operators	Eq. (19)
$\bar{\boldsymbol{X}}, \bar{\boldsymbol{Z}}$	$(m-1) \times m$ gauge matrix parameterizing gauges to reduce bare to dressed logical operators	Eq. (20)

Table 2: Summary of notations used in this work and their examples.

$$\underset{l=1:[[14,6,6,2]]}{\begin{bmatrix} 1 & 1 & & & & & \\ 1 & & 1 & & & & \\ & 1 & & 1 & & & \\ & & 1 & & 1 & & \\ & & & 1 & & 1 & \\ & & & & 1 & & 1 \\ & & & & & 1 & 1 \end{bmatrix}} \quad \underset{l=2:[[16,6,8,2]]}{\begin{bmatrix} 1 & 1 & & & & & \\ 1 & & 1 & & & & \\ 1 & & & 1 & & & \\ 1 & & & & 1 & & \\ & 1 & & & & 1 & \\ & & 1 & & & & 1 \\ & & & 1 & 1 & 1 & 1 \end{bmatrix}} \quad \underset{l=3:[[18,6,10,2]]}{\begin{bmatrix} 1 & 1 & & & & & \\ 1 & & 1 & & & & \\ 1 & & & 1 & & & \\ 1 & & & & 1 & & \\ 1 & & & & & 1 & \\ 1 & & & & & & 1 \\ & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}} \tag{15}$$

For the even-m codes, we have another three $m \times m$ A matrices where $m = 2k = 6$ for $1 \leq l \leq 3$ as shown in Eq. (16).

$$\underset{l=1:[[12,5,5,2]]}{\begin{bmatrix} 1 & 1 & & & & \\ 1 & & 1 & & & \\ & 1 & & 1 & & \\ & & 1 & & 1 & \\ & & & 1 & & 1 \\ & & & & 1 & 1 \end{bmatrix}} \quad \underset{l=2:[[14,5,7,2]]}{\begin{bmatrix} 1 & 1 & & & & \\ 1 & & 1 & & & \\ 1 & & & 1 & & \\ 1 & & & & 1 & \\ & 1 & & & & 1 \\ & & 1 & 1 & 1 & 1 \end{bmatrix}}$$

$$\underset{l=3:[[16,5,9,2]]}{\begin{bmatrix} 1 & 1 & & & & \\ 1 & & 1 & & & \\ 1 & & & 1 & & \\ 1 & & & & 1 & \\ 1 & & & & & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}} \tag{16}$$

4.3 Construction of logical operators

The choice of logical operators is not unique. Here, we give a set of logical operators as two $(m-1) \times m$ matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$, where each row of the matrices represents the bitstring corresponding to a logical operator. Let

$$J_i = \lfloor \frac{m-i-1}{2l} \rfloor \,, \quad \bar{J}_i = \lfloor \frac{i-1}{2l} \rfloor. \tag{17}$$

We define:

$$\boldsymbol{X}_{i,i+1+2lj} = 1 \quad \text{for } 1 \leq i \leq m-1 \text{ and } 0 \leq j \leq J_i, \tag{18a}$$

$$\boldsymbol{Z}_{i,i-2lj} = 1 \quad \text{for } 1 \leq i \leq m-1 \text{ and } 0 \leq j \leq \bar{J}_i. \tag{18b}$$

For the three $m = 7$ codes in Eq. (15), the logical operators constructed from Eq. (18) are given in the *operator matrices* below:

$$\boldsymbol{X} = \begin{bmatrix} 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}, \boldsymbol{Z} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \end{bmatrix}$$
$$l = 1 \tag{19a}$$

$$\boldsymbol{X} = \begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}, \boldsymbol{Z} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$$
$$l = 2 \tag{19b}$$

$$\boldsymbol{X} = \begin{bmatrix} 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}, \boldsymbol{Z} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \end{bmatrix}$$
$$l = 3 \tag{19c}$$

and gauge operators to reduce the weights of these logical operators can be represented in another two $(m-1) \times m$ *gauge matrices* below:

$$\bar{\boldsymbol{X}} = \begin{bmatrix} 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \bar{\boldsymbol{Z}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \end{bmatrix}$$
$$l = 1 \tag{20a}$$

$$\bar{\boldsymbol{X}} = \begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \bar{\boldsymbol{Z}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$l = 2 \tag{20b}$$

The $\bar{\boldsymbol{X}}$ and $\bar{\boldsymbol{Z}}$ for $l = 3$ code are two zero matrices since all bare logical operators are 2-local by construction.

We can construct a set of dressed, 2-local log-

ical operators using Definition 5, where the bitstrings $\boldsymbol{x}, \bar{\boldsymbol{x}}$ and $\boldsymbol{z}, \bar{\boldsymbol{z}}$ are taken from each row of $\boldsymbol{X}, \bar{\boldsymbol{X}}$ and $\boldsymbol{Z}, \bar{\boldsymbol{Z}}$, respectively. An example is illustrated in Fig. 1.

Definition 7 (Dressed, 2-local logical operators for trapezoid codes)**.** *Dressed logical operators are constructed according to Definition 5, i.e.,* $\hat{X}_i = Q^X(\boldsymbol{X}_i, \bar{\boldsymbol{X}}_i)$ *and* $\hat{Z}_i = Q^Z(\boldsymbol{Z}_i, \bar{\boldsymbol{Z}}_i)$ *where* $\boldsymbol{X}_i, \bar{\boldsymbol{X}}_i, \boldsymbol{Z}_i$ *and* $\bar{\boldsymbol{Z}}_i$ *are the* i*'th row of* $\boldsymbol{X}, \bar{\boldsymbol{X}}, \boldsymbol{Z}$ *and* $\bar{\boldsymbol{Z}}$*, respectively. By construction,* $Q^X(\boldsymbol{X}_i, \bar{\boldsymbol{X}}_i)$ *(*$Q^Z(\boldsymbol{Z}_i, \bar{\boldsymbol{Z}}_i)$*) is* $P^X(\boldsymbol{X}_i)$ *(*$P^Z(\boldsymbol{Z}_i)$*) multiplied by appropriate gauge operators that reduce it to 2-local.*

Table 2 summarizes frequently occurring notations we use in this work. Note that the reduction to a 2-local operator is possible as stated since, by Definition 1, every pair of qubits c, c' located in the same row of A contributes a 2-local gauge generator $X_c X_{c'}$. For $l = k$, the logical operator is 2-local by construction, hence, no further reduction is needed. Without loss of generality, the indexing in the proofs below will be for odd-m codes.

Lemma 2. *The set of logical operators given in Definition 7 is valid, i.e., for all* i, j*, (1)* $\hat{X}_i$ *and* $\hat{Z}_i$ *commute with the two stabilizers, and (2)* $\{\hat{X}_i, \hat{Z}_i\} = 0$ *and* $[\hat{X}_i, \hat{Z}_j] = 0$ *when* $i \neq j$*.*

Proof. To prove (1), recall from Lemma 1 that S_X is a product of all $4k + 2l$ physical X operators. Every dressed $\hat{Z}_i$ has two physical Z operators. Let $\hat{Z}_i = Z_{i_1} Z_{i_2}$. We have $[S_X, \hat{Z}_i] = [X_{i_1} X_{i_2}, Z_{i_1} Z_{i_2}] = 0$. Using a similar argument for S_Z, it follows that $[S_X, \hat{Z}_i] = [S_Z, \hat{X}_i] = 0$.

To prove (2), consider an operator $P^X(\boldsymbol{X}_i)$ and the corresponding A matrix. Each pair of columns of $P^X(\boldsymbol{X}_i)$, $(a+1)$'th column and $(b+1)$'s column with $|a - b| = 2l$ has two X operators in the same row. Multiplying these rows by X-type gauge operators and canceling out X operators on the same qubits, we are left with one X operator in the $(i+1)$'th column, $X_{i,i+1}$, and another X operator in the q_i'th column, X_{m,q_i}, where $q_i = i + 1 + 2lJ_i$. Thus, $\hat{X}_i = X_{i,i+1} X_{m,q_i}$.

A similar argument holds for $\hat{Z}_i$. Multiplying Z-type gauge (column) operators and canceling out Z operators on the same qubits, we are left with one Z operator in the i'th row, $Z_{i,i+1}$, and another Z operator in the $\bar{q}_i$'th row, $Z_{\bar{q}_i,1}$, where $\bar{q}_i = i - 2l\bar{J}_i$. Thus, $\hat{Z}_i = Z_{i,i+1} Z_{\bar{q}_i,1}$.

It is straightforward to see that $\{\hat{X}_i, \hat{Z}_i\} = 0$ since they overlap on exactly one qubit, $(i, i+1)$, and that $[\hat{X}_i, \hat{Z}_j] = 0$ when $i \neq j$ since the two operators do not overlap. □

We let $A \overset{\text{g}}{=} B$ denote $A = B$ up to multiplication by gauge operators, i.e., A and B have the same operation on the logical qubits.

Lemma 3. *The set of dressed logical operators given above is optimal in terms of physical locality, i.e., for all* i *and* j*, the operators* $\hat{X}_i, \hat{Z}_i, \hat{X}_i \hat{X}_j$*, and* $\hat{Z}_i \hat{Z}_j$ *are 2-local.*

Informally, the reason is that any operation on the gauge qubits can be ignored or discarded. Thus, even though the representation of a logical operator such as $\hat{X}_i \hat{X}_j$ is a 4-local physical operator, two of the physical operators act on the gauge qubits and can hence be ignored. The formal proof follows.

Proof. In Lemma 2, we showed that for all $i, \hat{X}_i$ and $\hat{Z}_i$ are 2-local. Now, we show that for all i and $j, \hat{X}_i \hat{X}_j$ and $\hat{Z}_i \hat{Z}_j$ are also 2-local. First, using Eq. (17) note that $q_i = i + 1 + 2lJ_i = m - 1$ when i is odd and $q_i = m$ when i is even. When i and j have the same parity, $\hat{X}_i \hat{X}_j = X_{i,i+1} X_{j,j+1}$ since $\hat{X}_i = X_{i,i+1} X_{m,q_i}$ and the X_{m,q_i}'s from the two terms cancel. When i and j have a different parity, we have $\hat{X}_i \hat{X}_j = X_{i,i+1} X_{j,j+1} X_{m,q_i} X_{m,q_j} \overset{\text{g}}{=} X_{i,i+1} X_{j,j+1}$ since $X_{m,q_i} X_{m,q_j}$ constitutes a gauge operator (as in Definition 1). A similar argument holds for $\hat{Z}_i \hat{Z}_j$. Hence, $\hat{Z}_i \hat{Z}_j \overset{\text{g}}{=} Z_{i,i+1} Z_{j,j+1}$. □

The logical operators constructed above are dressed logical operators. Encoding the problem Hamiltonian using dressed logical operators is sufficient to protect the computational space in the large penalty limit [39]. We show in Lemma 4 that bare logical operators cannot all be 2-local.

Lemma 4. *Bare logical operators* $\bar{X}$ *and* $\bar{Z}$ *cannot all be 2-local for a code in the trapezoid family with* $l < k$*.*

Proof. First, we show that an operator is 2-local if the corresponding bitstring has Hamming weight 1, which forms rows in the operator matrices of size $(m-1) \times m$. Next, we show that the operator matrices of rank $m-1$ consisting of only two local operators must be an $(m-1) \times (m-1)$

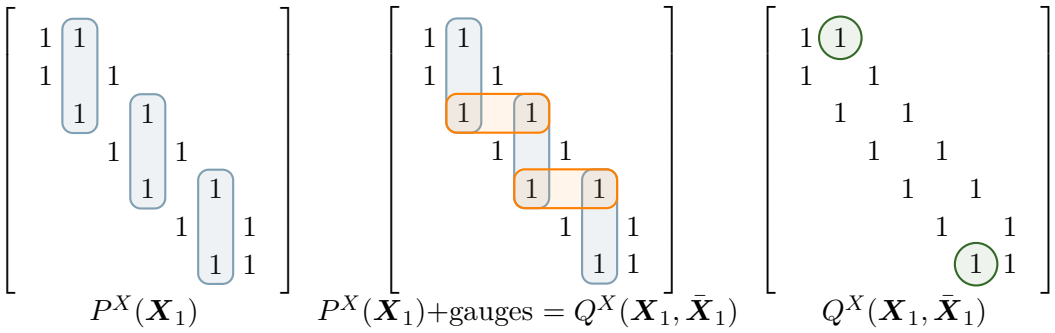


Figure 1: Construction of the dressed logical operator $\hat{X}_1$ for the $[[14, 6, 6, 2]]$ code. $\boldsymbol{X}_1$ from Eq. (19a) has 1-entries in the 2, 4, and 6 positions, corresponding to those columns of X operators in the A matrix (blue boxes). This is $P^X(\boldsymbol{X}_1)$, a six-local operator. Next, use $\bar{\boldsymbol{X}}_1$ from Eq. (20a), which has 1-entries in the 3 and 5 positions, corresponding to those rows of X gauges in the A matrix (orange boxes). The blue and orange operators make up the $Q^X(\boldsymbol{X}_1, \bar{\boldsymbol{X}}_1)$ operator. After multiplying all the X operators, double operators on the same qubit cancel. Hence, it simplifies to 2-local dressed logical operator $\hat{X}_1$ (green circles).

identity padded by an all-zero column up to permutation, as in Eq. (19c). Finally, we show that the operators found in these matrices do not satisfy the commutation condition for logical operators of codes with $l < k$.

First, a 1-entry in the bitstring in the operator matrix means that a column or a row of the A matrix contributes to the corresponding logical operator. From our construction, each row or column of the A matrix has at least two physical qubits. Since no gauge cancellation is allowed for bare logical operators, a bitstring in the operator matrix corresponding to a 2-local operator can only have Hamming weight 1.

Next, as shown in the previous paragraph, there is one 1-entry in each row of the $(m-1)\times m$ operator matrix. Since we need $m-1$ logical operators, the rank of the operator matrix must be $m-1$, which implies that there is one 1-entry in each column of the operator matrix. Thus, the only way to construct the operator matrix is to have an $(m-1)\times(m-1)$ identity padded by a $(m-1)\times 1$ column of zeros up to permutations. This is because each of the $m-1$ rows and columns must be linearly independent and have Hamming weight 1.

Finally, if we were to use the operator matrix found in the previous paragraph to construct the logical operators for $l < k$ codes, we could always find two $\bar{X}$ operators that anticommute with the same $\bar{Z}$ operator, which is a contradiction. For example, $\bar{X}_1$ and $\bar{X}_{2l+1}$ both anticommute with $\bar{Z}_{2l+1}$, as seen from using logical operators in Eq. (19c) for $l = 1$ and $l = 2$ codes in Eq. (15).

With these conditions, one or more bare logical operators cannot be 2-local for any $l < k$ code that satisfies the commutation relation. □

In order to find the dressed logical operators, a brute-force approach is to search the entire space of the logical operators multiplied by gauge operators. However, this incurs an exponential cost in m. We provide a method for reducing the search space dimension to polynomial in m in Appendix B.

4.4 Hamiltonian penalty

The goal is to protect a computation performed by a system Hamiltonian $H_S(t)$ in Eq. (1) against the system-bath interaction described by $I_S \otimes H_B + H_{SB}$ where $H_{SB} = \sum_\alpha \sigma_\alpha \otimes B_\alpha$ and σ_α is a 1-local system operator (i.e., the α index runs over the physical qubits and x, y, z in the Pauli basis). H_B is the pure-bath Hamiltonian.

The encoded Hamiltonian, $\bar{H}_S(t)$, is constructed by replacing every operator in $H_S(t)$ with the corresponding bare logical operators, i.e.,

$$\bar{H}_S(t) = \sum_i a_i \bar{X}_i + \sum_i b_i \bar{Z}_i + \sum_{(i,j)\in E_S^{XX}} c_{ij} \bar{X}_i \bar{X}_j + \sum_{(i,j)\in E_S^{ZZ}} d_{ij} \bar{Z}_i \bar{Z}_j, \tag{21}$$

where $\bar{X}_i$ and $\bar{Z}_i$ are bare logical operators. In this manner, $\bar{H}_S(t)$ is universal for AQC in the code subspace.

We refer to the reducible logical operators as those whose interactions can be reduced to 2-

local upon multiplication by one or more gauge operators. We multiply these operators in $\bar{H}_S(t)$ by appropriate gauge operators to reduce the many-body interactions down to 2-local and write the new Hamiltonian as $\hat{H}_S(t)$:

$$\hat{H}_S(t) = \sum_i a_i \bar{X}_i g_i^x + \sum_i b_i \bar{Z}_i g_i^z + \sum_{(i,j)\in E_S^{XX}} c_{ij} \bar{X}_i \bar{X}_j g_{ij}^x + \sum_{(i,j)\in E_S^{ZZ}} d_{ij} \bar{Z}_i \bar{Z}_j g_{ij}^z, \tag{22a}$$

$$= \sum_{(i,j)\in E^X} a'_{(i,j)} X_i X_j + \sum_{(i,j)\in E^Z} b'_{(i,j)} Z_i Z_j + \sum_{(i,j)\in E^{XX}} c'_{(i,j)} X_i X_j + \sum_{(i,j)\in E^{ZZ}} d'_{(i,j)} Z_i Z_j. \tag{22b}$$

In Eq. (22b), E^X, E^Z , E^{XX}, and E^{ZZ} represent the sets of edges in the induced graph G supporting the $\hat{X}$, $\hat{Z}$, $\hat{X}\hat{X}$, and $\hat{Z}\hat{Z}$ interactions, respectively. The operators X and Z are physical operators. This means that $\hat{H}_S(t)$ can be implemented using only 2-local interactions. The primed coefficients (a'_i, etc.) are related to the unprimed ones (a_i, etc.) via α_i in $\Pi_0 g_i \Pi_0 = \alpha_i \Pi_0$, where Π_0 is the projector to the ground subspace of H_P, as explained in detail in Appendix C.

A penalty term $\epsilon_P H_P$ is added to the encoded Hamiltonian to protect the computational space. The penalty Hamiltonian, H_P, is constructed using the generators $\mathcal{G}'$ of the gauge group $\mathcal{G}$, i.e.,

$$H_P = -\sum_{g_i\in\mathcal{G}'} g_i. \tag{23}$$

The ground subspace of the penalty Hamiltonian is in the codespace [51]. Thus, the penalty Hamiltonian penalizes transitions into or out of the codespace. In general, $[\hat{H}_S(t), H_P] \neq 0$ due to the gauge terms in Eq. (22), which may interfere with the computation in the codespace. However, Ref. [39] showed that by rescaling the coefficients, the coupling effect can be compensated. We use a slightly modified version; details can be found in Appendix C. The total Hamiltonian now is:

$$H(t) = [\hat{H}_S(t) + \epsilon_P H_P] \otimes I_B + I_S \otimes H_B + H_{SB}. \tag{24}$$

The 'penalty gap' is the energy difference between the ground state and the first excited state of H_P. The separation between the codespace and the lowest excited subspace depends on this gap, the energy penalty ϵ_P, and the geometry of the induced graph. We calculate the penalty gap for various code parameters for the induced graphs shown in Fig. 2, as well as for additional induced graphs with other values of k and l. The calculations and detailed proofs for the numerics in this section are provided in Appendix C.

Fig. 3 shows the results of numerical calculations of the penalty gap. We plot the penalty gap for codes with $l = 1, 2, \ldots, 7$ and $m = 2, 3, \ldots, 20$, where $m \times m$ is the size of the A matrix. The parameters m and k are related via $m = 2k + 1$ for odd-m codes and $m = 2k$ for even-m codes. Hence, m scales with the number of logical qubits k while the number of physical qubits scales with l.

Fig. 3 (left) (supported by a careful analysis of fitting function in Appendix D) shows that for a fixed l, the gap closes as a power-law for $l = 1, 2$ with respect to the number of logical qubits (recall that $k = \lfloor m/2 \rfloor$), and exponentially for $l \geq 3$. Fig. 3 (middle) shows the scaling as a function of the number of logical qubits for $l(k) = k - l'$ where $l' = 0, 1, \ldots, 7$; the gap closes exponentially for all $l(k)$. This gap closure behavior can be explained on the basis of the qubit connectivity of H_P in Fig. 2.

The connectivity for $l = 1$ is chain-like for every m (see Fig. 2). The penalty Hamiltonian then reduces to the (critical) one-dimensional 90° quantum compass model: $H_P = -\sum_{i=1}^{m/2} X_{2i-1}X_{2i} + Z_{2i}Z_{2i+1}$ (identifying qubit $m+1$ with qubit 1), which is dual to the one-dimensional transverse field Ising model [52], whose gap is well known to scale as $1/m$ [53]. Fitting the $l = 1$ results, we find that the gap scales as $1/m^\nu$, where $\nu = 1.032 \pm 0.004$ (see Table 10). Our numerical result is in agreement with the analytical results of Ref. [54] for the one-dimensional compass model at the critical point ($\alpha = 1$ in their notation), who showed that the gap between the two lowest energy states in the ground state sector scales as $1/m$. This power-law dependence is desirable because it means that the penalty gap ensures protection for all problems whose gap closes faster than $1/m$, as is typically the case in AQC [3].

For $l = 2$ we find two one-dimensional compass models connected via two links (see Fig. 2 for $k = 3$ and Fig. 11 for $k = 4$). Despite a slight

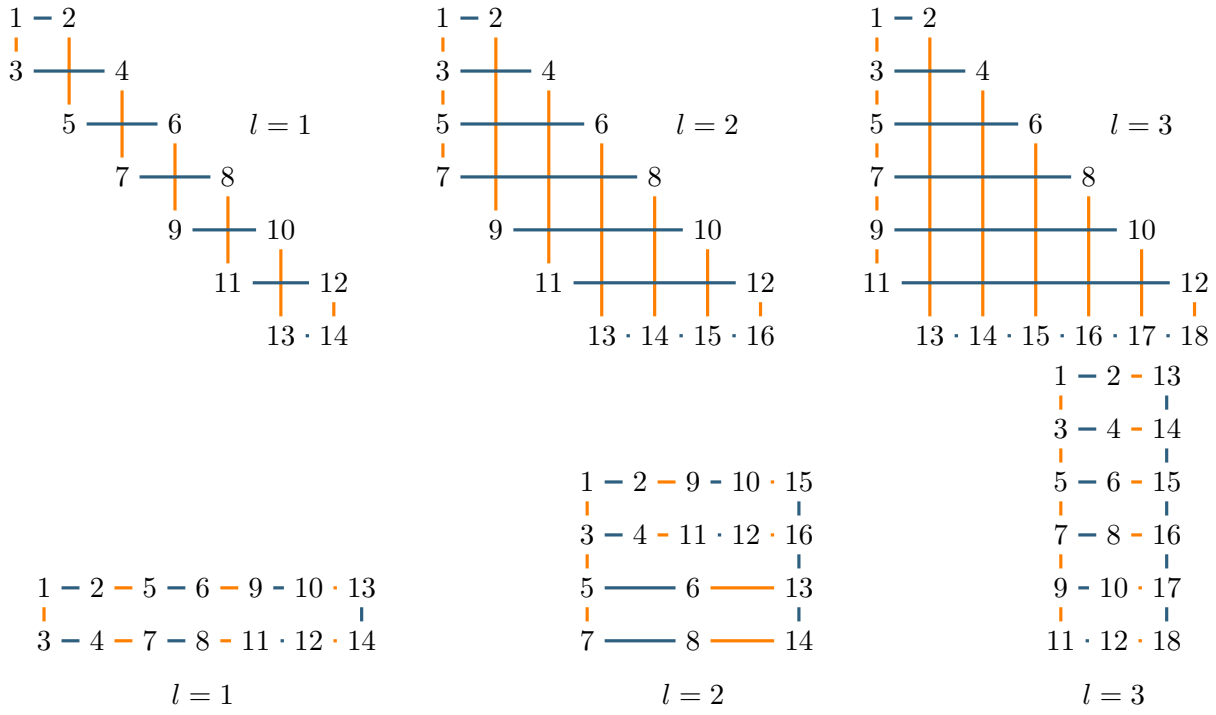


Figure 2: Gauge connectivity for $k = 3$ codes ($l = 1, 2, 3$). Blue (orange) lines are XX (ZZ) gauge operators. The top row arranges qubits in 2D in the form of the A matrix. The bottom row is a layout that minimizes the geometric distance between qubits. The $l = 1$ case is always a chain for any k.

curvature at small m visible in Fig. 3 (left), the gap scaling is best fit by a power law (see Table 11).

On the other hand, starting from $l = k$, we observe a ladder-like connectivity similar to the compass model [55, 56], whose gap decays exponentially with system size [57]. The difference between the $l = k$ connectivity and that of the standard 2D compass model is the missing middle leg of the ladder (see $l = k = 3$ in Fig. 2 and $l = k = 4$ in Fig. 11). It is reasonable to expect that despite this difference, H_P's spectrum is closer to the 2D than the 1D compass model, which leads to the penalty gap decaying exponentially rather than as a power law. This is confirmed numerically in Fig. 3 (middle), and we find that the gap scales as $e^{(-0.45\pm0.01)m}$ (for full details of the fitting parameters see Table 12). We note that this conclusion differs from [39], who claimed that the penalty gap closes polynomially.

For $2 < l < k$ we find a connectivity that is intermediate between the ladder-like structure of the $l = k$ case and the double one-dimensional compass model of the $l = 2$ case (see $l = 3$ in Fig. 11). This structure is closer to a 2D rather than 1D connectivity, suggesting that the gap closure for $2 < l < k$ is closer to the exponentially decaying gap of the 2D compass model than the power law decay of the 1D model. This is borne out by our numerics, which exhibit a more exponential-like decay the higher the value of l (see Table 11).

Fig. 3 (right) shows that the gap decreases exponentially as a function of l when m is fixed. This suggests that codes with smaller l (smaller number of physical qubits) provide an exponentially larger penalty gap. These codes with smaller l also have better code rates, which is desirable. However, as we discuss in the next section, there is a tradeoff between these advantages offered by small-l codes and the higher degree of physical qubit connectivity they require when we also incorporate the encoded system Hamiltonian $\hat{H}_S$.

5 Qubit Mapping

The induced graphs G that arise from the logical operators and gauges constructed in Section 4.3, have certain features in common. The central part of the induced graph forms a fully connected subgraph with $m - 1$ vertices (corre-

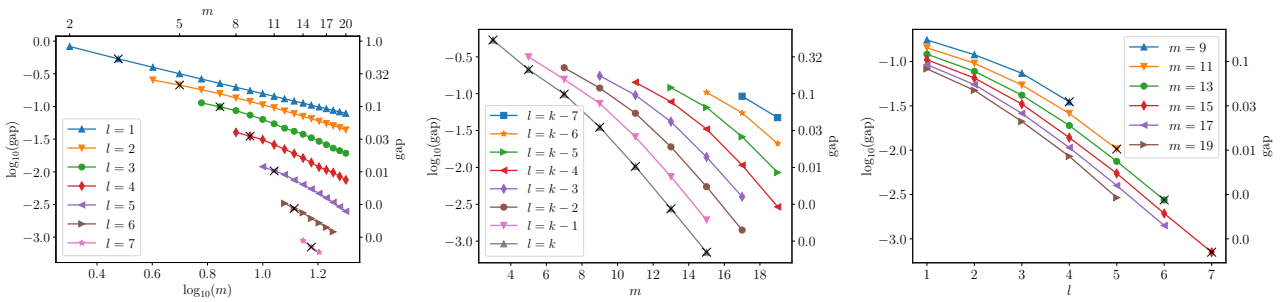


Figure 3: Penalty Hamiltonian gap as a function of the code matrix size m for different values of the code parameter l (left, middle) and as a function of l for different values of m (right). The number of logical qubits is $k = \lfloor m/2 \rfloor$ and the number of physical qubits is $4k+2l$ for odd m or $4k+2(l-1)$ for even m. The black crosses are the results for the odd-m codes with $l = k$ [39]. The gap is largest for $l = 1$ (which also maximizes the code rate) and decays more slowly than for $l > 1$. The decay with m at fixed l follows a power-law for $l = 1, 2$ and is exponential for $l > 2$ (left). The decay with m at fixed $l = k - l'$ is exponential in $m = 2k + 1$ (middle). The decay as a function of l at fixed m is exponential in all cases (right). See Appendix D for details. Missing data points for higher l and m are due to insufficient memory (we used 256GB GPUs).

sponds to the physical qubits on the superdiagonal in A matrix), representing the connectivity needed for two-qubit logical operators $\hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$, as shown by the yellow edges in Fig. 4. Additionally, qubits located in the first column and the last row of the A matrix are each coupled to $\lfloor \frac{m-1}{2l} \rfloor$ or $\lceil \frac{m-1}{2l} \rceil$ vertices in the fully connected subgraph in an alternating manner, accounting for the single-qubit logical operators $\hat{X}$ and $\hat{Z}$, as illustrated by the blue edges in Fig. 4. Finally, the qubits in the first column and the last row form two cyclic components, representing the connectivity required for gauge generators, corresponding to part of the green edges in Fig. 4.

The cyclic couplings among the qubits in the first column and the last row exist since we select nearest-neighbor gauge generators. The structure of these two components depends on the choice of gauge generators. For example, if the penalty Hamiltonian includes the entire gauge group, these two components will be fully connected subgraphs, each with $2l$ vertices.

For the case $l = 1$ (which maximizes the code rate), G consists of $m-1$ vertices (blue nodes in Fig. 4) with degrees of $m+1$ or $m+2$, $m-3$ vertices (green nodes in Fig. 4) with degrees of 2, and 4 vertices (light and dark brown nodes in Fig. 4) with degrees of $(m+1)/2$. At the other extreme, when $l = k$, G includes one fully connected subgraph and two cyclic graphs, each with $m-1$ vertices. In this case, the induced graph consists of $m-1$ vertices (blue nodes in Fig. 4) with degrees of $m+1$ and $2m-2$ vertices (light and dark brown nodes in Fig. 4) with degrees of 3, resulting in a more balanced induced graph.

Moving from $l = k$ to $l = 1$, more gauge cancellation is involved to ensure all logical operators become 2-local. Consequently, the connectivity induced by the logical operators overlaps less with that induced by the gauges. (As seen in Fig. 4, the number of green nodes and green edges excluding those connecting brown nodes increases from right to left.) In the $l = 1$ case, while the number of physical qubits required for the connectivity induced by the logical operators is smaller than in the $l = k$ case, additional qubits and edges are needed for the gauges, forming a graph with one degree more than that of the $l = k$ case. On the other hand, when $l = k$, the logical operators and gauges require the same connectivity, making the graph more balanced.

The degree of the induced graph increases linearly with the number of logical qubits, which is undesirable for practical implementation. Therefore, we aim to map it to a new graph that is more feasible or aligns well with the connectivity of existing hardware.

5.1 Mapping induced graph to mapped graph

Denote the set of physical qubits as V. Connect the physical qubits to support all logical operators $\hat{X}$, $\hat{Z}$, $\hat{X}\hat{X}$, $\hat{Z}\hat{Z}$, and gauge generators, resulting in the induced graph G illustrated in Fig. 4. The objective is to map the induced graph $G(V, E)$ to another graph $G_{\mathrm{m}}(V_{\mathrm{m}}, E_{\mathrm{m}})$, where G_{m} represents an existing hardware connectivity graph or a more feasible hardware graph, we call

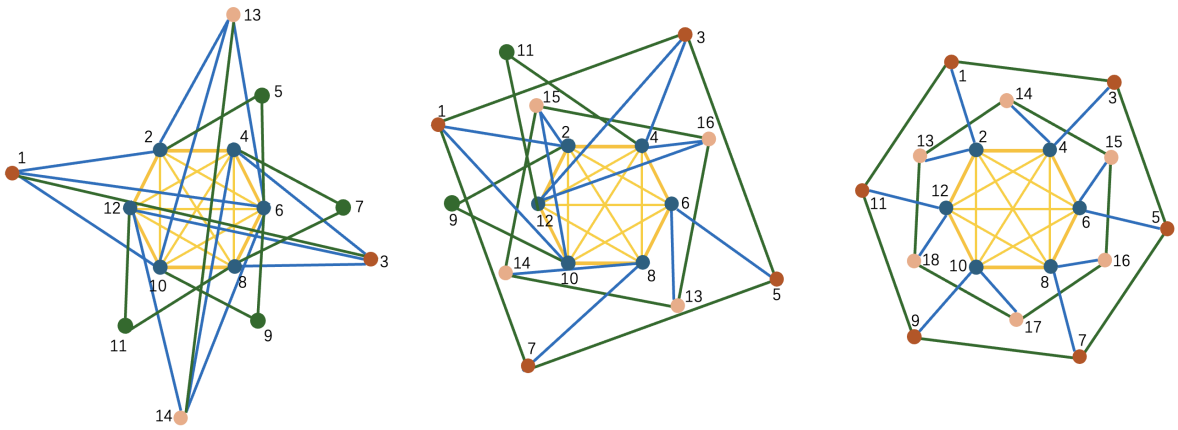


Figure 4: The induced graphs required to support the single-qubit logical operators, two-qubit logical operators, and gauge operators are shown for codes with $k = 3$ and $l = 1, 2, 3$, whose matrix representations are given in Eq. (19). Each node represents a physical qubit, whose index [as in Eq. (27)] is a node label. The dark blue nodes correspond to qubits located on the superdiagonal, which must be coupled to support the two-qubit logical operators $\hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$, with the yellow edges representing these couplings. The light brown nodes represent the qubits in the last row, and the dark brown nodes represent the qubits in the first column. The dark blue edges indicate the connectivity required for $\hat{X}$ and $\hat{Z}$. The green nodes represent qubits along the lower diagonal, with green edges showing the additional couplings required for gauge operators. The graphs shown in Fig. 2 are subgraphs of those shown here, representing just the connectivity required for the penalty Hamiltonians.

it the mapped graph. The distance between two vertices in $G_{\rm m}$ is quantified using the Manhattan distance.

Definition 8 (Manhattan Distance)**.** *Consider a graph $G(V, E)$. The Manhattan distance between $x, y \in V$, denoted as $m(x, y)$, is the length of the shortest path from x to y. In other words, $m(x, y)$ is the smallest number of edges in E necessary to connect x and y.*

Since we use undirected graphs, we have $m(x, y) = m(y, x)$. In addition, we define the total mapped Manhattan distance, or just the total Manhattan distance for short, as

Definition 9 (Total Manhattan Distance)**.** *Given two graphs $G(V, E)$ and $G_{\rm m}(V_{\rm m}, E_{\rm m})$ and a map from V to $V_{\rm m}$, the total Manhattan distance of $G_{\rm m}$ given G is defined as:*

$$m^E_{\rm total}(E_{\rm m}) = \sum_{(x,y)\in E} m_{E_{\rm m}}(x', y'), \tag{25}$$

where $x', y' \in V_{\rm m}$ are the mapped vertices corresponding to $x, y \in V$, and $m_{E_{\rm m}}(x', y')$ denotes the Manhattan distance between x' and y' in $G_{\rm m}$.

In the analog quantum computation model, since SWAP gates are not explicitly applied, the interactions in Eq. (22b) are implemented simultaneously. Therefore, although some edges are shared by both the logical operators and the gauges, they are counted only once. The goal is to find a mapping from V to $V_{\rm m}$ that minimizes the total Manhattan distance, equivalent to minimizing the number of SWAP gates required in the circuit model.

Another useful metric is the following:

Definition 10 (Average Manhattan distance)**.** *The average Manhattan distance is the average over the original edges*

$$\tilde{m} = \frac{m^E_{total}(E')}{|E|} = \frac{\sum_{(x,y)\in E} m_{E'}(x', y')}{|E|}. \tag{26}$$

A smaller average Manhattan distance is preferred.

5.2 Optimal mappings from a brute-force search

We present the mapping results for four cases with $m \in \{4, 5\}$ and $l \in \{1, 2\}$, and explore seven potential hardware layouts: Union Jack, square (Rigetti's Ankaa), triangular, IBM's heavy-hex, hexagonal, kagome, and Rigetti's Aspen, as depicted in Fig. 5. Given the NP-hardness of the mapping problem, we employ a brute-force search to find the optimal mappings for modest problem sizes. For larger problem sizes, the mixed-integer programming method provided in

Appendix E can be employed. Detailed optimal mapping results obtained through brute-force search are tabulated in Table 3.

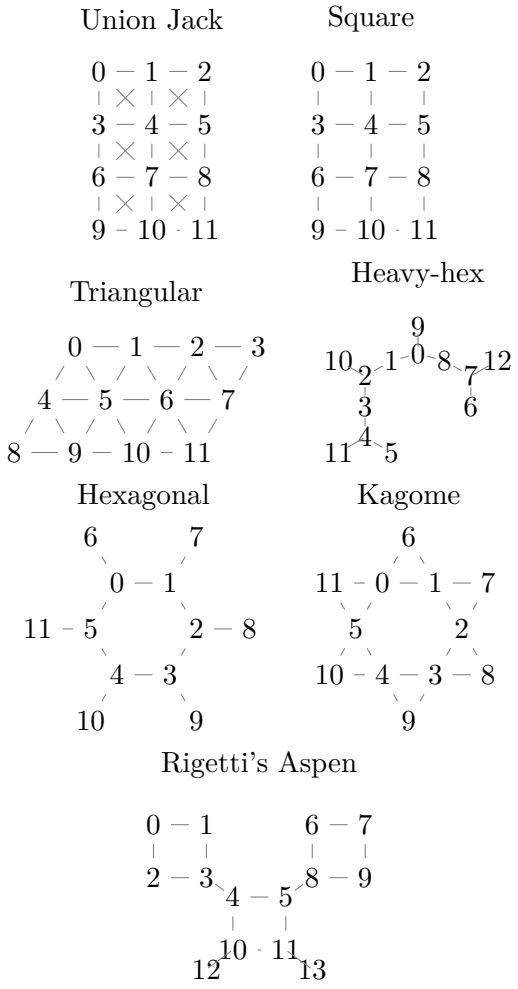


Figure 5: The seven mapped graphs (G_{m}) we consider. Numbers denote indices of vertices, which are used in the mapping. Solid lines denote enabled connectivity. We choose small portions of periodically repeating patterns in each graph type due to computational limitations.

In Table 3, the position in $[\,\cdot\,]$ represents the index of the physical qubits. We label the physical qubits from left to right and top to bottom, incrementing the index by one at each step. The number inside $[\,\cdot\,]$ corresponds to the index of a vertex in the mapped graph. For example, for the $[[10, 3, 5, 2]]$ code, when we map it to the square layout, the resulting vector representing the mapping is $[0, 1, 3, 4, 6, 5, 9, 10, 7, 8]$, and the 6 in the fifth entry of the vector means the physical qubit indexed as 5 is mapped to the vertex labeled 6 in the square layout in Fig. 5.

The properties of the mapped graphs are presented in Table 4. The first column lists the total Manhattan distances, while the second column provides the average Manhattan distances. Based on the criteria discussed in Section 7 below, the Union Jack graph outperforms all other layouts across all seven graphs, which can be attributed to its high degree and balanced structure. Despite the fact that the example is relatively small and the induced graph's degree aligns well with the Union Jack layout, larger examples may exhibit slight variations. However, due to its higher degree, the Union Jack graph is better suited for accommodating more complex graphs. At the other end of the spectrum, the heavy-hex graph performs the worst, due to its low degree and lack of balance.

6 Examples

We use the $[[14, 6, 6, 2]]$ and $[[12, 4, 6, 2]]$ codes as examples to provide a step-by-step explanation of the entire pipeline, which includes obtaining the 2-local logical operators from Eq. (19a) using gauge cancellations, constructing the induced graph G based on the obtained logical operators and gauge generators, and mapping the induced graph G to a mapped graph G_{m}.

6.1 $[[14, 6, 6, 2]]$ code

The $[[14, 6, 6, 2]]$ is the $l = 1$, $k = 3$ case of the $[[4k + 2, 2k, 2k + 2l - 2, 2]]$ subfamily of trapezoid codes. Recall that $l = 1$ yields the largest penalty gap and the highest code rate, so this is an example of particular interest.

6.1.1 Logical operators

Each entry 1 in Eq. (15) represents a physical qubit. These qubits are indexed from left to right and from top to bottom in Eq. (27) as follows:

$$\begin{bmatrix} 1 & 2 & & & & & & \\ 3 & & 4 & & & & & \\ & 5 & & 6 & & & & \\ & & 7 & & 8 & & & \\ & & & 9 & & 10 & & \\ & & & & 11 & & 12 & \\ & & & & & 13 & 14 & \end{bmatrix}. \tag{27}$$

Logical operators can be represented by the operator matrices in Eq. (19a). Each row is the bit-string representation of a $\hat{X}$ or $\hat{Z}$. A 1 in the i'th

Optimal mapping results				
Mapped graph	$[[8,3,3,2]]$	$[[10,3,5,2]]$	$[[10,4,4,2]]$	$[[12,4,6,2]]$
Union Jack	[1,3,5,7,0,4,10,6]	[0,3,1,4,2,6,5,7,8,10]	[1,3,5,7,0,4,11,8,6,10]	[0,3,1,4,2,6,5,7,11,8,10,9]
Square	[1,3,2,5,0,4,8,7]	[0,1,3,4,6,5,9,10,7,8]	[0,1,3,6,2,4,10,7,5,8]	[0,1,3,4,6,7,9,10,2,5,8,11]
Heavy-hex	[0,1,8,3,10,2,5,4]	[4,0,3,1,2,8,10,9,7,6]	[0,1,6,7,3,2,12,8,10,9]	[3,0,4,1,5,2,11,10,6,7,8,9]
Triangular	[0,1,4,5,2,6,9,10]	[0,1,4,5,7,6,3,2,10,11]	[0,1,4,5,2,6,8,9,11,10]	[0,4,1,5,2,6,3,7,8,9,10,11]
Hexagonal	[0,1,4,3,7,2,9,8]	[6,0,7,1,8,2,4,5,3,9]	[0,1,5,3,7,2,10,4,8,9]	[6,0,7,1,8,2,9,3,11,5,4,10]
Kagome	[0,1,4,3,6,2,8,7]	[6,0,1,5,2,11,3,9,4,10]	[0,1,5,3,6,2,9,4,7,8]	[0,1,11,2,5,3,10,4,6,7,8,9]
Rigetti	[1,3,0,2,5,4,11,10]	[0,1,2,3,5,4,8,11,10,12]	[1,3,6,5,2,4,9,8,10,11]	[0,1,2,3,10,4,11,5,6,7,9,8]

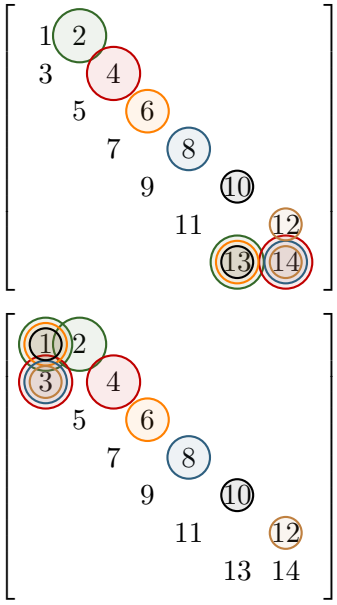
Table 3: The optimal mapping results. The position in $[\,\cdot\,]$ is the index of physical qubits. The number inside $[\,\cdot\,]$ is the index of a vertex in the mapped graph as in Fig. 5. For example, for a $[[10,3,5,2]]$ code mapped to the square layout, the mapping result is $[0,1,3,4,6,5,9,10,7,8]$. The 6 in the fifth entry means that the physical qubit in row 2 and column 0 is mapped to the vertex labeled 6 in the square layout in Fig. 5.

Graph properties from the mapping results								
Mapped graph	$[[8,3,3,2]]$		$[[10,3,5,2]]$		$[[10,4,4,2]]$		$[[12,4,6,2]]$	
	total	avg	total	avg	total	avg	total	avg
Union Jack	13	1	16	1.067	21	1.05	22	1.1
Triangular	15	1.154	18	1.2	25	1.25	24	1.2
Square	17	1.308	20	1.333	30	1.5	24	1.2
Kagome	18	1.385	21	1.4	29	1.45	30	1.5
Hexagonal	20	1.538	25	1.667	32	1.6	34	1.7
Rigetti Aspen	20	1.538	24	1.6	34	1.7	36	1.8
IBM Heavy-hex	23	1.769	30	2	41	2.05	44	2.2

Table 4: Graph properties from the mapping results. The "total" columns show the total Manhattan distances, the "avg" shows the average Manhattan distances. The mapped graphs appear in order of performance ranking, with the Union Jack graph exhibiting the lowest (i.e., best) total and average Manhattan distance.

position of the bitstring indicates that the i'th row or column in Eq. (27) is involved in the corresponding logical operator. For the $[[14,6,6,2]]$ code, the logical operators are as follows:

$$
\begin{aligned}
\hat{X}_1 &= 0101010 = X_2X_5X_6X_9X_{10}X_{13} = X_2X_{13},\\
\hat{X}_2 &= 0010101 = X_4X_7X_8X_{11}X_{12}X_{14} = X_4X_{14},\\
\hat{X}_3 &= 0001010 = X_6X_9X_{10}X_{13} = X_6X_{13},\\
\hat{X}_4 &= 0000101 = X_8X_{11}X_{12}X_{14} = X_8X_{14},\\
\hat{X}_5 &= 0000010 = X_{10}X_{13},\\
\hat{X}_6 &= 0000001 = X_{12}X_{14},\\
\hat{Z}_1 &= 1000000 = Z_1Z_2,\\
\hat{Z}_2 &= 0100000 = Z_3Z_4,\\
\hat{Z}_3 &= 1010000 = Z_1Z_2Z_5Z_6 = Z_1Z_6,\\
\hat{Z}_4 &= 0101000 = Z_3Z_4Z_7Z_8 = Z_3Z_8,\\
\hat{Z}_5 &= 1010100 = Z_1Z_2Z_5Z_6Z_9Z_{10} = Z_1Z_{10},\\
\hat{Z}_6 &= 0101010 = Z_3Z_4Z_7Z_8Z_{11}Z_{12} = Z_3Z_{12}.
\end{aligned}
\tag{28}
$$



(29)

The upper matrix in Eq. (29) illustrates the connectivity for $\hat{X}$, while the lower matrix corresponds to $\hat{Z}$. Different colors are used to represent distinct logical qubits. Circles of the same color in both matrices represent logical operators associated with the same logical qubit $(\hat{X}_i, \hat{Z}_i)$. For instance, the green circles denote physical operators required to execute $\hat{X}_1$ and $\hat{Z}_1$. Note that circles of the same color from the left and right matrices overlap exactly once, and circles of different colors do not overlap. This arrangement reflects the commutation relation of the logical operators, i.e., $\{\hat{X}_i, \hat{Z}_i\} = 0$ and $[\hat{X}_i, \hat{Z}_j] = 0$ for $i \neq j$.

To construct $\hat{X}\hat{X}$, we multiply two $\hat{X}$ operators. For example, executing $\hat{X}_1\hat{X}_2$ corresponds to $X_2X_{13}X_4X_{14}$ (represented by two large red and two large green circles). Since $X_{13}X_{14}$ is a gauge we can discard it, allowing us to execute $\hat{X}_1\hat{X}_2$ using just the 2-local operator X_2X_4 [recall Lemma 3]. Following similar reasoning, we can derive the following two-body logical operators:

$$\begin{aligned}
&\hat{X}_1\hat{X}_2 = X_2X_4, \hat{X}_1\hat{X}_3 = X_2X_6, \hat{X}_1\hat{X}_4 = X_2X_8,\\
&\hat{X}_1\hat{X}_5 = X_2X_{10}, \hat{X}_1\hat{X}_5 = X_2X_{12}, \hat{X}_2\hat{X}_3 = X_4X_6,\\
&\hat{X}_2\hat{X}_4 = X_4X_8, \hat{X}_2\hat{X}_5 = X_4X_{10}, \hat{X}_2\hat{X}_6 = X_4X_{12},\\
&\hat{X}_3\hat{X}_4 = X_6X_8, \hat{X}_3\hat{X}_5 = X_6X_{10}, \hat{X}_3\hat{X}_6 = X_6X_{12},\\
&\hat{X}_4\hat{X}_5 = X_8X_{10}, \hat{X}_4\hat{X}_6 = X_8X_{12}, \hat{X}_5\hat{X}_6 = X_{10}X_{12}.
\end{aligned} \tag{30}$$

The $\hat{Z}\hat{Z}$ operators exhibit a similar structure; we simply replace X with Z, noting that Z_1Z_2 is also a gauge operator.

6.1.2 Induced graph G

To implement the universal encoded Hamiltonian Eq. (22b) on hardware, we need to ensure that the physical qubits supporting all the aforementioned logical operators and gauge operators are coupled.

We begin by examining the connectivity induced by $\hat{X}$ and $\hat{Z}$. As illustrated in Eq. (29), for both $\hat{X}$ and $\hat{Z}$, half of the qubits on the superdiagonal must be coupled to qubit 13 and qubit 1, while the other half must connect to qubit 14 and qubit 3. The resulting connectivity is depicted in the top left panel of Fig. 6.

Next, for $\hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$, we observe that they lead to the same connectivity. As shown in Eq. (30), the qubits on the superdiagonal must be coupled. This results in a fully connected graph with 6 vertices, as illustrated in the top middle panel of Fig. 6.

Finally, since the penalty Hamiltonian consists of gauge generators, we need to incorporate additional qubits and edges. Specifically, we need to connect qubit 1 to qubit 3 and qubit 13 to qubit 14. We then add the qubits on the lower diagonal, ensuring that each connects to two other qubits, one in the same row and the other in the same column. The result is shown in the top right panel of Fig. 6.

Thus, we have the following edge set:

$$\begin{aligned}
E = \{&(2,13),(4,14),(6,13),(8,14),(10,13),\\
&(12,14),(1,2),(3,4),(1,6),(3,8),(1,10),\\
&(3,12),(2,4),(2,6),(2,8),(2,10),(2,12),\\
&(4,6),(4,8),(4,10),(4,12),(6,8),(6,10),\\
&(6,12),(8,10),(8,12),(10,12),(1,3),\\
&(2,5),(4,7),(6,9),(8,11),(5,6),(7,8),\\
&(9,10),(11,12),(13,14)\}.
\end{aligned} \tag{31}$$

6.1.3 Mapped graph G_{m} with Union Jack

The induced graph (Fig. 6, right) does not align with any existing hardware connectivity graphs. To illustrate the mapping to a more feasible graph, we select the Union Jack as the target graph and demonstrate how to interpret the results.

The Union Jack layout is shown in the left panel of Fig. 7. Given the layout, we can obtain

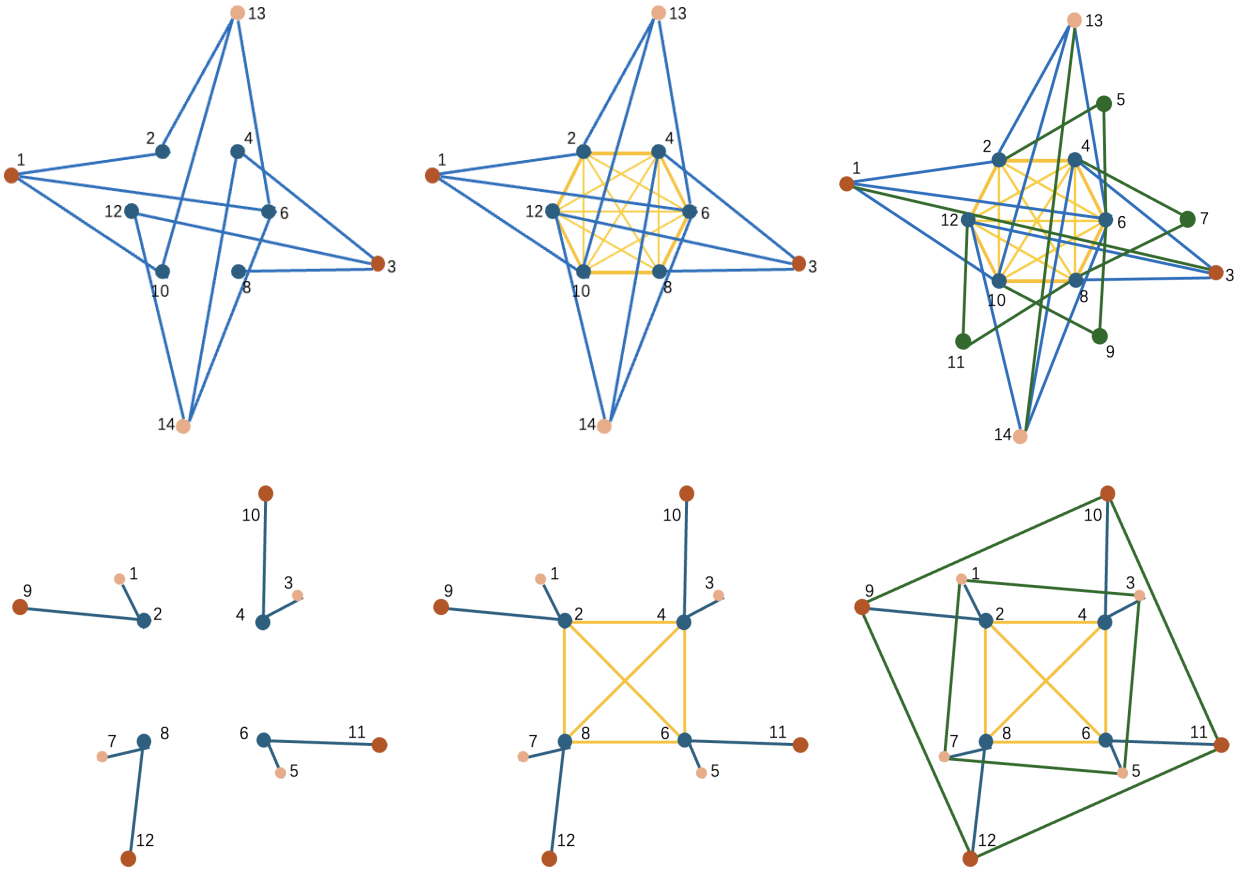


Figure 6: Top row: steps to construct the induced graph G for the $[[14, 6, 6, 2]]$ code. The left panel shows the connectivity required for $\hat{X}$ and $\hat{Z}$, the middle panel for $\hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$, and the right panel for the gauge generators. Bottom row: the same for the $[[12, 4, 6, 2]]$ code.

the edge set as:

$$
\begin{aligned}
E_{\mathrm{m}} = \{&(0,1),(0,4),(0,5),(1,2),(1,4),(1,5),\\
&(1,6),(2,3),(2,5),(2,6),(2,7),(3,6),\\
&(3,7),(4,5),(4,8),(4,9),(5,6),(5,8),\\
&(5,9),(5,10),(6,7),(6,9),(6,10),(6,11),\\
&(7,10),(7,11),(8,9),(8,12),(8,13),\\
&(9,10),(9,12),(9,13),(9,14),(10,11),\\
&(10,13),(10,14),(10,15),(11,14),\\
&(11,15),(12,13),(13,14),(14,15)\}. \quad (32)
\end{aligned}
$$

The mapping results are presented by the vector $[8, 6, 13, 14, 1, 5, 15, 10, 0, 4, 11, 9, 2, 7]$, which is of the same format as in Table 3. As explained in Section 5.2, the position i in the vector denotes the index of the physical qubit in the induced graph, while the value of the i'th entry denotes the location of that qubit on the mapped graph. Thus, we have the following correspondence:

8	6	13	14	1	5	15	10	0	4	11	9	2	7
1	2	3	4	5	6	7	8	9	10	11	12	13	14

The first row is the mapping result vector, which contains the position of the physical qubits, while the second row shows the indices of physical qubits. We can then label the physical qubits on the mapped graph, as shown in the right panel of Fig. 7.

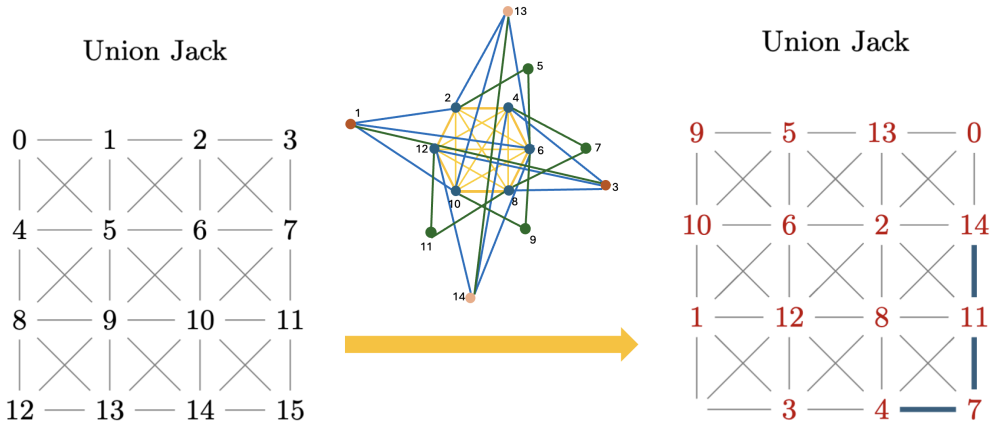


Figure 7: Interpretation of the mapping result for the $[[14, 6, 6, 2]]$ code. The mapping result is $[8, 6, 13, 14, 1, 5, 15, 10, 0, 4, 11, 9, 2, 7]$. The red numbers represent the indices of the physical qubits, corresponding to the indices in the left panel, while the black numbers represent the positions of these physical qubits in the mapped graph. Blue edges denote the shortest path to connect qubits 4 and 14 in G_{m}.

To illustrate how to calculate the Manhattan distance, consider, e.g., b $\hat{X}_2 = X_4X_{14}$. Qubit 4 is mapped to vertex 14, and qubit 14 is mapped to vertex 7. The induced map contains the edge $4-14$, which support $\hat{X}_2$. In the mapped graph, the Manhattan distance between vertices 7 and 14 is used as the metric for the geometric locality of qubits 4 and 14 on G_{m}. One of the shortest paths from 7 to 14 is $7-11-15-14$, resulting in $m(7, 14) = 3$. To compute the total Manhattan distance of G_{m} given E, this process is repeated for all the edges e in the set E, and the distances are summed.

6.2 $[[12, 4, 6, 2]]$ code

We use the $[[12, 4, 6, 2]]$ code as another example to illustrate a different case when $l = k$. We focus on the induced graph G and the mapped graph G_{m} in this example. The physical qubits in the A matrix for the $[[12, 4, 6, 2]]$ code are indexed as follows:

$$\begin{bmatrix} 1 & 2 & & & & \\ 3 & & 4 & & & \\ 5 & & & 6 & & \\ 7 & & & & 8 & \\ & 9 & 10 & 11 & 12 & \end{bmatrix}. \tag{33}$$

Following the same steps as Section 6.1.1, we obtain the logical operators as:

$$\begin{aligned} &\hat{X}_1 = X_2X_9, \hat{X}_2 = X_4X_{10}, \hat{X}_3 = X_6X_{11}, \\ &\hat{X}_4 = X_8X_{12}, \hat{Z}_1 = Z_1Z_2, \hat{Z}_2 = Z_3Z_4, \\ &\hat{Z}_3 = Z_5Z_6, \hat{Z}_4 = Z_7Z_8, \\ &\hat{X}_1\hat{X}_2 = X_2X_4, \hat{X}_1\hat{X}_3 = X_2X_6, \hat{X}_1\hat{X}_4 = X_2X_8, \\ &\hat{X}_2\hat{X}_3 = X_4X_6, \hat{X}_2\hat{X}_4 = X_4X_8, \hat{X}_3\hat{X}_4 = X_6X_8, \\ &\hat{Z}_1\hat{Z}_2 = Z_2Z_4, \hat{Z}_1\hat{Z}_3 = Z_2Z_6, \hat{Z}_1\hat{Z}_4 = Z_2Z_8, \\ &\hat{Z}_2\hat{Z}_3 = Z_4Z_6, \hat{Z}_2\hat{Z}_4 = Z_4Z_8, \hat{Z}_3\hat{Z}_4 = Z_6Z_8. \end{aligned} \tag{34}$$

In the induced graph G, we need to couple pairs of qubits that are necessary to support logical operators and gauge operators. For $\hat{X}$ and $\hat{Z}$, a qubit on the superdiagonal must be coupled to a qubit in the first column and a qubit in the last row, which results in the connectivity shown in the bottom left panel of Fig. 6. For $\hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$, the qubits on the superdiagonal must be coupled, which leads to a fully connected graph with 4 vertices, as illustrated in the bottom middle panel of Fig. 6. Finally, the qubits in the first column and the last row form two cyclic squares to support the gauge operators, as shown in the bottom right panel of Fig. 6. Thus, the edge set is:

$$\begin{aligned} E = \{&(2, 9), (4, 10), (6, 11), (8, 12), (1, 2), (3, 4), \\ &(5, 6), (7, 8), (2, 4), (2, 6), (2, 8), (4, 6), \\ &(4, 8), (6, 8), (1, 3), (3, 5), (5, 7), (7, 1), \\ &(9, 10), (10, 11), (11, 12), (12, 9)\}. \end{aligned} \tag{35}$$

This G is of low degree and is balanced, which is implementable on current devices. However, to demonstrate the results of larger system size, we map it to a kagome layout. The kagome layout is shown in the left panel of Fig. 8. Given the layout, we can obtain the edge set as:

$$\begin{aligned} E_{\mathrm{m}} = \{&(0,1),(1,2),(2,3),(3,4),(4,5),\\ &(5,0),(0,6),(1,6),(1,7),(2,7),\\ &(2,8),(3,8),(3,9),(4,9),(4,10),\\ &(5,10),(5,11),(0,11)\}. \end{aligned} \tag{36}$$

The optimal mapping result is give by the vector $[0,1,11,2,5,3,10,4,6,7,8,9]$. Thus, we have the following correspondence:

0	1	11	2	5	3	10	4	6	7	8	9
1	2	3	4	5	6	7	8	9	10	11	12

To illustrate how to calculate the Manhattan distance, consider, e.g., $\hat{X}_2\hat{X}_4 = X_4X_8$. Qubit 4 is mapped to vertex 2, and qubit 8 is mapped to vertex 4. The induced map contains the edge $4-8$, which support $\hat{X}_2\hat{X}_4$. One of the shortest paths from 2 to 4 on G_m is $2-3-4$, resulting in $m(2,4)=2$.

7 Discussion

We now revisit the five criteria we outlined in Section 2 and discuss them in light of our results.

1. Code rate: For $[[4k+2l, 2k, g, 2]]$ trapezoid codes, we aim to maximize the code rate $k/(2k+l)$. We showed in Section 4.1.1 that codes with $l=1$ have the maximum code rate of $r = k/(2k+1)$.

2. Physical Locality: With our choice of optimal dressed logical operators, all the logical operators required are 2-body, as detailed in Lemma 3. We have also demonstrated that bare logical operators cannot all be 2-local in Lemma 4.

3. Induced graph properties: The induced graph is 2-local. However, the degree of this graph grows linearly with the number of logical qubits, complicating implementation. Ideally, every logical operator and gauge is geometrically 2-local in a fixed degree mapped graph. However, none of the 2D geometries we explored satisfies this ideal for $l < k$ codes. Codes with $l = k$ ($[[6k, 2k, 2]]$ codes) inherently have almost-constant degree induced graphs (if we ignore the fully connected component needed for logical XX and ZZ interactions), as shown in [39]. Instead, we use the total and average Manhattan distances as metrics to measure the geometrical locality of the code. The total Manhattan distance can be interpreted as the number of SWAPs that would be needed in the circuit model if we were to apply all dressed logical operators exactly once.

4. Mapped graph properties: We provide an explicit mapping scheme to feasible hardware graphs in Section 5 and illustrate it in detail in Section 6. Another approach involves constructing a possibly optimal new graph based on the induced graph with constraints such as having a constant degree, but this method is harder to implement.

5. Penalty Gap: We numerically calculate the penalty gaps of a subset of trapezoid codes (Fig. 3). Fixing m, for $l = 1, 2$ the gap closes as a power law; for $l = 1$ the penalty Hamiltonian is equivalent to the 1D compass model, for which the gap is known to close as the inverse of the number of (logical) qubits, which is optimal. For $l > 2$ the gap closes exponentially in l. It also closes exponentially in m at fixed l. Recall that the number of logical qubits is $k = \lfloor m/2 \rfloor$ and the number of physical qubits is $4k+2l$ for odd m or $4k+2(l-1)$ for even m. Thus, codes with smaller l provide exponentially larger penalty gaps in the number of physical qubits. Trapezoid codes with $l < k$ therefore have larger penalty gaps than the $[[6k, 2k, 2]]$ code family in Ref. [39].

8 Conclusions

Analog, Hamiltonian quantum computation can be effectively protected against decoherence via quantum error detection codes implementing energy penalties. Our goal in this work was to identify a family of codes that accomplish this goal subject to common constraints imposed by hardware limitations. This includes no higher

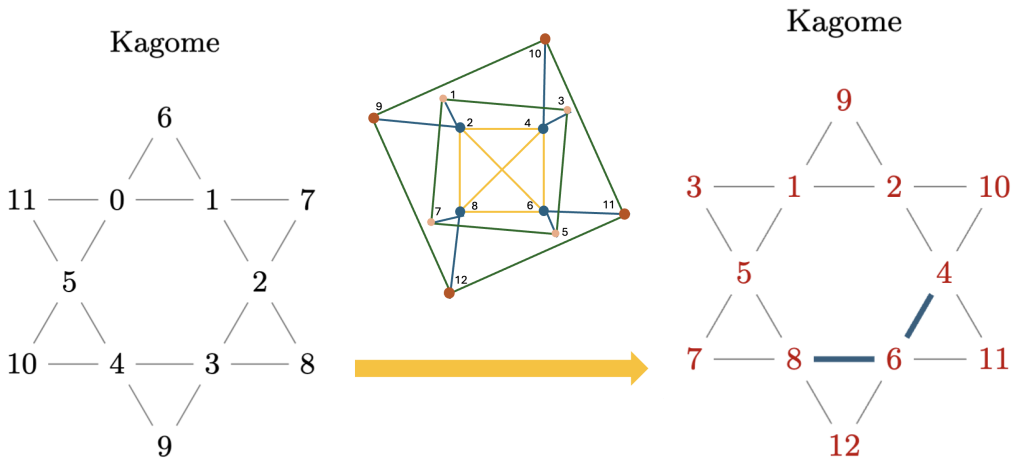


Figure 8: The mapping results. The mapping result is $[0, 1, 11, 2, 5, 3, 10, 4, 6, 7, 8, 9]$. The red numbers represent the indices of the physical qubits, corresponding to the indices in the left panel, while the black numbers represent the positions of these physical qubits in the mapped graph. Blue edges denote the shortest path to connect qubits 4 and 8 in G_{m}.

than 2-local interactions, geometric locality, and compatibility with existing qubit connectivity graphs. It is also crucial to maintain as large an energy gap as possible above the ground subspace of the penalty Hamiltonian, in order to prevent deviations from the intended quantum evolution.

In light of these goals, we studied subsystem codes derived from Bravyi's A matrices, which are naturally suitable for 2D qubit interaction graphs. This led to the identification of a code family we call 'trapezoid codes,' which exhibit favorable characteristics compatible with our criteria. We examined the trade-offs associated with various factors affecting the practical implementation of trapezoid codes, as detailed in Section 7. In particular, we demonstrated that every code within the trapezoid family possesses all 2-local one-qubit logical operators and two-qubit dressed logical operators.

We found that the optimal code rate achievable within this family is $k/(2k+1)$, a rate manifested by the $[[4k+2, 2k, g, 2]]$ code for any positive integer k (here $g = 2k + 2l - 2$). This code can be mapped to feasible hardware connectivity graphs, as discussed in Section 5. Moreover, the $[[4k + 2, 2k, g, 2]]$ code also has the largest penalty gap among the codes in the trapezoid family, as can be seen in Fig. 3, and this gap scales as $1/(2k)$, ensuring a slower scaling with problem size k than for most problems of interest in AQC.

We used the total Manhattan distance as a performance metric to evaluate the geometric locality of the mapping results and examined a set of common native hardware graph geometries. Within this set, we found that the Union Jack lattice exhibits the lowest Manhattan distance overall (Table 4).

In conclusion, the $[[4k+2, 2k, g, 2]]$ code implemented on a native Union Jack lattice presents an optimal combination of favorable properties, and appears to be a promising choice for future implementations.

Our framework is flexible enough to allow each code within the trapezoid family to be evaluated based on other criteria. We hope that the results reported here will contribute to the advancement of error-suppressed, analog Hamiltonian quantum computing.

Acknowledgments

This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number W911NF2310255, and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) and the Army Research Office, under the Entangled Logical Qubits program through Cooperative Agreement Number W911NF-23-2-0216. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of IARPA, the Army Research Office, or

the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. We thank Seolhwa Kim, Yanlin Cheng, Adam Bistagne, and Larry Gu for providing an early analysis that inspired some of the results presented here.

References

[1] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, *Quantum Computation by Adiabatic Evolution*, arXiv:quant-ph/0001106 (2000).

[2] D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, *Adiabatic quantum computation is equivalent to standard quantum computation*, SIAM Journal on Computing **37**, 166–194 (2007).

[3] T. Albash and D. A. Lidar, *Adiabatic quantum computation*, Reviews of Modern Physics **90**, 015002– (2018).

[4] T. Kadowaki and H. Nishimori, *Quantum annealing in the transverse ising model*, Phys. Rev. E **58**, 5355–5363 (1998).

[5] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, *Perspectives of quantum annealing: Methods and implementations*, Reports on Progress in Physics (2020).

[6] E. J. Crosson and D. A. Lidar, *Prospects for quantum enhancement with diabatic quantum annealing*, Nature Reviews Physics **3**, 466–489 (2021).

[7] P. Zanardi and M. Rasetti, *Holonomic quantum computation*, Physics Letters A **264**, 94–99 (1999).

[8] L. M. Duan, J. I. Cirac, and P. Zoller, *Geometric manipulation of trapped ions for quantum computation*, Science **292**, 1695–1697 (2001).

[9] J. Zhang, T. H. Kyaw, S. Filipp, L.-C. Kwek, E. Sjöqvist, and D. Tong, *Geometric and holonomic quantum computation*, Physics Reports **1027**, 1–53 (2023).

[10] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, V. Vuletić, and M. D. Lukin, *Probing many-body dynamics on a 51-atom quantum simulator*, Nature **551**, 579–584 (2017).

[11] A. D. King, S. Suzuki, J. Raymond, A. Zucca, T. Lanting, F. Altomare, A. J. Berkley, S. Ejtemaee, E. Hoskinson, S. Huang, E. Ladizinsky, A. J. R. MacDonald, G. Marsden, T. Oh, G. Poulin-Lamarre, M. Reis, C. Rich, Y. Sato, J. D. Whittaker, J. Yao, R. Harris, D. A. Lidar, H. Nishimori, and M. H. Amin, *Coherent quantum annealing in a programmable 2,000 qubit ising chain*, Nature Physics **18**, 1324–1328 (2022).

[12] E. Altman, K. R. Brown, G. Carleo, L. D. Carr, E. Demler, C. Chin, B. DeMarco, S. E. Economou, M. A. Eriksson, K.-M. C. Fu, M. Greiner, K. R. A. Hazzard, R. G. Hulet, A. J. Kollár, B. L. Lev, M. D. Lukin, R. Ma, X. Mi, S. Misra, C. Monroe, K. Murch, Z. Nazario, K.-K. Ni, A. C. Potter, P. Roushan, M. Saffman, M. Schleier-Smith, I. Siddiqi, R. Simmonds, M. Singh, I. B. Spielman, K. Temme, D. S. Weiss, J. Vučković, V. Vuletić, J. Ye, and M. Zwierlein, *Quantum simulators: Architectures and opportunities*, PRX Quantum **2**, 017003–(2021).

[13] E. Farhi and S. Gutmann, *Quantum computation and decision trees*, Physical Review A **58**, 915–928 (1998).

[14] A. M. Childs, D. Gosset, and Z. Webb, *Universal computation by multiparticle quantum walk*, Science **339**, 791–794 (2013).

[15] O. Mülken and A. Blumen, *Continuous-time quantum walks: Models for coherent transport on complex networks*, Physics Reports **502**, 37–87 (2011).

[16] O. Oreshkov, T. A. Brun, and D. A. Lidar, *Fault-tolerant holonomic quantum computation*, Phys. Rev. Lett. **102**, 070502– (2009).

[17] S. P. Jordan, E. Farhi, and P. W. Shor, *Error-correcting codes for adiabatic quantum computation*, Phys. Rev. A **74**, 052322 (2006).

[18] D. Gottesman, *Class of quantum error-correcting codes saturating the quantum hamming bound*, Phys. Rev. A **54**, 1862 (1996).

[19] A. R. Calderbank, E. M. Rains, P. M. Shor, and N. J. A. Sloane, *Quantum error correction via codes over $gf(4)$*, IEEE Transactions on Information Theory **44**, 1369–1387 (1998).

[20] S. Bravyi and M. Vyalyi, *Commutative ver-*

sion of the k-local hamiltonian problem and common eigenspace problem, Quantum Inf. and Comp. **5**, 187–215 (2005).

[21] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological quantum memory*, Journal of Mathematical Physics **43**, 4452–4505 (2002).

[22] R. Alicki, M. Fannes, and M. Horodecki, *A statistical mechanics view on kitaev's proposal for quantum memories*, Journal of Physics A: Mathematical and Theoretical **40**, 6451 (2007).

[23] K. C. Young, R. Blume-Kohout, and D. A. Lidar, *Adiabatic quantum optimization with the wrong hamiltonian*, Phys. Rev. A **88**, 062314– (2013).

[24] K. C. Young, M. Sarovar, and R. Blume-Kohout, *Error suppression and error correction in adiabatic quantum computation: Techniques and challenges*, Phys. Rev. X **3**, 041013– (2013).

[25] M. Sarovar and K. C. Young, *Error suppression and error correction in adiabatic quantum computation: non-equilibrium dynamics*, New J. of Phys. **15**, 125032 (2013).

[26] A. Ganti, U. Onunkwo, and K. Young, *Family of* $[[6k, 2k, 2]]$ *codes for practical, scalable adiabatic quantum computation*, Phys. Rev. A **89**, 042313– (2014).

[27] A. D. Bookatz, E. Farhi, and L. Zhou, *Error suppression in hamiltonian-based quantum computation using energy penalties*, Phys. Rev. A **92**, 022317– (2015).

[28] I. Marvian, *Exponential suppression of decoherence and relaxation of quantum systems using energy penalty*, arXiv:1602.03251 (2016).

[29] M. Marvian and D. A. Lidar, *Error suppression for hamiltonian quantum computing in markovian environments*, Physical Review A **95**, 032302– (2017).

[30] D. A. Lidar, *Arbitrary-time error suppression for markovian adiabatic quantum computing using stabilizer subspace codes*, Physical Review A **100**, 022326– (2019).

[31] A. Mizel and D. A. Lidar, *Three- and four-body interactions in spin-based quantum computers*, Phys. Rev. Lett. **92**, 077903– (2004).

[32] J. H. Gurian, P. Cheinet, P. Huillery, A. Fioretti, J. Zhao, P. L. Gould, D. Comparat, and P. Pillet, *Observation of a resonant four-body interaction in cold cesium rydberg atoms*, Physical Review Letters **108**, 023005– (2012).

[33] H.-N. Dai, B. Yang, A. Reingruber, H. Sun, X.-F. Xu, Y.-A. Chen, Z.-S. Yuan, and J.-W. Pan, *Four-body ring-exchange interactions and anyonic statistics within a minimal toric-code hamiltonian*, Nature Physics **13**, 1195–1200 (2017).

[34] L. Chirolli, A. Braggio, and F. Giazotto, *Cooper quartets in interacting hybrid superconducting systems*, Physical Review Research **6**, 033171– (2024).

[35] I. Marvian and D. A. Lidar, *Quantum error suppression with commuting hamiltonians: Two local is too local*, Phys. Rev. Lett. **113**, 260504– (2014).

[36] Z. Jiang and E. G. Rieffel, *Non-commuting two-local hamiltonians for quantum error suppression*, Quant. Inf. Proc. **16**, 89 (2017).

[37] M. Marvian and D. A. Lidar, *Error Suppression for Hamiltonian-Based Quantum Computation Using Subsystem Codes*, Phys. Rev. Lett. **118**, 030504– (2017).

[38] D. Poulin, *Stabilizer formalism for operator quantum error correction*, Phys. Rev. Lett. **95**, 230504 (2005).

[39] M. Marvian and S. Lloyd, *Robust universal hamiltonian quantum computing using two-body interactions*, arXiv:1911.01354 (2019).

[40] S. Bravyi, *Subsystem codes with spatially local generators*, Phys. Rev. A **83**, 012320 (2011).

[41] J. D. Biamonte and P. J. Love, *Realizable hamiltonians for universal adiabatic quantum computers*, Phys. Rev. A **78**, 012352 (2008).

[42] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, *Quantum annealing with manufactured spins*, Nature **473**, 194–198 (2011).

[43] S. Boixo, T. F. Ronnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, *Evidence for quan-*

tum annealing with more than one hundred qubits, Nat. Phys. **10**, 218–224 (2014).

[44] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. H. Amin, A. Y. Smirnov, M. Mohseni, and H. Neven, *Computational multiqubit tunnelling in programmable quantum annealers*, Nat Commun **7** (2016).

[45] T. Albash and D. A. Lidar, *Demonstration of a scaling advantage for a quantum annealer over simulated annealing*, Physical Review X **8**, 031016– (2018).

[46] S. Mandrà and H. G. Katzgraber, *A deceptive step towards quantum speedup detection*, Quantum Sci. Technol. **3**, 04LT01 (2018).

[47] M. Kowalsky, T. Albash, I. Hen, and D. A. Lidar, *3-regular three-xorsat planted solutions benchmark of classical and quantum heuristic optimizers*, Quantum Science and Technology **7**, 025008 (2022).

[48] E. Knill and R. Laflamme, *Theory of quantum error-correcting codes*, Phys. Rev. A **55**, 900–911 (1997).

[49] D. Bacon, *Operator quantum error-correcting subsystems for self-correcting quantum memories*, Phys. Rev. A **73**, 012340 (2006).

[50] P. Aliferis and A. W. Cross, *Subsystem fault tolerance with the bacon-shor code*, Phys. Rev. Lett. **98**, 220502 (2007).

[51] S. Burton, *Spectra of gauge code hamiltonians*, arXiv:1801.03243 (2018).

[52] Z. Nussinov and J. van den Brink, *Compass models: Theory and physical motivations*, Reviews of Modern Physics **87**, 1–59 (2015).

[53] P. Pfeuty, *The one-dimensional ising model with a transverse field*, Annals of Physics **57**, 79–90 (1970).

[54] W. Brzezicki, J. Dziarmaga, and A. M. Oleś, *Quantum phase transition in the one-dimensional compass model*, Physical Review B **75**, 134415– (2007).

[55] W. Brzezicki and A. M. Oleś, *Symmetry properties and spectra of the two-dimensional quantum compass model*, Phys. Rev. B **87**, 214421 (2013).

[56] W. Brzezicki and A. M. Oleś, *Exact solution for a quantum compass ladder*, Phys. Rev. B **80**, 014405 (2009).

[57] J. Dorier, F. Becca, and F. Mila, *Quantum compass model on the square lattice*, Phys. Rev. B **72**, 024448 (2005).

[58] G. Nannicini, L. S. Bishop, O. Günlük, and P. Jurcevic, *Optimal qubit assignment and routing via integer programming*, ACM Transactions on Quantum Computing **4**, 1–31 (2022).

A Equivalence of A matrix via permuting rows and columns

There exist multiple approaches to constructing Bravyi's A matrix corresponding to a specified set of code parameters $[[n, k, g, d]]$. Lemma 5 below demonstrates that these various forms are equivalent, as the code parameters are preserved under both row and column permutations. Specifically, while gauge generators undergo modifications through permutations, the gauge group and stabilizers remain invariant.

Lemma 5. *A code associated with a given A matrix is invariant under permutations of rows and columns of the same A matrix. I.e., the gauge group is invariant under permutations.*

Proof. Any permutations of rows and columns can be decomposed into a series of transpositions, $\prod_{i,j}(\sigma_i, \sigma_j)$, where $\sigma \in \{r, c\}$ denotes a row or a column. Without loss of generality, consider a transposition of two rows, (r_i, r_j). This does not change X-type gauge operators because the matrix structure is preserved within the same row. Z-type gauge operators are pairs of Z operators in the same column. Let $Z_{(i,a)}Z_{(k\neq i,a)}$ be a gauge operator such that one of its operators is in the i'th row. If $k = j$, the transposition (r_i, r_j) does not change the gauge operator. Otherwise, this gauge operator becomes $Z_{(j,a)}Z_{(k\neq j,a)}$. However, the column indices are dummy variables and can be relabelled without affecting physical operations. We recover the original gauge operators after relabelling the indices $i \to j, j \to i$. Since any transposition of any rows and columns preserves the gauge group $\mathcal{G}$, the permutations of any rows and columns also preserve the gauge group $\mathcal{G}$. □

To illustrate the result in Lemma 5, the three matrices in Eq. (37) are associated with the same code. In this work, we use the 'trapezoid' representation on the left hand side. The 1-entries are indexed to show that the gauge group $\mathcal{G}$ is preserved, e.g., X_aX_b is always presented across all permutations.

$$\begin{bmatrix} \mathbf{1}_a & \mathbf{1}_b & 0 & 0 & 0 \\ \mathbf{1}_j & 0 & \mathbf{1}_i & 0 & 0 \\ 0 & \mathbf{1}_c & 0 & \mathbf{1}_d & 0 \\ 0 & 0 & \mathbf{1}_h & 0 & \mathbf{1}_g \\ 0 & 0 & 0 & \mathbf{1}_e & \mathbf{1}_f \end{bmatrix} \equiv \begin{bmatrix} 0 & \mathbf{1}_a & 0 & 0 & \mathbf{1}_b \\ \mathbf{1}_f & 0 & \mathbf{1}_e & 0 & 0 \\ 0 & \mathbf{1}_j & 0 & \mathbf{1}_i & 0 \\ 0 & 0 & \mathbf{1}_d & 0 & \mathbf{1}_c \\ \mathbf{1}_g & 0 & 0 & \mathbf{1}_h & 0 \end{bmatrix} \equiv \begin{bmatrix} \mathbf{1}_a & \mathbf{1}_b & 0 & 0 & 0 \\ 0 & \mathbf{1}_c & \mathbf{1}_d & 0 & 0 \\ 0 & 0 & \mathbf{1}_e & \mathbf{1}_f & 0 \\ 0 & 0 & 0 & \mathbf{1}_g & \mathbf{1}_h \\ \mathbf{1}_j & 0 & 0 & 0 & \mathbf{1}_i \end{bmatrix} \tag{37}$$

The second matrix is obtained by permuting the columns $(1, 2, 3, 4, 5)$ into $(2, 5, 4, 3, 1)$ and rows $(1, 2, 3, 4, 5)$ into $(1, 3, 4, 5, 2)$ from the first matrix. The third matrix is obtained by permuting the columns $(1, 2, 3, 4, 5)$ into $(1, 2, 5, 3, 4)$ and rows $(1, 2, 3, 4, 5)$ into $(1, 5, 2, 4, 3)$ from the first matrix. The gauge generators associated with the three matrices are the same, as follows:

$$\mathcal{G} = \langle X_aX_b, X_jX_i, X_cX_d, X_hX_g, X_eX_f, Z_aZ_j, Z_bZ_c, Z_iZ_h, Z_dZ_e, Z_gZ_f \rangle. \tag{38}$$

B Reduction of search space's cardinality in searching for optimal logical operators

Recall that Definition 3 and Definition 5 establish how a bitstring parameterizes bare and dressed logical operators, respectively. Our goal is to construct the m optimal pairs of logical operators, $\hat{X}$ and $\hat{Z}$. By 'optimal pairs,' we refer to those minimizing the weight of $\hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$ terms. However, for a code defined by an $m \times m$ A matrix, there are 2^m possible configurations of bitstrings of length m that can parameterize logical operators, resulting in an exponentially large search space.

In this section, we exploit symmetries to reduce the search space and systematically exclude bitstrings that generate operators with locality greater than two. Let $\mathcal{S}_0$ denote the set of all possible bitstrings of length m, i.e., $\mathcal{S}_0 = \{\boldsymbol{a} \in \{0, 1\}^m\}$. The search space for logical operators of both X- and Z-type is given by $\mathcal{S}_0 \times \mathcal{S}_0$, which is exponentially large, as $|\mathcal{S}_0| = 2^m$.

Ideally, we want to achieve the highest code rate with purely 2-local $\hat{X}, \hat{Z}, \hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$ operators. However, since Ref. [39] has shown that the trapezoid code with $l = k$ has two-body interactions for every $\bar{X}, \bar{Z}, \hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$, it is unavoidable for $l < k$ codes with higher rates to have higher than 2-local $\hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$ operators, although all $\bar{X}, \bar{Z}$ operators are 2-local.

Using symmetries, we apply two successive reductions: $\mathcal{S}_0 \to \mathcal{S}_1 \to \mathcal{S}_2$, which reduce the cardinality of the search space from $|\mathcal{S}_0 \times \mathcal{S}_0| = 2^{2m}$ to $|\mathcal{S}_2 \times \mathcal{S}_2| = m^2(m-1)^2/4$, as summarized below in Theorem 2. Using this result on $l < k$ codes, we search the space to find m optimal pairs of 2-local logical operators $\hat{X}$ and $\hat{Z}$, such that the weight of $\hat{X}\hat{X}$ and $\hat{Z}\hat{Z}$ terms is minimized.

Theorem 2. *The size of the search space for 2-local logical operators is $|\mathcal{S}_2 \times \mathcal{S}_2| = m^2(m-1)^2/4$, where $\mathcal{S}_2$ is a set of binary bitstrings that parameterize 2-local logical operators.*

Proof. Using Lemma 9, there are $m(m-1)/2$ bitstrings that parameterize 2-local $\hat{Z}$ operators and likewise $\hat{X}$ operators. The reduced search space is therefore $\mathcal{S}_2 \times \mathcal{S}_2$ and its cardinality is $|\mathcal{S}_2 \times \mathcal{S}_2| = m^2(m-1)^2/4$. □

The rest of this section is devoted to building up to and proving Lemma 9.

Note that odd-m codes consistently exhibit a higher code rate compared to even-m codes [Eq. (14)], making them more desirable. Consequently, the results in this section focus on odd-m codes. Readers interested in even-m codes can apply analogous reasoning to derive the corresponding results.

B.1 Reduction by stabilizer symmetry ($\mathcal{S}_0 \to \mathcal{S}_1$)

Let $\boldsymbol{x} = x_1 x_2 \dots x_m$ and $\boldsymbol{z} = z_1 z_2 \dots z_m$ represent length-m binary bitstrings, where $x_i, z_i \in \{0,1\}$. We demonstrate that it is sufficient to restrict our consideration to $\boldsymbol{x}$ with $x_1 = 0$ and $\boldsymbol{z}$ with $z_m = 0$. This restriction resolves the degeneracy arising from equivalence under multiplication by the stabilizers S_X and S_Z, as demonstrated in Lemma 6 below. Imposing this symmetry reduces the search space to $\mathcal{S}_1 \times \mathcal{S}_1 = \{\boldsymbol{x} \in \{0,1\}^m | x_1 = 0\} \times \{\boldsymbol{z} \in \{0,1\}^m | z_m = 0\}$, which is half the size of the original space: $|\mathcal{S}_1| = |\mathcal{S}_0|/2$.

Lemma 6. *$\boldsymbol{x}$ ($\boldsymbol{z}$) and $\tilde{\boldsymbol{x}} = 1^m \oplus \boldsymbol{x}$ ($\tilde{\boldsymbol{z}} = 1^m \oplus \boldsymbol{z}$) parameterize X-type (Z-type) operators that are equivalent up to multiplication by a stabilizer.*

Proof. Without loss of generality, consider an X-type operator, $P^X(\boldsymbol{x})$, parameterized by a bitstring $\boldsymbol{x} = x_1 x_2 \dots x_m$. Writing it in terms of column operators, $P^X(\boldsymbol{x}) = \prod_{i=1}^m C_i^{x_i}$. Recall from Lemma 1 that the stabilizer $S_X = \prod_{i=1}^m C_i$, i.e., $S_X = P^X(1^m)$.

We can also write $P^X(\tilde{\boldsymbol{x}}) = \prod_{i=1}^m C_i^{\tilde{x}_i} = \prod_{i=1}^m C_i^{1 \oplus x_i} = \prod_{i=1}^m C_i \prod_{i=1}^m C_i^{x_i} = S_X P^X(\boldsymbol{x})$, which shows that $P^X(\boldsymbol{x})$ and $P^X(\tilde{\boldsymbol{x}})$ are equivalent up to a stabilizer multiplication.

The proof for Z-type operators follows by replacing the bitstring $\boldsymbol{x}$ by $\boldsymbol{z}$, S_X by S_Z and the column operator C_i with the row operator R_i. □

B.2 Reduction to 2-local operators ($\mathcal{S}_1 \to \mathcal{S}_2$)

Constructing 2-local operators is desirable since performing operations with higher-weight interactions is less practical. Here, we further constrain the search space to contain only bitstrings that parameterize operators that are 2-local. Lemma 7 and Lemma 8 provide a list of rules for constructing these bitstrings for Z- and X-type operators, respectively. Appendix B.3 provides an example of valid sets of $\hat{Z}$ and $\hat{X}$ logical operators for the $[[16, 6, 8, 2]]$ code constructed using these rules.

For the convenience of proofs in this section, we rewrite, in Eq. (39), the structure of a $m \times m$ A matrix from Eq. (12). Likewise, Corollary 1 explicitly lists out the Z gauge generators, which follows directly from Definition 1.

$$
A = \begin{array}{c|cccccccccc}
1 & 1 & 1 & & & & & & & & \\
 & 1 & & 1 & & & & & & & \\
 & \vdots & & & \ddots & & & & & & \\
2l & 1 & & & & 1 & & & & & \\
2l+1 & & 1 & & & & 1 & & & & \\
 & & & \ddots & & & & \ddots & & & \\
 & & & & 1 & & & & 1 & & \\
m-1 & & & & & 1 & & & & 1 & \\
m & & & & & & 1 & \cdots & 1 & 1 & \\
\hline
 & 1 & & & & m-2l+1 & & & & m &
\end{array}
\tag{39}
$$

Corollary 1. *The complete set of Z-type gauge generators of the gauge group $\mathcal{G}$ is:*

- *first column: $Z_{i,1}Z_{i+1,1}$ for $1 \leq i \leq 2l-1$*
- *columns 2 to $m-2l$: $Z_{j-1,j}Z_{j-1+2l,j}$ for $2 \leq j \leq m-2l$*
- *columns $m-2l+1$ to $m-1$: $Z_{j-1,j}Z_{m,j}$ for $m-2l+1 \leq j \leq m-1$*

Proof. Following Definition 1, we construct Z-type gauge generators by writing out pairs of Z_iZ_j in the same column of the A matrix in Eq. (39).

The first column has 1's in rows 1 to $2l$. Since $\mathcal{G}$ is closed under multiplication, we only need to consider consecutive pairs of $(i, i+1)$ in this column. In the rest of the columns $i \in [2, m-1]$, the gauge generators are pairs of two Z operators in the same columns.

Note that the gauge operator $Z_{m-1,m}Z_{m,m}$ in the last column is not a generator because it can be obtained from the multiplication of all the other generators and the Z-stabilizer. □

In Lemma 7, we show the 2-local reduction for $\hat{Z}$ operators, and later in Lemma 8, we generalize this result to $\hat{X}$ operators.

Lemma 7. *For a $[[4k+2l, 2k, g, 2]]$ trapezoid code, a 2-local $\hat{Z}$ operator can be constructed from a bitstring $\boldsymbol{z} = z_1 z_2 \dots z_m$ if and only if the bitstring has 1 entries according to one of the following rules and 0 entries in the remaining positions.*

(1) $\forall n \in [0, N]$*:* $z_{i+2nl} = 1$ *where* $i \in [1, m-1]$*,* $N \in [0, M]$ *and* $M = \lfloor \frac{m-1-i}{2l} \rfloor$

(2) $\forall n_{(i,j)} \in [0, N_{(i,j)}]$*:* $z_{i+2n_il} = z_{j+2n_jl} = 1$ *where* $i, j \in [1, 2l]$*,* $i < j$*,* $N_{(i,j)} \in [0, M_{(i,j)}]$*,* $M_{(i,j)} = \lfloor \frac{m-1-(i,j)}{2l} \rfloor$

Proof. ($\Rightarrow$) First, we show that if a $\boldsymbol{z}$ follows one of the rules, it parameterizes a 2-local $\hat{Z}$. We start by considering the special case of each rule, and then generalize it.

Consider the special case of Rule (1) when $N = 0$, a $\boldsymbol{z}$ has Hamming weight 1, hence, $\hat{Z} = \bar{Z}$ is 2-local by construction. For a general case with $N > 0$, $\forall n \in [1, N]$, multiplying by the gauges $Z_{i+2(n-1)l,i+1}Z_{i+2nl,i+1}$ removes the Z operators in the same column. The remaining operators are $Z_{i,1}$ and $Z_{i+2nl,i+2(n+1)l}$, which constructs a 2-local $\hat{Z}$.

Next, consider the special case of Rule (2) when $N_i = N_j = 0$, $\hat{Z}$ is 2-local after multiplying by the gauges $Z_{i,1}Z_{j,1}$. For a general case with $N_i > 0$, $\forall n_j \in [1, N_j]$, multiplying by the gauges $Z_{i+2(n_j-1)l,i+1}Z_{i+2n_jl,i+1}$ removes the Z operators in the columns $i+1$. Then, we can multiply Z gauges in every other $2l$ column according to n_j. The same cancellation method applies for $N_j > 0$, replacing the index $i \to j$, $j \to i$. The remaining operators are $Z_{i+2n_jl,i+2(n_j+1)l}$ and $Z_{j+2n_jl,j+2(n_j+1)l}$, which constructs a 2-local $\hat{Z}$.

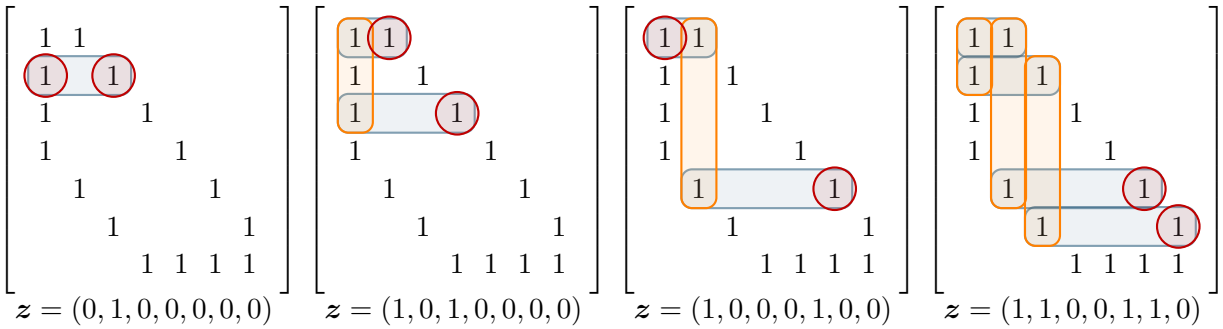


Figure 9: Bitstrings that parameterize 2-local $\hat{Z}$ operators according to the rules in Lemma 7. The top row represents special cases, while the bottom row represents general cases. The left column corresponds to Rule (1), and the right column corresponds to Rule (2). The blue horizontal boxes are $\bar{Z}$ operators parameterized by the bitstrings $\boldsymbol{z}$. The orange vertical boxes are Z-type gauge operators multiplied to $\bar{Z}$ to obtain a 2-local $\hat{Z}$. The red circles are the remaining Z physical operators of $\hat{Z}$.

($\Leftarrow$) We now show that if a $\hat{Z}$ is 2-local, it must be parameterized by a bitstring that follows one of the rules. We show this by considering all possible Z cancellations by multiplying $\bar{Z}$ with all possible combinations of Z-type gauges in Corollary 1.

Recall the reduced search space $\mathcal{S}_1 = \{\boldsymbol{z} | z_m = 0\}$. This means that $\{\boldsymbol{z} | z_m = 1\} \notin \mathcal{S}_1$, i.e., Z operators in the last row are not involved. Hence, we do not need to consider 2-local $\hat{Z}$ that comes from Z cancellations involving the third type of gauge operators in Corollary 1, since those gauges contain $Z_{m,j}$ which is in the last row of the A matrix.

By construction, the A matrix has two Z physical operators in each remaining row. This implies that a bitstring $\boldsymbol{z}$ of Hamming weight w parameterizes a $2w$-local $\bar{Z}$. To construct a 2-local $\hat{Z}$, we must multiply gauge operators with a total of $2(w-1)$ overlapping physical operations.

When $w = 1$, the gauge operator is not required. This falls into the special case of Rule (1). To use the first type of gauge operators in Corollary 1, one requires overlaps in multiples of $2l$, i.e., the $(i + 2nl)$'th rows, which is covered in the general case of Rule (1). To use the second type of gauge operators in Corollary 1, the overlap is constructed using the special case of Rule (2). When both types of gauge operators are used, one requires overlaps in multiples of $2l$, but starting with both rows i'th and j'th, hence, falls into the general case of Rule (2).

□

To illustrate the procedures in Lemma 7, Fig. 9 shows a diagram representing the special and general cases of both rules. The example is constructed for a $[[16, 6, 8, 2]]$ code ($m = 7, l = 2$). The special case of Rule (1) is 2-local by construction (top left), hence, does not require gauge cancellations. The special case of Rule (2) requires gauge cancellation in the first column (top right). The general case of each rule (bottom row) adopts the gauge cancellation from its special case and requires extra cancellations in each column with two Z operators.

Having established the rules to construct 2-local $\hat{Z}$ operators in Lemma 7, we move on to show how to apply this for $\hat{X}$ operators in Lemma 8.

Lemma 8. *For a $[[4k+2l, 2k, g, 2]]$ trapezoid code, a bitstring $\boldsymbol{x}$ parameterizes a 2-local logical operator $\hat{X}$ if and only if it is inverted of a bitstring in Lemma 7, where the i'th component of a length-m inverted bitstring $\boldsymbol{a}^{-1}$ is defined as $a_i^{-1} = a_{m-i}$, such that a_i is the i'th component of $\boldsymbol{a}$.*

Proof. Note that the A matrix is symmetric upon reflection about the main diagonal that runs from bottom left to top right. Perform the reflection about this diagonal and follow the proof in Lemma 7 on the rows and columns, which are now the inverted columns and rows, respectively, after reflection. It follows that $\hat{X}_i$ is parameterized by $\boldsymbol{x} = \boldsymbol{z}^{-1}$, when $\boldsymbol{z}$ parameterizes $\hat{Z}_i$ of a logical qubit i. □

We have shown that Lemma 7 and Lemma 8 have a complete set of bitstrings $\boldsymbol{z}$ and $\boldsymbol{x}$ that parameterize 2-local logical operators. The operator $\hat{L}_i \hat{L}_j$ where $L \in \{X, Z\}$ is parameterized by a

bitstring $\boldsymbol{a}_{i,j} = \boldsymbol{a}_i \oplus \boldsymbol{a}_j$ where $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ parameterize $\hat{L}_i$ and $\hat{L}_j$, respectively. This implies that the bitstring that parameterizes $\hat{Z}\hat{Z}$ and $\hat{X}\hat{X}$ must also follow the rules in Lemma 7 and Lemma 8, respectively.

B.3 Example

This section presents an example of how to construct 2-local $\hat{Z}$ and $\hat{X}$ of the $[[16, 6, 8, 2]]$ trapezoid code ($m = 7, k = 3, l = 2$). The A matrix of the code is given in Eq. (40). The row and column are indexed from 1 to m.

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \tag{40}$$

Using Lemma 7 and Lemma 8, a complete set of bitstrings $\boldsymbol{z}$ and $\boldsymbol{x}$ that parameterizes 2-local $\hat{Z}$ and $\hat{X}$ operators of the $[[16, 6, 8, 2]]$ code can be listed as in Table 5.

Z-type			
(1) Special	(2) Special	(1) General	(2) General
1 0 0 0 0 0 0	1 1 0 0 0 0 0	1 0 0 0 1 0 0	1 1 0 0 1 0 0
0 1 0 0 0 0 0	1 0 1 0 0 0 0	0 1 0 0 0 1 0	1 1 0 0 0 1 0
0 0 1 0 0 0 0	1 0 0 1 0 0 0		1 1 0 0 1 1 0
0 0 0 1 0 0 0	0 1 1 0 0 0 0		1 0 1 0 1 0 0
0 0 0 0 1 0 0	0 1 0 1 0 0 0		1 0 0 1 1 0 0
0 0 0 0 0 1 0	0 0 1 1 0 0 0		0 1 1 0 0 1 0
			0 1 0 1 0 1 0
X-type			
(1) Special	(2) Special	(1) General	(2) General
0 0 0 0 0 0 1	0 0 0 0 0 1 1	0 0 1 0 0 0 1	0 0 1 0 0 1 1
0 0 0 0 0 1 0	0 0 0 0 1 0 1	0 1 0 0 0 1 0	0 1 0 0 0 1 1
0 0 0 0 1 0 0	0 0 0 1 0 0 1		0 1 1 0 0 1 1
0 0 0 1 0 0 0	0 0 0 0 1 1 0		0 0 1 0 1 0 1
0 0 1 0 0 0 0	0 0 0 1 0 1 0		0 0 1 1 0 0 1
0 1 0 0 0 0 0	0 0 0 1 1 0 0		0 1 0 0 1 1 0
			0 1 0 1 0 1 0

Table 5: A complete set of bitstrings $\boldsymbol{z}$ and $\boldsymbol{x}$ that parameterizes 2-local $\hat{Z}$ and $\hat{X}$ operators of the $[[16, 6, 8, 2]]$ code. Orange-colored bitstrings are those chosen as logical operators in Eq. (19b). The column headers refer to the terminology used in Lemma 7.

Bitstrings listed in Table 5 are possible candidates for constructing six pairs of logical operators $(\hat{X}, \hat{Z})$ of the $[[16, 6, 8, 2]]$ trapezoid code, however, we need to make sure that the commutation relation $[\hat{X}_i, \hat{Z}_j] = \delta_{ij}$ is satisfied.

For dressed logical operators, we follow Proposition 2 and pick a set of bitstrings that satisfies $\boldsymbol{z}_j^T A \boldsymbol{x}_i + \bar{\boldsymbol{z}}_j^T A^T \bar{\boldsymbol{x}}_i = \delta_{ij}$. Our choice for the $[[16, 6, 8, 2]]$ code is given in Eq. (19b) and Eq. (20b), i.e., $\boldsymbol{Z}^T A \boldsymbol{X} + \bar{\boldsymbol{Z}}^T A^T \bar{\boldsymbol{X}} = I$, where I is a $(m-1) \times (m-1)$ identity matrix.

B.4 Counting the cardinality of $\mathcal{S}_2$

We have demonstrated that $\mathcal{S}_2$ exclusively contains bitstrings that parameterize 2-local operators. In this section, we determine the cardinality of this set and establish the relation that it depends on m but not on l. Specifically, trapezoid codes with A matrices of identical dimensions will have $\mathcal{S}_2$ sets of equal size.

Lemma 9. *Let $\mathcal{S}_2$ be the set of bitstrings that parameterize 2-local $\hat{Z}$ operators. For odd-m codes, the total number of such bitstrings is $|\mathcal{S}_2| = m(m-1)/2$, which is a function of m (or k) but not l.*

Proof. We prove by induction that a code corresponding to an $m \times m$ A matrix ($m = 2k+1$) has $m(m-1)/2$ bitstrings that parameterize 2-local $\hat{Z}$ operators. We start by showing the base case ($k = 1, m = 3; l = 1$), then assume that it is true for $k > 1$, and lastly, show that it is also true for $k' = k+1$, which corresponds to an $m' \times m'$ A' matrix, where $m' = m+2$. Since bitstrings for $\hat{X}$ operators are the inverted of the $\hat{Z}$ operators, this proof also applies for $\boldsymbol{x}$ without loss of generality.

The A matrix for the base case is given by

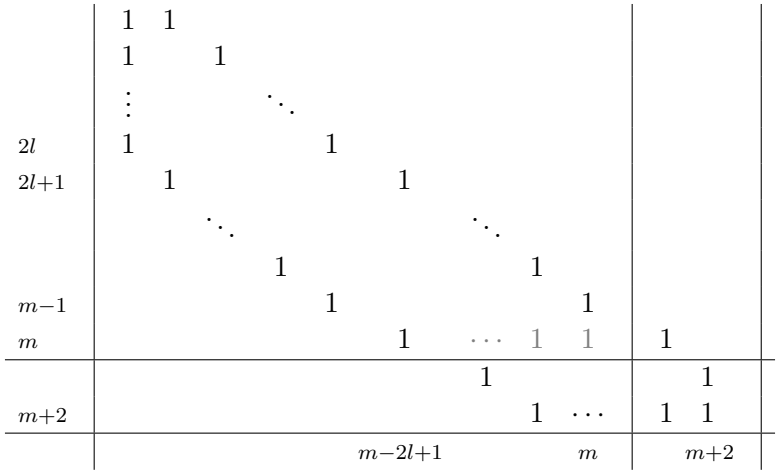
$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{41}$$

Using Lemma 7, we list bitstrings $\boldsymbol{z}$ that parametrize 2-local $\hat{Z}$ operators in Table 6. The total number of bitstrings is 3, which agrees with $m(m-1)/2 = 3$ when $m = 3$.

(1) Special	(2) Special	(1) General	(2) General
1 0 0	**1 1** 0		
0 **1** 0			

Table 6: A complete set of bitstrings $\boldsymbol{z}$ that parameterize 2-local $\hat{Z}$ operators for the $[[6, 2, 2, 2]]$ code.

Next, we assume that the total number of bitstrings is $m(m-1)/2$, where $m = 2k+1$, for the code corresponding to the $m \times m$ A matrix, drawn as the inner box in Eq. (42). Consider modifying the code parameter $k \to k' = k+1$, which corresponds to modifying the matrix $A \to A'$, drawn as the outer box in Eq. (42). The A' matrix is of size $m' \times m'$, where $m' = m+2$. The gray-colored 1-entries are not included in the A' matrix.



(42)

We now consider the addition of bitstrings $\boldsymbol{z}$ resulting from this modification, i.e., the additional bitstrings using Rule (1) and Rule (2) from Lemma 7.

Rule (1) contribution: The special case of Rule (1) contributes as two added rows, $m+1$ and $m+2$. The general case counts the number of Rule (1) bitstrings that have a 1-entry in the $(m-2l)$'th

or $(m+1-2l)$'th position. Only these bitstrings can have an extra 1-entry in the m'th or $(m+1)$'th position because positions of the 1-entries need to be separated by $2l$. The total number of Rule (1) bitstrings having their last 1-entry in the i'th row is $\lfloor\frac{i-1}{2l}\rfloor$. Hence, the total contribution from Rule (1) is $2+\lfloor\frac{m-1}{2l}\rfloor+\lfloor\frac{m}{2l}\rfloor$.

Rule (2) contribution: To extend Rule (2) bitstrings to $\boldsymbol{z}'$ of length $m+2$, the extension can only come from the original length-m $\boldsymbol{z}$ bitstring that has $(1,1)$, $(0,1)$ or $(1,0)$ in the $(m-2l, m+1-2l)$'th positions. This is because the positions of each 1-entry must be $2l$ apart for the gauge operators in the same column to cancel. Let $i_0 = m \mod 2l$ and $i_1 = m+1-2l \mod 2l$. It follows that only bitstrings with the first 1-entry in the i_0 and/or i_1 positions can have $(1,1)$, $(0,1)$ or $(1,0)$ in the $(m-2l, m+1-2l)$'th positions.

Consider a bitstring with a 1-entry in the i_0 position and has $(1,0)$ in the $(m-2l, m+1-2l)$'th positions. The m'th row adds another variation of bitstring by adding a 1-entry in the m'th position. Rule (2) bitstrings with a 1-entry in the i_0 position also has another 1-entry in the i position, where $i \in [1,2l]\setminus\{i_0\}$. Likewise, for bitstrings with a 1-entry in the i_1 position and $(0,1)$ in the $(m-2l, m+1-2l)$'th positions, the $(m+1)$'th row adds another variation of bitstring by adding a 1-entry in the $(m+1)$'th position. Counting the number of these bitstrings, we have $\sum_{i=i_0,i_1}\sum_{\substack{j=1\\ j\neq i}}^{2l} 1+\lfloor\frac{m-1-j}{2l}\rfloor$. Lastly, we count the contribution the bitstring with 1-entries in both i_0 and i_1 positions and $(1,1)$ in the $(m-2l, m+1-2l)$'th positions.

Total contribution: The total contribution from both Rule (1) and Rule (2) bitstrings is

$$\begin{aligned}
&\underbrace{2+\lfloor\frac{m-1}{2l}\rfloor+\lfloor\frac{m}{2l}\rfloor}_{\text{Rule (1)}}+\underbrace{1+\sum_{i=i_0,i_1}\sum_{\substack{j=1\\ j\neq i}}^{2l} 1+\lfloor\frac{m-1-j}{2l}\rfloor}_{\text{Rule (2)}}\\
&=1+\lfloor\frac{m-1}{2l}\rfloor+\lfloor\frac{m}{2l}\rfloor-\lfloor\frac{m-1-i_0}{2l}\rfloor-\lfloor\frac{m-1-i_1}{2l}\rfloor+2\left(\sum_{j=1}^{2l}1+\lfloor\frac{m-1-j}{2l}\rfloor\right)\\
&=1+2x-2(x-1)+2(m-1)=2m+1,
\end{aligned}\tag{43}$$

where $x=\lfloor\frac{m-1}{2l}\rfloor$.

To see this, let $\frac{m-1}{2l}=x+\frac{y}{2l}$, where $0\le y<2l$. Using this definition, we can write

$$\begin{aligned}
\lfloor\frac{m-1}{2l}\rfloor&=x+\lfloor\frac{y}{2l}\rfloor,\\
\lfloor\frac{m}{2l}\rfloor&=x+\lfloor\frac{y+1}{2l}\rfloor,\\
\lfloor\frac{m-1-i_0}{2l}\rfloor&=x-1+\lfloor\frac{1}{2l}\rfloor,\\
\lfloor\frac{m-1-i_1}{2l}\rfloor&=x-1+\lfloor\frac{2}{2l}\rfloor.
\end{aligned}\tag{44}$$

In the second line, note that since m is odd, $y\equiv m-1 \mod 2l$ must be even. It follows that $0\le y\le 2l-2$, which implies that $y+1<2l$. In the third and fourth lines, we have used the definition $i_0=m \mod 2l$, which implies $m-1-i_0\equiv -1 \mod 2l$ and likewise for $i_1=i_0+1$.

Similarly, $\sum_{j=1}^{2l}\lfloor\frac{m-1-j}{2l}\rfloor=xy+(2l-y)(x-1)=2xl-2l+y=m-1-2l$ because $\frac{m-1-y}{2l}=x$ by definition. Hence, $2(\sum_{j=1}^{2l}1+\lfloor\frac{m-1-j}{2l}\rfloor)=2(2l+m-1-2l)=2(m-1)$.

The total sum of bitstrings is $2m+1$. Hence, we have $m(m-1)/2+(2m+1)=(m+2)(m+1)/2=m'(m'-1)/2$.

□

C Penalty Hamiltonian

C.1 Construction of the dressed Hamiltonian

We follow the proof in Ref. [39] and apply it to our setting. The goal is to show that in the large penalty limit $H(t)$ [Eq. (24)] generates the desired computation, i.e., the same computation as generated by $H_S(t)$ in the absence of coupling to a bath.

Lemma 10. *[39, Lemma 1.1] Let $H_P = -\sum_{g_i \in \mathcal{G}'} g_i$, where $\mathcal{G}'$ is a set of X-type and Z-type gauge operators that generates the gauge group $\mathcal{G}$. Denote the projector to the ground subspace of H_P by Π_0. Then*

1. *Any ground state of H_P is stabilized by the stabilizers of the subsystem code.*
2. *$\Pi_0 g_i \Pi_0$ is proportional to Π_0, i.e., $\Pi_0 g_i \Pi_0 = \alpha_i \Pi_0$.*

Corollary 2. *[39, Eq. (6)] For $H_{SB} = \sum \sigma_\alpha \otimes B_\alpha$, where σ_α is a 1-local system operator, $\Pi_0 H_{SB} \Pi_0 = 0$.*

Proof. The first part of Lemma 10 guarantees that the ground subspace of the penalty Hamiltonian is in the codespace and hence can detect any 1-local error σ_α. The codespace projector can be written as $\Pi_0 = \prod_{i=1}^{n-k} \frac{1}{2}(I + S_i)$. For a stabilizer code, errors are detectable if and only if they anticommute with at least one stabilizer generator, e.g., S_j. Therefore, $(I+S_j)\sigma_\alpha(I+S_j) = (I+S_j)(I-S_j)\sigma_\alpha = 0$. Thus, $\Pi_0 \sigma_\alpha \Pi_0 = 0, \forall \alpha$, and $\Pi_0 H_{SB} \Pi_0 = 0$. □

Lemma 11. *[39, Eq. (8)] For $\bar{H}_S(t)$ and $\hat{H}_S(t)$ in Eq. (21) and Eq. (22b), respectively, $\Pi_0 \hat{H}_S(t) \Pi_0 = \Pi_0 \bar{H}_S(t)$.*

Proof. Using $\hat{H}_S(t)$ in the form of Eq. (22) and applying the projectors Π_0 on both sides, we can write:

$$\begin{aligned}
&\Pi_0 \hat{H}_S(t) \Pi_0 \\
&= \Pi_0 [\sum_i a_i \Pi_0 \bar{X}_i g_i^x \Pi_0 + \sum_i b_i \Pi_0 \bar{Z}_i g_i^z \Pi_0 + \sum_{(i,j) \in E_S^{XX}} c_{ij} \Pi_0 \bar{X}_i \bar{X}_j g_{ij}^x \Pi_0 + \sum_{(i,j) \in E_S^{ZZ}} d_{ij} \Pi_0 \bar{Z}_i \bar{Z}_j g_{ij}^z \Pi_0] \qquad (45a) \\
&= \Pi_0 [\sum_i a_i \bar{X}_i \Pi_0 g_i^x \Pi_0 + \sum_i b_i \bar{Z}_i \Pi_0 g_i^z \Pi_0 + \sum_{(i,j) \in E_S^{XX}} c_{ij} \bar{X}_i \bar{X}_j \Pi_0 g_{ij}^x \Pi_0 + \sum_{(i,j) \in E_S^{ZZ}} d_{ij} \bar{Z}_i \bar{Z}_j \Pi_0 g_{ij}^z \Pi_0], \qquad (45b)
\end{aligned}$$

where to go from the first to the second equality, we used the fact that by definition, bare logical operators leave the ground subspace of the penalty Hamiltonian invariant, so that $[\bar{X}_i, \Pi_0] = [\bar{Z}_i, \Pi_0] = 0$.

The situation differs slightly when we use dressed logical operators. The optimal set of logical operators we constructed in Section 4.3 are dressed logical operators $\hat{X}_i$ and $\hat{Z}_i$, which can be represented by products of bare logical operators and gauges g_i^x and g_i^z. Note that the gauges g_i^x and g_i^z might be products of 2-local gauge generators, which also belong to the gauge group $\mathcal{G}$. One special case is when $n = l$, as shown in [39]. They use the same logical operators, but they have $\hat{X}_i = \bar{X}_i$ and $\hat{Z}_i = \bar{Z}_i$ due to $n = l$. However, their $\hat{X}_i \hat{X}_j$ and $\hat{Z}_i \hat{Z}_j$ are dressed logical operators, which corresponds to the case where $\alpha_i^x = \alpha_i^z = 1$. The bare logical operator commutes with Π_0, so we arrive at the second line assuming that g_i^x is as discussed above (i.e., a product of 2-local gauge generators). The same argument applies to $\hat{Z}_i$.

Using the second part of Lemma 10, $\Pi_0 g_i^{\{x,z\}} \Pi_0 = \alpha_i^{\{x,z\}} \Pi_0$, we arrive at:

$$\begin{aligned}
\Pi_0 \hat{H}_S(t) \Pi_0 = \Pi_0 [&\sum_i \alpha_i^x a_i' \bar{X}_i + \sum_i \alpha_i^z b_i' \bar{Z}_i \\
&+ \sum_{(i,j) \in E_S^{XX}} \alpha_{ij}^x c_{ij}' \bar{X}_i \bar{X}_j g_{ij}^x + \sum_{(i,j) \in E_S^{ZZ}} \alpha_{ij}^z d_{ij}' \bar{Z}_i \bar{Z}_j g_{ij}^z], \qquad (46)
\end{aligned}$$

g^x	operators	index	g^z	operators	index
$g^x_{a,i}$	$X_{i,1}X_{i,i+1}$	$1 \le i \le 2l$	$g^z_{a,i}$	$Z_{m+i-2l,m_g+i}Z_{m,m_g+i}$	$0 \le i \le 2l-1$
$g^x_{b,i}$	$X_{i,i+1}X_{i,i+1+2l}$	$2l+1 \le i \le m-1$	$g^z_{b,i}$	$Z_{i,i+1}Z_{i+2l,i+1}$	$1 \le i \le m_g-2$
$g^x_{c,i}$	$X_{m,m_g}X_{m,m_g+i}$	$1 \le i \le 2l-1$	$g^z_{c,i}$	$Z_{m_l,1}Z_{i,1}$	$1 \le i \ne m_l \le 2l$

Table 7: A complete set of X-type and Z-type gauge operators, denoted by $g^{(x,z)}_{(a,b,c),i}$, and their corresponding physical operators on the A matrix. There is a total of $n_g = m-2+2l$ gauges for each X- and Z-type operator.

which is $\bar{H}_S(t)$ in Eq. (21) when the constants are set to $a'_i = a_i/\alpha^x_i, b'_i = b_i/\alpha^z_i, c'_{ij} = c_{ij}/\alpha^x_{ij}$ and $d'_{ij} = d_{ij}/\alpha^z_{ij}$. □

C.2 Simplification of the penalty gap calculation

Let each qubit be indexed according to its row and column. For an A matrix of size $m \times m$ parameterizing the code $[[4k+2l, 2k, g, 2]]$, the penalty Hamiltonian $H_P = -\sum_{g_i \in \mathcal{G}} g_i$ can be written explicitly as

$$H_P = -\sum_{i=1}^{n_g} g^x_i - \sum_{i=1}^{n_g} g^z_i, \tag{47}$$

where $n_g = m-2+2l$ is the number of gauge generators of each type. To see this, consider the A matrix and note that for X-type gauge generators, the first $m-1$ rows each contributes one g^x_i because there are two X operators in each of the first $m-1$ row. The last row has $2l$ operators, hence, it contributes $2l-1$ non-repeating gauge generators. The total is then $(m-1)+(2l-1) = m-2+2l$. A similar argument holds for the Z-type generators. The physical Z operators composing the Z-type gauge generators are given in Corollary 1.

In this subsection, we use the notation $g^{(x,z)}_{(a,b,c),i}$ to label the gauge generators according to their shape and position in the A matrix. An example of this indexing is drawn in Fig. 10 on the A matrix of a $[[16,6,8,2]]$ code. The definition is given in Table 7, where there are $2l$, $m-1-2l$, and $2l-1$ operators of type a, b, and c, respectively, which adds up to $m-2+2l = n_g$ gauges as required. Let $m_g = m+1-2l$ be the column index of the leftmost 1-entry of the last row of A.

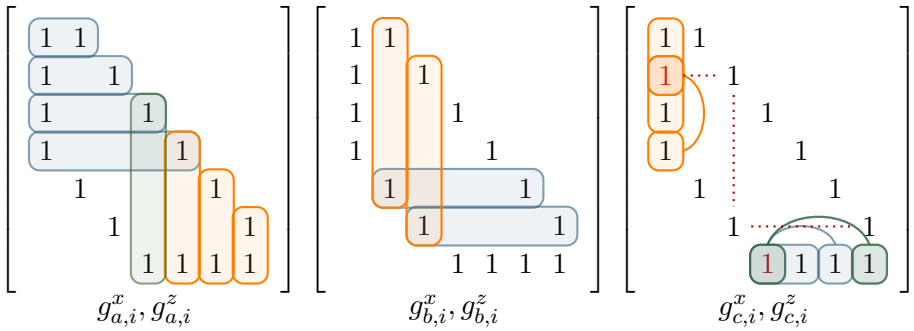


Figure 10: Gauge operators for evaluating the penalty Hamiltonian. Blue (orange) boxes are X-type (Z-type) operators. The green boxes are the two operators chosen to be written in terms of the product of other gauges in Eq. (49).

The most straightforward way to calculate the penalty gap of H_P in Eq. (47) is to evaluate H_P in the entire Hilbert space of $n = 4k+2l$ qubits, which scales as $\mathcal{O}(2^n)$ and quickly becomes intractable. However, it turns out that we can alleviate this problem by constructing a new, reduced operator space from the gauge operators, which removes one of the gauge operators. As a result, evaluating H_P in the correspondingly reduced Hilbert space quadratically reduces the complexity to $\mathcal{O}(2^{n/2})$.

Table 8 defines the reduced operators in terms of gauge operators $g^{(x,z)}_{(a,b,c),i}$. There are $n_g - 1$ reduced operators for each type (X and Z). To simplify the notation, we define the following functions. The

row index for $\hat{X}_i, \hat{Z}_i$	X-type gauges	Z-type gauges
$1 \le i \ne m_l \le 2l$	$g^x_{a,i}$	$g^z_{a,2l-1} g^z_{c,i} g^z_b(n_l)$
$i = m_l$	$g^x_{a,i}$	$g^z_{a,2l-1} g^z_b(n_l)$
$2l+1 \le i \le m-2$	$g^x_{b,i}$	$g^z_{a,2l-1} g^z_{c,\mathsf{M}(i)} g^z_b(n_l) \prod_{n=1}^{\mathsf{N}_\uparrow(i)} g^z_{b,i-2nl}$
$i = m-1$	$g^x_{b,i}$	$g^z_{a,2l-1}$
$m \le i \le n_g - 1$	$g^x_{c,i-m+1}$	$g^z_{a,i-m+1} g^z_{a,2l-1} g^z_{c,\mathsf{M}(i+1)} g^z_b(n_l) \prod_{n=2}^{\mathsf{N}_\uparrow(i+1)} g^z_{b,i+1-2nl}$

Table 8: The set of reduced operators in terms of gauge operators whose physical operators are in Table 7. Here $g^z_b(n_l) = \prod_{n=0}^{n_l} g^z_{b,m_l+2nl}$.

g^x	operators	index	g^z	operators	index
$g^x_{a,i}$	$\hat{X}_i$	$1 \le i \le 2l$	$g^z_{a,i}$	$\hat{Z}_{m-1+i}\hat{Z}_{m+i-2l}$	$1 \le i \le 2l-2$
				$\hat{Z}_{m-1}$	$i = 2l-1$
$g^x_{b,i}$	$\hat{X}_i$	$2l+1 \le i \le m-1$	$g^z_{b,i}$	$\hat{Z}_i\hat{Z}_{i+2l}$	$1 \le i \le m_g - 2$
$g^x_{c,i}$	$\hat{X}_{i+m-1}$	$1 \le i \le 2l-1$	$g^z_{c,i}$	$\hat{Z}_{m_l}\hat{Z}_i$	$1 \le i \ne m_l \le 2l$

Table 9: The gauge operators in terms of the new set of reduced operators.

row and column number is counted from 1 to m.

$$\mathsf{M}(x) = x \mod (2l), \tag{48a}$$

$$\mathsf{N}_\downarrow(x) = \lfloor \frac{m-1-x}{2l} \rfloor - 1, \tag{48b}$$

$$\mathsf{N}_\uparrow(x) = \lfloor \frac{x}{2l} \rfloor. \tag{48c}$$

We also define two variables that appear often: $m_l = \mathsf{M}(m-1)$ and $n_l = \mathsf{N}_\downarrow(m_l)$. Without loss of generality, we choose to define the X-type operators in a simpler format than the Z-type operators.

Using the reduced operators defined in Table 8, we can invert the relation and rewrite the gauge operators in terms of the reduced operators instead. Table 9 shows the gauge operators written in terms of the new reduced operators. There are n_g gauge operators but $n_g - 1$ reduced operators, hence, only $n_g - 1$ gauge operators are defined in terms of the reduced operators. The two remaining gauge operators can be written as the product of the stabilizer and other logical operators, so that the dimension of the Hilbert space can be reduced by half, i.e.,

$$g^x_{n_g} = S^X \prod_{i=1}^{n_g-1} \hat{X}_i \quad \text{and} \quad g^z_{n_g} = S^Z \hat{Z}_{m-2l} \prod_{i=n_g-1}^{m} \hat{Z}_i, \tag{49}$$

where these (arbitrary) chosen gauge operators are colored in green in Fig. 10.

Using the gauge operators in this form, we can rewrite the penalty Hamiltonian in Eq. (47) as

$$\begin{aligned} H_P = &- \sum_{i=1}^{n_g-1} \hat{X}_i - S^X \prod_{i=1}^{n_g-1} \hat{X}_i - S^Z \hat{Z}_{m-2l} \prod_{i=n_g-1}^{m} \hat{Z}_i \\ &- \hat{Z}_{m-1} - \sum_{\substack{i=1 \\ i \ne m_l}}^{2l} \hat{Z}_i \hat{Z}_{m_l} - \sum_{i=1}^{m_g-2} \hat{Z}_i \hat{Z}_{i+2l} - \sum_{i=1}^{2l-2} \hat{Z}_{m-1+i} \hat{Z}_{m+i-2l}, \end{aligned} \tag{50}$$

which has a quadratically smaller Hilbert space dimension compared to Eq. (47) and is the form we used to calculate the penalty gap in Fig. 3.

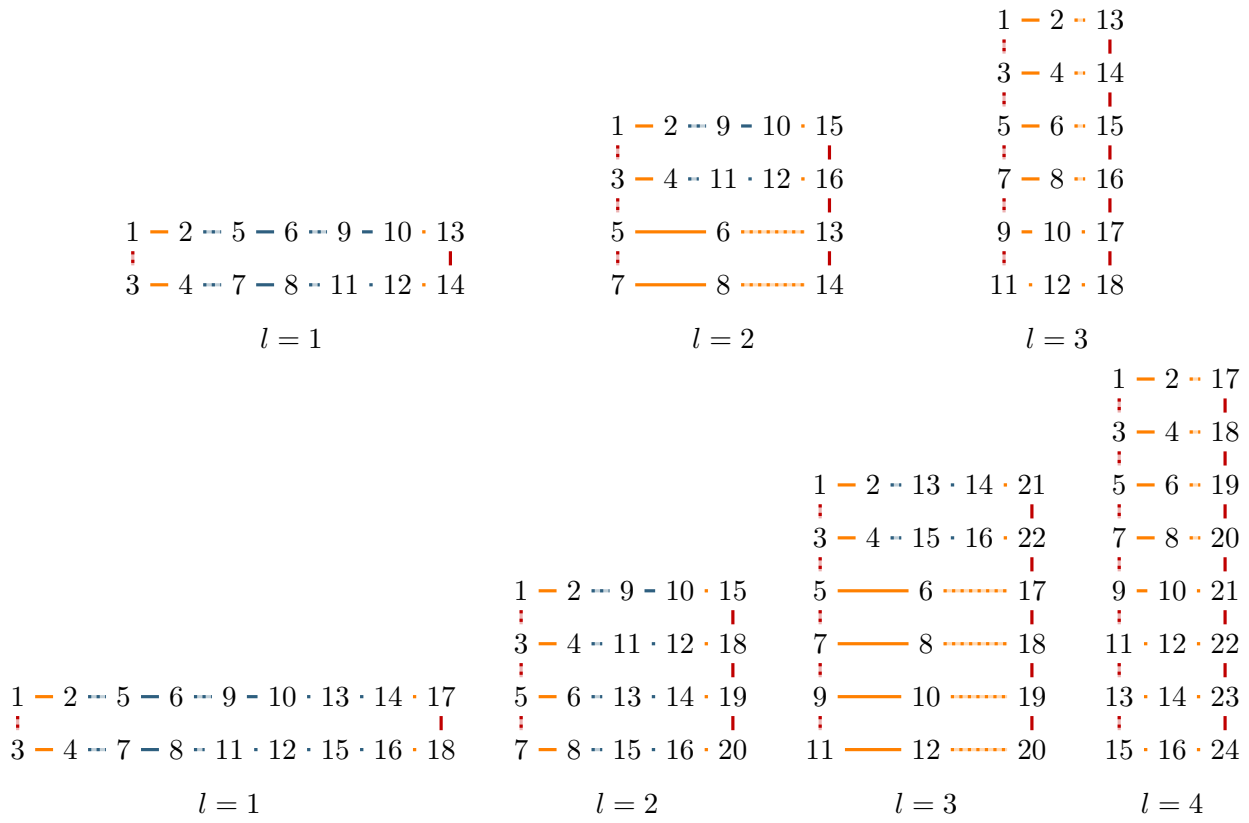


Figure 11: Gauge connectivity for $k = 3$ and $k = 4$ codes according to Table 7 and Fig. 10. Solid (dashed) lines are XX (ZZ) gauge operators. Orange, blue, and red colors denote $g_{a,i}^{(x,z)}$, $g_{b,i}^{(x,z)}$, and $g_{c,i}^{(x,z)}$ in the notation in Table 7, respectively. The first row of this figure is the same configuration as the second row in Fig. 2 with a different color code.

D Scaling of penalty Hamiltonian gap

To determine the scaling of the data presented in Fig. 3, we fit the results to a power-law model

$$y_{\text{power}} = a_{\text{power}} x^{-\nu_{\text{power}}}, \tag{51}$$

and to an exponential model

$$y_{\text{expo}} = a_{\text{expo}} e^{-\nu_{\text{expo}} x}. \tag{52}$$

Here y_{model} is the numerically computed penalty gap and x_{model} is the code parameter m (l) on the left (right) of Fig. 3. The fitting parameters are a_{model} and ν_{model}, where ν_{model} is the scaling parameter that we are most interested in.

D.1 Penalty gap as a function of m

Recall that m denotes the dimension of the matrix A. An A matrix of dimension $m \times m$ can support a family of k codes, where $k = \lfloor m/2 \rfloor$. These codes are indexed by l, ranging from $l = 1$ to $l = k$.

Table 10 presents the fitting parameters for both models, along with their associated standard errors as determined by the NonlinearModelFit function in Mathematica. Table 11 provides the Akaike Information Criterion (AIC) and R^2 values for both models. A lower AIC and higher R^2 values indicate a better fit to the data between the two models.

The results indicate a transition in behavior from $l = 2$ to $l = 3$, shifting from power-law decay to exponential decay of the penalty gap.

We also consider l as a function of k, i.e., the scaling of the penalty gap when $l = k - l'$ where we fix $l' = 0, 1, 2, \ldots$. The gap scaling in this case exhibits an exponential decay (rather than a

l	a_{power}	a_{power} (SE)	ν_{power}	ν_{power} (SE)	a_{expo}	a_{expo} (SE)	ν_{expo}	ν_{expo} (SE)
1	1.683	0.008407	1.032	0.003665	1.036	0.09426	0.1966	0.01909
2	1.107	0.04963	1.032	0.02311	0.3899	0.01495	0.1234	0.004746
3	1.419	0.1472	1.377	0.04799	0.256	0.005904	0.1371	0.002445
4	1.545	0.2443	1.728	0.06736	0.1322	0.003552	0.1478	0.002448
5	1.565	0.3098	2.098	0.0791	0.05851	0.001213	0.1576	0.001647
6	1.405	0.1859	2.432	0.05058	0.02522	0.0009054	0.1699	0.002589

Table 10: Fitting parameters a_{model} and ν_{model} for the penalty gap as a function of m for fixed l values from Fig. 3 (left) and their standard error (SE). Grey boxes denote the values of the fit parameters corresponding to the better fits, as determined by the AIC and R^2 values reported in Table 11.

l	AIC power	AIC expo	R^2 power	R^2 expo
1	-165.6	-50.15	0.9999	0.9631
2	-125.3	-110.9	0.9983	0.996
3	-122.1	-144.6	0.9967	0.9993
4	-131.3	-155	0.9969	0.9995
5	-140.1	-169.2	0.998	0.9999
6	-119.1	-123.7	0.9998	0.9999

Table 11: Akaike Information Criterion (AIC) and R^2 values for each l as a function of m reported to 4 significant figures. The better values are highlighted in gray, i.e., lower is better for AIC, and higher is better for R^2.

l'	a_{expo}	a_{expo} (SE)	ν_{expo}	ν_{expo} (SE)
0	2.053	0.07242	0.4482	0.009973
1	2.101	0.1999	0.3767	0.01671
2	2.886	0.4887	0.3623	0.02195
3	4.434	1.275	0.3573	0.02964
4	7.864	2.807	0.3628	0.03058
5	14.13	7.021	0.3651	0.03644

Table 12: Fitting parameters a_{expo} and ν_{expo} for the penalty gap as a function of m for different $l = k - l'$ values from Fig. 3 (middle) and their standard error (SE).

power law decay) as shown in Fig. 3 (middle). We provide the fitting parameters for $l = k - l'$ where $l' = 0, 1, \ldots, 5$ in Table 12.

D.2 Penalty gap as a function of l

The penalty gap can also be analyzed as a function of l for fixed values of m. Table 13 and Table 14 present fits analogous to those in the previous subsection, but are now fitted as a function of l.

In contrast to the gap scaling as a function of m, which exhibited a transition from power law to exponential, the decay as a function of l is exponential for all values of m, as seen in Table 14.

Fits for smaller values of m are not provided due to the limited number of data points. Specifically, an A matrix of size $m \times m$ can accommodate at most $\lfloor m/2 \rfloor$ codes within the family (e.g., for $m = 9$, there are $\lfloor 9/2 \rfloor = 4$ codes in the family). We only provide fits for m values with at least 5 data points.

E Mixed Integer Program for Optimization

The mixed integer program is inspired by Ref. [58], but we ignore the temporal sequencing of gates. We define the following binary variables:

$$w_{q,i} = 1 \text{ iff } q \in V \text{ resides at node } i \in V_{\text{m}}. \tag{53}$$

m	a_{power}	a_{power} (SE)	ν_{power}	ν_{power} (SE)	a_{expo}	a_{expo} (SE)	ν_{expo}	ν_{expo} (SE)
10	0.1645	0.01971	0.9866	0.2072	0.2675	0.02311	0.5036	0.04889
11	0.1494	0.01798	1.016	0.2136	0.2478	0.0218	0.522	0.05077
12	0.1375	0.01576	1.123	0.2089	0.2346	0.01798	0.5541	0.04504
13	0.1268	0.01444	1.147	0.2127	0.2196	0.01707	0.5691	0.04645
14	0.1179	0.01263	1.217	0.2048	0.2078	0.0146	0.5887	0.04264
15	0.1099	0.01163	1.244	0.2082	0.1974	0.01409	0.6071	0.04418
16	0.1027	0.0105	1.277	0.2079	0.1886	0.01297	0.6286	0.0435
17	0.0961	0.01009	1.263	0.2196	0.1798	0.0132	0.6437	0.0473
18	0.09057	0.009235	1.292	0.2195	0.1728	0.0123	0.6629	0.04664
19	0.08533	0.00907	1.26	0.235	0.1651	0.01334	0.6733	0.05379
20	0.08091	0.008369	1.289	0.2349	0.1596	0.01239	0.6921	0.0525

Table 13: Fitting parameters a_{model} and ν_{model} as a function of l for different m values and their standard error (SE).

m	AIC power	AIC expo	R^2 power	R^2 expo
10	-20.84	-29.47	0.9703	0.9947
11	-21.77	-30.41	0.9696	0.9946
12	-28.51	-40.06	0.9619	0.9944
13	-29.58	-41.06	0.9619	0.9944
14	-37.05	-51.19	0.957	0.9943
15	-38.22	-52.24	0.9575	0.9943
16	-39.68	-54.05	0.9598	0.9948
17	-33.94	-46	0.966	0.9955
18	-35	-47.34	0.9675	0.9958
19	-28.7	-38.32	0.9736	0.9961
20	-29.51	-39.44	0.9747	0.9965

Table 14: Akaike Information Criterion (AIC) and R^2 values for each m as a function of l. The exponential model is always a better fit.

We then have the following two constraints:

1. $\sum_{j\in V_{\text{m}}} w_{q,j} = 1 \ \forall q \in V$ to make sure each qubit is located at exactly one node,
2. $\sum_{q\in V} w_{q,j} = 1 \ \forall j \in V_{\text{m}}$ to make sure each node can host exactly one qubit.

We calculate the Manhattan distance table from every node to every node using shortest path algorithms and denote the table by M, where $M_{ij} = m(i,j)$. The cost function is:

$$\sum_{(p,q)\in E} \sum_{i,j\in V_{\text{m}}} w_{p,i} \times w_{q,j} \times M_{ij}. \tag{54}$$

Then we can use the following mixed integer programming to define the problem:

$$\min_{w} \sum_{(p,q)\in E} \sum_{i,j\in V_{\text{m}}} w_{p,i} \times w_{q,j} \times M_{ij} \tag{55a}$$

$$\text{s.t.} \sum_{j\in V_{\text{m}}} w_{q,j} = 1 \ \forall q \in V \tag{55b}$$

$$\sum_{q\in V} w_{q,j} = 1 \ \forall j \in V_{\text{m}} \tag{55c}$$

Since the variable w is binary, we can further reduce the problem to a linear program by defining

another binary variable:

$$y_{(p,q)(i,j)} = 1 \text{ iff } (p,q) \in E \text{ with } p \in V \text{ locate at } i \in V_{\mathrm{m}} \\ \text{and } q \in V \text{ locate at } j \in V_{\mathrm{m}} \tag{56}$$

Thus, we have the following constraints:

1. $\sum_{i,j \in V_{\mathrm{m}}} y_{(p,q)(i,j)} = 1 \ \forall (p,q) \in E$ to make sure each edge is implemented exactly once
2. $y_{(p,q)(i,j)} = w_{p,i} \times w_{q,j}$

Using McCormick's inequality, we can reduce the second constraint to:

$$\{w_{p,i}, w_{q,j}\} \geq y_{(p,q)(i,j)} \geq \{0, w_{p,i} + w_{q,j} - 1\} \tag{57}$$

Then the linear program is:

$$\min_{w} \sum_{(p,q) \in E} \sum_{i,j \in V_{\mathrm{m}}} y_{(p,q)(i,j)} \times M_{ij} \tag{58a}$$

$$\text{s.t.} \sum_{j \in V_{\mathrm{m}}} w_{q,j} = 1 \ \forall q \in V \tag{58b}$$

$$\sum_{q \in V} w_{q,j} = 1 \ \forall j \in V_{\mathrm{m}} \tag{58c}$$

$$\sum_{i,j \in V_{\mathrm{m}}} y_{(p,q)(i,j)} = 1 \ \forall (p,q) \in E \tag{58d}$$

$$y_{(p,q)(i,j)} \leq w_{(p,i)} \ \forall (p,q) \in E, \forall i,j \in V_{\mathrm{m}} \tag{58e}$$

$$y_{(p,q)(i,j)} \leq w_{(q,j)} \ \forall (p,q) \in E, \forall i,j \in V_{\mathrm{m}} \tag{58f}$$

$$y_{(p,q)(i,j)} \geq w_{(p,i)} + w_{(q,j)} - 1 \ \forall (p,q) \in E, \forall i,j \in V_{\mathrm{m}} \tag{58g}$$