Enumerating Pauli Stabilizer Codes through Ten Qubits

Sanchayan Dutta* and Greg Kuperberg†
University of California, Davis

We classify qubit stabilizer codes of length at most ten up to local Clifford transformations and qubit permutations. The classification contains 58 217 368 classes at length ten and includes every stabilizer rank, extending the classification through nine qubits by Cross and Vandeth (*arXiv:2501.17447*). We construct generator matrices one row at a time and discard partial rows that cannot have a lexicographically first completion. Graph canonical labeling identifies equivalent completed codes, and canonical reduced row echelon matrices give compact exact certificates. Of the ten-qubit classes, 56 743 701 are indecomposable. We check the Pauli enumeration using exact symmetry mass identities. We also adapt the search to Hermitian self-orthogonal $\mathbb{F}_4$-linear codes and binary self-orthogonal codes. The ten-qubit run took about eight hours in Python with pynauty.

1. INTRODUCTION

A stabilizer code is specified by a set of commuting Pauli checks on n qubits. If there are r independent checks, the code encodes $k = n - r$ qubits and is called an $[[n,k]]$ code. Local Clifford transformations change the Pauli operators used on each qubit, and coordinate permutations change the order of the qubits. We regard codes related by these transformations as equivalent.

For prescribed n and k, the classification problem is to find one representative of every equivalence class. We impose no restriction on minimum distance. Once the representatives are available, further properties such as indecomposability can be studied without repeating the classification.

Cross and Vandeth (2025) classified stabilizer codes through nine qubits under local Clifford and permutation equivalence [1]. Some important special cases have been classified at greater lengths. The $k = 0$ codes correspond to trace-Hermitian self-dual additive codes over $\mathbb{F}_4$, which Danielsen and Parker (2006) classified through length twelve [2]. We extend the full stabilizer classification to ten qubits, including all ranks.

The main difficulty is the number of descriptions of the same code. Changing a generating basis leaves the stabilizer space unchanged, while local Clifford transformations and qubit permutations give equivalent spaces. On phase-free Pauli labels, the coordinate group has order $6^n n!$, approximately 2.2×10^{14} at $n = 10$. It is therefore useful to recognize redundant descriptions before they become completed candidates for an equivalence test.

Our search builds a generator matrix one row at a time. We choose a definite target in each class, namely the lexicographically first matrix among all equivalent ordered bases. A partial row can be rejected as soon as its assigned entries show that every completion has a smaller equivalent description. The same comparisons also allow a second saving. If an alternative can no longer become smaller, we stop tracking it. Graph canonical labeling settles the equivalences left among completed codes.

We enumerate unpadded codes and recover the full table by restoring unused and fixed coordinates. The accompanying source package contains three related searches, for Pauli, Hermitian self-orthogonal $\mathbb{F}_4$-linear, and binary self-orthogonal codes. They share the row-extension and pruning framework but use different inner products, equivalence groups, and auxiliary graphs. Exact symmetry masses provide an independent check of the Pauli enumeration.

Let $A_{n,k}$ denote the number of classes of $[[n,k]]$ codes. The ten-qubit counts appear in Table I, with $r = 10 - k$ ranging from zero to ten. Their sum is $\sum_{k=0}^{10} A_{10,k} = 58\,217\,368$. Together with the shorter lengths, this gives 58 971 374 classes for $1 \le n \le 10$, compared with 754 006 through nine qubits. Of the ten-qubit classes, 56 743 701 are indecomposable. Their joint weight statistics are given in Section 9.

2. CODE EQUIVALENCE

2.1. Phase-free Pauli spaces

We work with Pauli words modulo phases. Each coordinate has four possible symbols, and we identify them with binary pairs by assigning $(0,0)$, $(1,0)$, $(0,1)$, and $(1,1)$ to I, X, Z, and Y, respectively. Thus the phase-free Pauli group on n qubits is $\mathbb{F}_2^{2n}$. We write a word as (x,z) with $x,z \in \mathbb{F}_2^n$, and multiplication modulo phase becomes vector addition.

The commutation relations are recorded by the symplectic form

$$\langle (x,z),(x',z') \rangle_s = x \cdot z' + z \cdot x'. \tag{2.1}$$

Two Pauli operators commute exactly when this expression is zero. A stabilizer space is therefore a subspace $C \le \mathbb{F}_2^{2n}$ on which the form vanishes for every pair of words. Such a subspace is called totally isotropic. Its dimension r is at most n, and it defines an $[[n, n-r]]$ stabilizer code [3].

A local Clifford transformation fixes I and permutes the three nonidentity symbols. Every permutation of X, Z, Y occurs, so the induced group on one phase-free coordinate is $\mathrm{Sp}(2,2) \cong S_3$. We allow these transformations independently at each coordinate, together with arbitrary coordinate permutations. The resulting group is

$$G_n = S_3^n \rtimes S_n, \qquad |G_n| = 6^n n!. \tag{2.2}$$

* dutta@ucdavis.edu
† greg@math.ucdavis.edu

Two stabilizer spaces are equivalent when they lie in the same G_n-orbit. We write $\mathrm{Aut}(C)$ for the subgroup that maps C onto itself. An automorphism preserves the space and need not fix its chosen generators.

There is also freedom in the choice of those generators. Applying an invertible binary row operation to a generator matrix changes the basis but not the space C. Thus, when comparing generator matrices, we must account for both row operations and the coordinate transformations in G_n.

2.2. Removing trivial coordinates

Some coordinates can be separated from the code without any search. If every word of C has I at a coordinate, that coordinate is *unused*. Removing it leaves the stabilizer rank unchanged, so an $[[n,k]]$ code becomes an $[[n-1,k-1]]$ code. Conversely, adjoining an unused coordinate adds one logical qubit.

A different situation occurs when C contains a weight-one word supported on a coordinate. We call the coordinate *fixed*. By a local Clifford transformation we may take the nonidentity symbol of this word to be X. Since the word is I everywhere else, commutation forces every word of C to have I or X at that coordinate. We take the weight-one word as one generator and add it to each other generator with X there. All the other generators then have I at the coordinate. This splits off a one-qubit stabilizer state and leaves an $[[n-1,k]]$ code.

The weight-one word need not be among the generators initially chosen. For example, XXI and XXX generate a space containing their sum IIX, which fixes the third coordinate. After splitting off this coordinate, the remaining code is generated by XX. Thus the removal concerns the entire stabilizer space, not just its displayed rows.

We call a code *unpadded* if it has full support and contains no weight-one stabilizer word. Removing all unused and fixed coordinates leaves such a code. Clearing a fixed coordinate does not alter the entries at other coordinates. In particular, a weight-one word in the remaining space would already have been a word of C. No new unused or fixed coordinates appear during removal. Since both sets of positions are preserved by equivalence, the remaining unpadded class is uniquely determined.

Let $U_{n,k}$ be the number of unpadded classes, with $U_{0,0}=1$ and zero for invalid parameters. Suppose an $[[n,k]]$ code has u unused coordinates and p fixed coordinates. Removing them leaves an unpadded code with parameters $[[n-u-p,k-u]]$. Conversely, adjoining u unused coordinates and p fixed coordinates gives one class for each unpadded class. The positions can be permuted, and the nonidentity symbols at fixed coordinates can be changed by local Clifford transformations. Hence, with $r=n-k$,

$$A_{n,k}=\sum_{u=0}^{k}\sum_{p=0}^{r}U_{n-u-p,k-u}. \tag{2.3}$$

We can also recover the unpadded counts from the full table. Removing one unused coordinate gives a bijection between classes having at least one such coordinate and all $[[n-1,k-1]]$ classes. The corresponding count for a fixed coordinate is $A_{n-1,k}$, and classes with both types are counted by $A_{n-2,k-1}$. Inclusion-exclusion therefore gives

$$\begin{aligned}U_{n,k}=A_{n,k}-A_{n-1,k-1}-A_{n-1,k}\\ +A_{n-2,k-1},\end{aligned} \tag{2.4}$$

where $A_{n,k}=0$ for invalid parameters. Thus either table determines the other, and the search need only construct unpadded codes.

TABLE I. Equivalence classes at length ten, with $r=10-k$ and unrestricted minimum distance. The unpadded counts follow from (2.4).

k	r	All codes	Unpadded
10	0	1	0
9	1	10	1
8	2	136	36
7	3	2,834	1,462
6	4	87,636	68,461
5	5	2,197,201	2,036,061
4	6	18,326,571	17,837,766
3	7	29,545,493	29,020,895
2	8	7,708,447	7,509,651
1	9	345,049	324,428
0	10	3,990	3,315

3. LEX-FIRST GENERATION AND THE ACTIVATION STAIRCASE

3.1. A canonical target for generation

A recursive construction produces many matrices for the same code. To decide which ones can be discarded, we first specify the representative we want to keep. We order the Pauli symbols by

$$I<X<Z<Y, \qquad I=0,\ X=1,\ Z=2,\ Y=3. \tag{3.1}$$

We then compare generator matrices by reading each row from left to right and the rows from top to bottom. Among all ordered bases in all equivalent coordinate systems, there is a least matrix. We call it the *lexicographically first matrix* of the class.

This lexicographic order has a useful hereditary property. A smaller equivalent prefix would already make the whole matrix smaller, regardless of the last row.

Lemma 3.1 (Canonical prefixes)**.** *For $r\geq 1$, the first $r-1$ rows of a lexicographically first rank-r matrix are lexicographically first for their row space. The same holds after deleting columns that vanish in all $r-1$ rows.*

Proof. Suppose the prefix had a smaller equivalent representative. Apply the same coordinate transformation to the last row and append it to that representative. The rows remain independent, so this gives a smaller matrix for an equivalent rank-r code, a contradiction.

All zero columns of the prefix occur first. Otherwise, moving one to the left would decrease the prefix. If deleting these columns gave a matrix with a smaller equivalent representative, we could restore the zero columns on the left and again obtain a smaller prefix. □

We may therefore start from the least representatives at rank $r-1$ and adjoin a row. After identifying equivalent completed children, we retain the least matrix in each class. This does not require a full minimality test at every intermediate node. It is enough that our partial tests never discard a prefix of the least matrix. We develop these tests in Section 4.

3.2. The two types of Pauli extension

Deleting the last row may leave some coordinates unused. This is why a parent in the recursion can have shorter length than its child. The second part of the lemma allows us to delete those unused coordinates and work with the shorter parent.

There are two possibilities at the leftmost nonzero column of a lexicographically first matrix. Rows with I come first. The projection of the row space onto this coordinate has binary dimension one or two, so row operations and a local Clifford transformation reduce the remaining entries to a single X or the pair X,Z. Deleting the last row either makes the coordinate unused or leaves one Pauli direction there.

In the first case, take a parent of length $m < n$ and pad it on the left with $n-m$ identity columns. The new row must be nonzero at every added coordinate, since the child has full support. The parent is zero there, so independent local Clifford transformations make all these new entries X without changing it. The new row has the form $X^{n-m}v$, where v has length m. We call this an X *extension*.

For example, padding the seed XX gives the row IXX. Adjoining XZZ produces a commuting pair, since the two anticommutations at the last two coordinates cancel. We return to this parent when discussing orbit-minimum tests.

In the second case the parent already has length n. Its last row begins with a block X^b, and all earlier rows have I throughout this block. The new row must begin with Z^b. To see this, suppose one of its entries in the block were I or X. Move that column to the front of the block, which leaves the parent unchanged. If the new entry is X, add the last parent row to make it I. Exchanging these two rows now gives a smaller matrix. An entry Y is also excluded, since a local Clifford transformation fixing X lowers it to Z while preserving the parent. This is a Z *extension*.

The two extensions activate a new block or supply a second direction on a block already active. This gives the staircase form used by the two loops in `paulisearch.py`. The rank-one seeds are X^n for $n \geq 2$. Length one is omitted because its only nonzero stabilizer has weight one. At later ranks, a subspace of an unpadded code cannot contain a weight-one word, so deleting its unused coordinates again gives an unpadded parent.

For binary and $\mathbb{F}_4$-linear codes, the projection onto any single coordinate has dimension at most one over the underlying field. Row operations therefore allow us to make the first active column nonzero only in the last row of a lexicographically first matrix. Deleting that row makes the coordinate unused. These searches therefore need only parents of shorter length, with no second completion step.

3.3. Pivot constraints and early orthogonality

Before testing coordinate transformations, we can remove some freedom to add parent rows to the new row. In the implementation, each Pauli symbol is stored with its X bit low and its Z bit high, and the leftmost symbol of a row is most significant. Numerical order then agrees with (3.1), and addition of phase-free words is bitwise exclusive-or.

Let a parent pivot be the most significant nonzero bit of a parent row. If the new row has a 1 there, adding the parent row changes it to 0 without changing any more significant bit. This makes the new row smaller. We therefore set every parent pivot bit to zero. The mask `pmask` excludes these bits, while the extension rule fixes the leading X or Z block.

The remaining entries must make the new row commute with the parent. For each parent row, exchanging its X and Z bits gives the coefficient vector of one binary linear equation for the new row. We eliminate these equations from the least significant end, whereas entries of the new row are assigned from the most significant end. An equation can be tested as soon as all its bits are known, often before the row is complete. If it fails, no choice of later entries can repair it, and we discard the branch.

There is no separate self-commutation test, since $\langle v, v\rangle_s = 0$ for every Pauli row by alternation of the symplectic form. For the binary and $\mathbb{F}_4$-linear searches, the new row's product with itself under the dot product or Hermitian form must also be checked. The distinction will matter when we describe these searches.

4. OUTER AND INNER ORBIT-MINIMUM PRUNING

The staircase fixes the leading entries of a new row, but the remaining entries can still give many equivalent matrices. Moreover, adding a row creates new codewords that may improve an earlier row of the matrix. The parent was chosen to be least among descriptions of its own space. The larger space gives us further choices of generators. We therefore test every row position as the new row is constructed.

To make these tests inexpensive, we use transformations that keep the preceding rows fixed. Their action can be found directly from the entries already present. We first describe these transformations, then the words to which we apply them, and finally the bounds that allow a test before the new row is complete.

4.1. The pointwise prefix stabilizer

For a test at row i, we require every preceding row to remain unchanged. A smaller entry at the first differing position of row i will then make the whole matrix smaller, regardless of the rows below it. Columns with identical entries in the preceding rows can still be permuted. In a lexicographically first matrix these columns occur in contiguous blocks.

Local Clifford transformations give some further freedom within a column. If only I occurs above row i, all permutations of X,Z,Y preserve the preceding rows. If X is the only nonidentity symbol above, we may still interchange Z and Y. Once two independent Pauli symbols have appeared, no nontrivial relabeling fixes them both. The smallest obtainable symbols are therefore

	I	X	Z	Y
zero above	I	X	X	X
only X above	I	X	Z	Z
two directions above	I	X	Z	Y

(4.1)

We minimize each symbol by this table and sort the results within each block of identical columns. The Pauli program precomputes these choices in `minperms`. The binary and $\mathbb{F}_4$ programs use bounded caches for their corresponding computations.

Return to the parent with rows IXX and XZZ from the X extension example. Adjoin ZIX, which commutes with both rows and satisfies the Z extension rule. Although it is larger than the last parent row, it can improve an earlier position. At the second row position, we may interchange X and Z on the first qubit without changing the first row IXX. This sends the new word to XIX and the old second row to ZZZ. Reordering the last two rows gives the equivalent matrix

$$\begin{pmatrix} I & X & X \\ X & I & X \\ Z & Z & Z \end{pmatrix}.$$

Its second row is smaller, since $XIX < XZZ$. Thus a new row can show that an earlier row was not the least choice for the child, even though the parent itself was already chosen to be least.

These tests use transformations that fix the preceding rows individually. They need not include every automorphism of the parent code. For rejection, it is enough to exhibit one smaller equivalent matrix. If these transformations find none, we continue the search and test equivalence of the completed codes later.

4.2. Which words are compared

Let C_0 be the parent space and v the new row. Every word of the child that is not in the parent belongs to the coset $v+C_0$. It is useful to consider all these words, since adding parent generators to v can expose an improvement that is not apparent from v alone.

At each row position i, we take every word in $v+C_0$ and minimize it under the pointwise stabilizer of the preceding rows. Each word of the coset lies outside C_0, so it is independent of those preceding rows. We may put it in position i and complete a basis of the child code. After applying the coordinate transformation, the earlier rows are still unchanged. Hence a minimized word smaller than row i proves that the matrix cannot be lexicographically first.

For a rank-r Pauli extension, the coset contains 2^{r-1} words. At rank seven, for example, there are sixty-four words to compare at each row position. The binary search uses the same coset. For an $\mathbb{F}_4$-linear extension, the words outside the parent instead have the form

$$av+w, \qquad a \in \mathbb{F}_4^*, \quad w \in C_0, \tag{4.2}$$

where the nonzero scalars allow the different choices of a new $\mathbb{F}_4$ generator.

4.3. Bounds from a partial interval

The preceding comparisons can often be settled without finishing the row. We process the blocks of identical columns as intervals, from left to right. A comparison reaches the next interval only when the minimized competitor agrees with the comparison row on every completed interval. A smaller completed prefix rejects the branch. A larger one shows that this particular competitor cannot help, so we remove that comparison.

Now fix a competitor and an interval of b columns. Suppose its first s symbols are determined by the assigned part of the new row. Normalize and sort these symbols to obtain a word a, interpreted as an integer in the appropriate base. When more symbols are inserted and the interval is sorted again, its first s entries can only decrease. Thus a is an upper bound on the first s entries of the completed interval.

For example, suppose an interval has four columns and the two known normalized symbols are X and Z. Whatever the other symbols are, the first two entries of the final sorted interval are no larger than XZ. If the corresponding comparison interval begins with XY, the competitor is smaller for every completion. We can reject the branch immediately. In general, this is the outer test. It compares a with the corresponding s symbols of the comparison row.

We also have a lower bound on the whole interval. Fill its $b-s$ unknown positions with zeros and place the sorted word a after them. Every completion is at least this large, since zero is the smallest symbol. If the lower bound already exceeds the comparison interval, this competitor will never become smaller and can be dropped. In the four-column example, the lower bound is $IIXZ$. Against the interval $IIXX$, it proves that this comparison can be removed. Other competitors and completions of the new row remain under consideration. This is the inner test.

The comparison row may itself be incomplete when we test the last row. We then fill the unknown positions of its current interval with the largest alphabet symbol. This gives an upper bound on the comparison interval. If the competitor's lower

bound exceeds it, the competitor cannot become smaller under any completion of either interval and can again be dropped. In `guesspauli`, the upper-bound test rejects branches, while the lower-bound test removes individual comparisons.

Proposition 4.1 (Safety of the two pruning levels)**.** *The outer and inner pruning tests do not remove any lexicographically first matrix.*

Proof. An outer rejection establishes a smaller allowed competitor for every completion of the partial row. None of these completions can be lexicographically first. An inner removal establishes that the competitor exceeds the comparison row, or an upper bound on that row if it is incomplete. The competitor cannot become smaller later, so removing it does not affect the outer test. □

We carry the surviving comparisons through the recursion. When a new symbol is assigned, we update their partial words and apply the bounds again. In this way, a comparison that has already been settled does not need to be repeated at every subsequent node.

4.4. Completeness and residual duplicates

Theorem 4.2 (Correctness of the generation scheme)**.** *Assume that the interval minima and canonical equivalence tests are exact. The staircase search, using the pivot, orthogonality, and orbit-minimum rules described above, enumerates every positive-length unpadded Pauli equivalence class. Retaining the least completed matrix for each canonical key gives one representative per class.*

Proof. We induct on the rank. Every unpadded rank-one code is equivalent to $\mathrm{span}(X^n)$ for $n \geq 2$, so the seeds give all such classes. Let M be the lexicographically first matrix of an unpadded code of rank r. Delete its last row and then the columns that become unused. The remaining space has no weight-one words and is an unpadded parent of rank $r-1$. By the canonical-prefix lemma and induction, its least matrix is already stored.

The deleted row of M gives an X or Z extension of this parent. It satisfies the pivot restrictions and orthogonality equations, since M is already a valid stabilizer matrix in least form. An outer test cannot reject any prefix of this row, since that would give a smaller equivalent matrix. The inner tests only remove comparisons. Thus the search reaches M.

Conversely, the activation rules make the new row independent of the parent and give full support. The orthogonality tests ensure that the generators commute. If the resulting space contained a weight-one word, that word would lie in $v + C_0$, since the parent has none. At the first row position, local Clifford transformations and qubit permutations bring it to $I^{n-1}X$. This is smaller than the first parent row, which has weight at least two. The completed test would therefore reject the code. The surviving codes are unpadded. Finally, canonical keys identify equivalent completed matrices. Retaining the least matrix for each key leaves one representative per class. □

5. EXACT CERTIFICATES FROM GRAPH CANONICALIZATION

The pruning tests discard a matrix when they find a smaller equivalent description, but they need not settle all equivalences. Completed matrices can therefore still describe the same code class. To identify these, we transform code equivalence into graph isomorphism and use nauty to compute a canonical labeling [4]. The graph must be determined by the space, so that changing generators does not change it.

5.1. The Pauli graph

On one qubit, a local Clifford transformation permutes X, Z, Y arbitrarily. A triangle has exactly these vertex permutations, so we draw one for each qubit and label its vertices X, Z, Y. Exchanging entire triangles represents a qubit permutation.

We then add a vertex for every word of C, including zero. At each coordinate, connect the word vertex to the triangle vertex labeled by its symbol. Add no edge for I. Thus the neighbors of a word vertex record all its nonidentity entries. Using all words makes the construction independent of the chosen basis. A rank-r code has 2^r words, so the graph has

$$3n + 2^r \tag{5.1}$$

vertices, or 1 054 when $n = r = 10$.

Proposition 5.1. *Two Pauli spaces are G_n-equivalent if and only if their graphs are isomorphic. Moreover, the automorphism group of the graph is naturally isomorphic to* $\mathrm{Aut}(C)$.

Proof. A code equivalence permutes the coordinate triangles and the word vertices, giving a graph isomorphism. For the converse, observe that the coordinate triangles are the only triangles in the graph. Word vertices are pairwise nonadjacent, and each is adjacent to at most one vertex of any coordinate triangle. Thus a graph isomorphism must permute the coordinate triangles and their vertices, giving an element of G_n. The neighborhood of each word vertex determines its Pauli word, so this element maps one code onto the other.

The same argument applies to automorphisms. Distinct words have distinct neighborhoods, so an automorphism is determined by its action on the coordinate triangles. This identifies the graph automorphism group with $\mathrm{Aut}(C)$. □

5.2. Compact canonical RREF

A canonical graph labeling also gives an ordering of the qubits and their Pauli symbols. Order the coordinate triangles by their smallest canonical vertex labels. Within each triangle, assign X, Z, Y in increasing label order. Apply these coordinate transformations to the generator matrix.

The coordinates are now in canonical form, but the matrix still depends on the chosen generators. We remove this dependence by computing its binary reduced row echelon form

(RREF) in the fixed $[X \mid Z]$ column order. Packing its rows into a single integer gives the canonical key. Reducing the original matrix alone would remove the basis choice but would not identify different equivalent coordinate arrangements. The graph labeling supplies that part of the construction.

The key represents an actual stabilizer space rather than a hash of the graph. Equivalent codes give the same key regardless of their original generators or qubit ordering. Conversely, if two keys agree, their transformed row spaces are identical and the original codes are equivalent.

At rank r, the key requires at most $2nr$ bits. For $n = r = 10$, this is 200 bits, or 25 bytes. Python integers and dictionary entries have additional overhead, and we also store the least generator matrix found for each class. Even so, the packed RREF is much smaller than storing a canonical adjacency matrix with vertices for all 2^r words. The dictionary uses the RREF as its key and the least matrix as its value. The key identifies equivalent codes, while the stored matrix is extended at the next rank.

5.3. Invariant spanning subsets

For the field-linear searches, it is enough to use a selected spanning set of words in the graph. The selection must be preserved by equivalence. Otherwise, two equivalent codes could give graphs with different selected words. In particular, an arbitrary basis does not suffice, since different bases can have different weights and give different graphs.

We sort the nonzero words by Hamming weight and take complete weight layers, beginning with the smallest, until the selected words span the code. The allowed field-linear coordinate transformations preserve both weights and linear spans, so they preserve this choice. If two graphs are isomorphic with the prescribed vertex colors preserved, the induced coordinate transformation maps the selected words of one code onto those of the other. Since these words span, it maps the codes themselves.

For an $\mathbb{F}_4$-linear code, each selected weight layer contains all nonzero scalar multiples of its words. Consequently, the $\mathbb{F}_2$-span of the selected words equals their $\mathbb{F}_4$-span, viewed as subsets of $\mathbb{F}_4^n$. We use the same words in the $\mathbb{F}_4$ program when testing local Clifford and qubit permutation equivalence.

6. THE $\mathbb{F}_4$-LINEAR SEARCH

6.1. Algebra and extension conditions

The search also applies to stabilizer spaces with additional linear structure. We use $\mathbb{F}_4 = \{0, 1, \omega, \omega^2\}$, where $\omega^2 = \omega + 1$, identified with I, X, Z, Y in that order. An s-dimensional subspace of $\mathbb{F}_4^n$ is a rank-s code. Each field generator gives two binary Pauli generators, so its binary rank is $2s$. Working with an $\mathbb{F}_4$ basis therefore uses fewer generators, but we must impose commutation on all their field multiples.

We require $H(u,v) = 0$ for all codewords, where $H(u,v) = \sum_{j=1}^{n} u_j v_j^2$ is the Hermitian form. Such a code has $s \le n/2$ and defines an $[[n, n-2s]]$ stabilizer code [3]. The trace $\mathrm{Tr}(a) = a + a^2 \in \mathbb{F}_2$ relates the Hermitian condition to Pauli commutation.

$$\langle u, v \rangle_s = \mathrm{Tr}(H(u,v)). \tag{6.2}$$

A single commutation test gives only a trace condition. For each parent generator u, we therefore check that the new row v commutes with both u and ωu. Since 1 and ω form an $\mathbb{F}_2$-basis of $\mathbb{F}_4$, and the trace pairing is nondegenerate, the two tests together imply $H(u,v) = 0$. Neither test alone suffices.

There is one more pair to check. The two binary generators v and ωv must commute with each other. A Pauli row commutes with itself, but this does not settle the relation between these two rows. The Hermitian form is not alternating, and its self-product gives the condition

$$H(v,v) = \mathrm{wt}(v) \pmod 2 = 0. \tag{6.3}$$

Each nonzero coordinate contributes 1 to this product. The program checks the condition as $\langle v, \omega v \rangle_s = 0$. The two new binary generators then commute with each other and with the parent.

At rank one, coordinate scalings make every nonzero entry 1. The full-support seed is therefore the all-ones word, and the self-product condition permits it exactly at even lengths. At higher ranks we extend full-support parents of shorter length. We scale newly activated entries to 1 and clear pivot entries by $\mathbb{F}_4$ row operations. Orthogonality is checked while the new row is filled. The interval pruning methods of the Pauli search apply with the appropriate $\mathbb{F}_4$ equivalence transformations.

6.2. Monomial and Pauli equivalence

An $\mathbb{F}_4$-linear code can be classified as a code over $\mathbb{F}_4$ or through its underlying binary Pauli space. These give different equivalence questions. We first use monomial equivalence, allowing coordinate permutations and nonzero field scalings. The group is

$$M_n = (\mathbb{F}_4^*)^n \rtimes S_n, \qquad |M_n| = 3^n n!. \tag{6.4}$$

Field conjugation is excluded from this group, so conjugate codes need not be monomially equivalent.

The Pauli triangle allows all six permutations of its ports. For monomial equivalence we need only the three field scalings. We therefore use a nine-cycle at each coordinate, with three vertex colors repeated in a fixed order (Figure 1). Every third vertex is a Pauli port. The coloring excludes reflections and leaves exactly three rotations, corresponding to nonzero field scalings. Codeword vertices have a fourth color and attach to the appropriate ports.

We can also compare the underlying stabilizer spaces under the larger group G_n of local Clifford transformations and qubit permutations. For this comparison, each $\mathbb{F}_4$ generator u is replaced by the binary generators u and ωu. The Pauli graph then gives a canonical binary RREF. Different monomial classes may have the same certificate, so the distinct certificates count Pauli classes having an $\mathbb{F}_4$-linear representative.

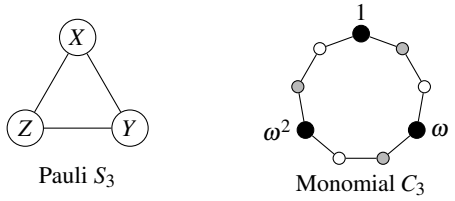


Figure 1. The two local gadgets. The Pauli triangle (left) has three ports, whose permutations form S_3. The monomial nine-cycle (right) has black, gray, and white vertices in repeating order, with the black vertices serving as ports. Its color-preserving automorphisms are the three rotations forming C_3. In both cases, word vertices attach to the ports and have a separate color.

The local Clifford transformations used here need not preserve $\mathbb{F}_4$-linearity.

7. BINARY SELF-ORTHOGONAL CODES

For binary codes, each active coordinate has only one nonzero value. Both the extension and the equivalence graph simplify. We consider binary linear codes $B \le \mathbb{F}_2^n$ satisfying

$$u \cdot v = 0 \qquad \text{for every } u, v \in B. \tag{7.1}$$

Since $B \subseteq B^{\perp}$, its rank r is at most $\lfloor n/2 \rfloor$. Taking the product of a word with itself shows that every codeword has even weight, since $u \cdot u$ is the parity of $\mathrm{wt}(u)$.

We classify these codes under coordinate permutations. There are no nontrivial coordinate scalings over $\mathbb{F}_2$, so the equivalence group is S_n and no local gadget is needed. We use a bipartite graph with differently colored sets of coordinate vertices and selected nonzero codeword vertices. A word vertex joins each coordinate where it has a 1. As in the $\mathbb{F}_4$ search, we select complete Hamming weight layers until they span the code. Graph canonical labeling orders the coordinates, and binary RREF gives the equivalence key.

The binary staircase has one extension type. The new row begins with a block of ones on unused coordinates. At rank one, the full-support generator is the all-ones word, which satisfies the self-product condition exactly at even lengths. At higher ranks, `binarysearch.py` extends full-support parents of shorter length. We clear pivot bits and check orthogonality against the parent as the row is filled. The condition $v \cdot v = 0$ is checked on completion.

Columns with identical entries in the preceding rows can be permuted freely. Thus an interval with w ones has its least arrangement when all zeros precede the ones, giving the integer $2^w - 1$. The outer and inner orbit-minimum tests are as in Section 4.

The program stores only full-support codes. To recover unrestricted counts, we need only restore unused coordinates, since a self-orthogonal code cannot contain a weight-one word. Deleting unused coordinates preserves the rank. Write $B^{\mathrm{full}}_{n,r}$ and $B^{\mathrm{all}}_{n,r}$ for the full-support and unrestricted counts. Then

$$B^{\mathrm{all}}_{n,r} = \sum_{m=0}^{n} B^{\mathrm{full}}_{m,r}. \tag{7.2}$$

Here $B^{\mathrm{full}}_{0,0} = 1$, and counts with invalid parameters are zero. The same formula applies to monomial counts of Hermitian self-orthogonal $\mathbb{F}_4$-linear codes. Before comparing with published tables, we account for any difference in zero-column conventions. For binary codes, Hou's formulas give an independent reference [5].

8. SYMMETRY MASSES AND EXACT VALIDATION

The search produces a list of inequivalent representatives. For an independent completeness check, we count the labeled spaces represented by these classes. Each class is an orbit of G_n, and its size is determined by the automorphism group of a representative. We can therefore compare the total size of the represented orbits with a direct count of all labeled isotropic spaces.

8.1. Counting labeled isotropic subspaces

Let $L(n,k)$ count the rank-r totally isotropic subspaces of $\mathbb{F}_2^{2n}$, where $r = n-k$. We first count their ordered bases. Suppose j independent, pairwise orthogonal vectors have been chosen. Their span has dimension j, and its symplectic complement has dimension $2n-j$. The next vector must belong to this complement but lie outside the span, giving $2^{2n-j} - 2^j$ choices. Multiplying these numbers counts all ordered isotropic bases of length r. Every rank-r space has $\prod_{j=0}^{r-1}(2^r - 2^j)$ ordered bases, so division gives

$$L(n,k) = \prod_{j=0}^{r-1} \frac{2^{2(n-j)} - 1}{2^{r-j} - 1}. \tag{8.1}$$

The empty product is 1, and invalid parameters give $L(n,k) = 0$. We count phase-free Pauli spaces, so there is no additional factor for stabilizer signs.

Now let $\mathcal{R}_{n,k}$ contain one representative of each G_n-class. The orbit of C has $|G_n|/|\mathrm{Aut}(C)|$ members. Since the orbits partition the labeled spaces, summing their sizes and dividing by $|G_n| = 6^n n!$ gives

$$\sum_{C \in \mathcal{R}_{n,k}} \frac{1}{|\mathrm{Aut}(C)|} = \frac{L(n,k)}{6^n n!}. \tag{8.2}$$

The left side is called the symmetry mass. Thus the automorphism orders supply a check even though the number of equivalence classes is not known in advance. Cross and Vandeth used this identity to check their enumeration [1]. The mass also bounds the number of classes from below, since each summand is at most one.

8.2. The unpadded labeled count

The count $L(n,k)$ includes spaces with unused or fixed coordinates. Our search stores unpadded spaces, so its mass must instead be compared with the number $L^{\mathrm{un}}(n,k)$ of labeled spaces having neither type of coordinate. We obtain this number by inclusion-exclusion.

Require a chosen set of u coordinates to be unused and another set of p coordinates to be fixed. There are $\binom{n}{u}\binom{n-u}{p}$ choices of these sets. At each fixed coordinate there are three possible nonidentity Pauli lines, giving a further factor 3^p. Removing the chosen coordinates leaves a rank-$(r-p)$ isotropic space on $n-u-p$ qubits, with $k-u$ logical qubits. Its remaining coordinates are unrestricted. A term may therefore include spaces with further unused or fixed coordinates, as required for inclusion-exclusion. Hence

$$L^{\mathrm{un}}(n,k) = \sum_{u=0}^{k}\sum_{p=0}^{r}(-1)^{u+p}\binom{n}{u}\binom{n-u}{p}3^p \cdot L(n-u-p,k-u). \tag{8.3}$$

For a complete set $\mathcal{R}^{\mathrm{un}}_{n,k}$ of unpadded representatives, the corresponding mass identity is

$$\sum_{C\in\mathcal{R}^{\mathrm{un}}_{n,k}}\frac{1}{|\operatorname{Aut}(C)|} = \frac{L^{\mathrm{un}}(n,k)}{6^n n!}. \tag{8.4}$$

Both the positions of the chosen coordinates and their Pauli lines distinguish labeled spaces. They contribute no multiplicity to (2.3), since equivalence allows coordinate permutations and local Clifford transformations. This accounts for the extra factors in (8.3).

8.3. Checks and computational results

To evaluate the left side of (8.4), we compute the automorphism order of each representative. We implemented this calculation independently in Python with pybliss [6] and in Sage. The Python checker, `symmetry_mass.py`, constructs the full colored Pauli graph. The Sage checker uses `Graph.automorphism_group` with a vertex partition. Both use exact integer and rational arithmetic. All completed checks agreed exactly with (8.4).

For this equality to certify completeness, the checked representatives must be valid unpadded spaces of the specified rank, no two may be G_n-equivalent, and their automorphism orders must be correct. Under these conditions their orbits are disjoint. Multiplying the mass by $6^n n!$ gives their total size, which equals $L^{\mathrm{un}}(n,k)$. No labeled space is then missing. The same method applies to the field-linear searches with their respective equivalence groups and labeled counting formulas.

Table II records the original ten-qubit run, which took about eight hours. The recorded representatives have full support but may contain weight-one stabilizers. Its counts $F_{10,k}$ therefore differ from the unpadded counts in Table I. Most of the elapsed time was spent in the stages with $k=3$ and $k=2$.

TABLE II. Stage records for the original ten-qubit run. The representatives are full-support classes $F_{10,k}$, allowing weight-one stabilizers. Times are elapsed seconds.

k	Representatives	nauty calls	Seconds
8	37	37	0.00
7	1,490	2,075	0.13
6	69,191	118,525	6.33
5	2,050,543	3,160,862	199.46
4	17,973,002	25,070,292	2,756.59
3	29,364,454	39,658,723	11,174.28
2	7,688,319	10,394,423	11,602.72
1	344,374	482,706	2,511.25
0	3,990	5,225	228.38

TABLE III. Full-support self-orthogonal code counts. The $\mathbb{F}_4$-linear counts at length 14 use monomial and Pauli equivalence. Binary counts at length 12 use coordinate permutations. Dimensions s and r are over $\mathbb{F}_4$ and $\mathbb{F}_2$, respectively.

	$\mathbb{F}_4$-linear ($n=14$)		Binary ($n=12$)	
s	Monomial	Pauli	r	Permutation
1	1	1	1	1
2	12	12	2	6
3	168	156	3	12
4	1,294	1,027	4	16
5	1,773	1,363	5	11
6	323	291	6	3
7	21	21	–	–

Table III gives the full-support counts from the two field-linear searches. At dimension $s=7$, the length-14 $\mathbb{F}_4$-linear search gives 21 classes under both monomial and Pauli equivalence. At intermediate dimensions the two counts can differ, as distinct monomial classes may belong to the same Pauli class. The binary column records permutation classes at length 12.

9. INDECOMPOSABLE CODES AND JOINT WEIGHT STATISTICS

The full classification includes codes formed by placing shorter codes on disjoint sets of qubits. We now separate these product codes from the indecomposable classes and record the stabilizer and logical weights of the latter.

A stabilizer space is *decomposable* if a partition of the qubits into two nonempty sets makes it a direct sum of spaces supported on the two sets. Otherwise it is *indecomposable*. An unused or fixed coordinate splits off a one-qubit factor, so every indecomposable code of length greater than one is unpadded. The converse fails. An unpadded code may still split into larger factors.

To find these factors, we put the generators in full binary RREF in the packed bit order and form a graph on the qubits. Two qubits are joined when they occur together in the support of a reduced row. Disconnected components give a direct-sum decomposition. Conversely, for a decomposable space, the component RREF bases together form its full RREF, in global

TABLE IV. Joint statistics of indecomposable ten-qubit stabilizer codes. The four panels list all 106 nonzero counts N and continue from left to right. The weights are defined in (9.1). Quantum distance is undefined for $k = 0$ and is marked by a dash. The counts sum to 56 743 701.

k	d_{iso}	d_{coiso}	d_Q	N	k	d_{iso}	d_{coiso}	d_Q	N	k	d_{iso}	d_{coiso}	d_Q	N	k	d_{iso}	d_{coiso}	d_Q	N
0	2	2	–	2436	2	6	4	4	2	4	5	3	3	1387	6	7	2	2	83
0	3	3	–	576	3	2	1	1	3406726	4	6	1	1	4298	6	8	2	2	5
0	4	4	–	120	3	2	2	2	2833834	4	6	2	2	32805	7	2	1	1	130
1	2	1	1	33084	3	2	2	3	13008	4	6	3	3	594	7	2	2	2	1
1	2	2	2	97364	3	3	1	1	3750392	5	2	1	1	195949	7	3	1	1	163
1	2	2	3	57237	3	3	2	2	4205811	5	2	2	2	13841	7	3	2	2	1
1	2	2	4	15	3	3	3	3	35370	5	3	1	1	280713	7	4	1	1	432
1	3	1	1	10177	3	4	1	1	4327008	5	3	2	2	21923	7	4	2	2	4
1	3	2	2	45616	3	4	2	2	8856052	5	4	1	1	749308	7	5	1	1	343
1	3	3	3	44298	3	4	3	3	259187	5	4	2	2	104279	7	5	2	2	2
1	4	1	1	1778	3	5	1	1	123501	5	5	1	1	454649	7	6	1	1	272
1	4	2	2	11321	3	5	2	2	1043434	5	5	2	2	151278	7	6	2	2	7
1	4	3	3	20884	3	5	3	3	143577	5	6	1	1	25819	7	7	1	1	39
1	4	4	4	48	3	6	1	1	14	5	6	2	2	33668	7	7	2	2	4
1	5	4	4	2	3	6	2	2	740	5	7	2	2	7	7	8	1	1	3
2	2	1	1	900112	3	6	3	3	355	6	2	1	1	6534	7	8	2	2	2
2	2	2	2	1651997	4	2	1	1	1895006	6	2	2	2	113	8	2	1	1	2
2	2	2	3	161043	4	2	2	2	538460	6	3	1	1	8919	8	3	1	1	2
2	3	1	1	621744	4	2	2	3	1	6	3	2	2	115	8	4	1	1	5
2	3	2	2	1684893	4	3	1	1	2590355	6	4	1	1	24244	8	5	1	1	5
2	3	3	3	273811	4	3	2	2	913372	6	4	2	2	669	8	6	1	1	8
2	4	1	1	292897	4	3	3	3	6	6	5	1	1	19437	8	7	1	1	4
2	4	2	2	1384879	4	4	1	1	5378839	6	5	2	2	893	8	8	1	1	4
2	4	3	3	511051	4	4	2	2	3272333	6	6	1	1	5708	8	9	1	1	1
2	5	1	1	55	4	4	3	3	143	6	6	2	2	1043	8	10	2	2	1
2	5	2	2	4322	4	5	1	1	1193617	6	7	1	1	44	9	10	1	1	1
2	5	3	3	10194	4	5	2	2	2000843										

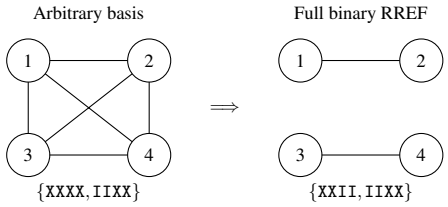


Figure 2. Support graphs from two bases of the same decomposable space. The arbitrary basis (left) gives a connected graph. The full binary RREF (right) reveals the two components $\{1,2\}$ and $\{3,4\}$.

pivot order. Uniqueness of RREF therefore prevents a reduced row from crossing a factor. An arbitrary basis would not suffice. For example, $\mathrm{span}\{XXXX, IIXX\} = \mathrm{span}\{XXII, IIXX\}$, although the first displayed generator meets all four qubits. Figure 2 illustrates why the graph must be constructed from the full RREF.

The joint census contains 56 743 701 indecomposable classes among the 58 217 368 ten-qubit classes. They are obtained by filtering the 56 802 076 unpadded representatives. Thus 58 375 unpadded classes are still decomposable. There is no indecomposable rank-zero code at length ten. Table IV records the distribution by the following three weights.

The weight of a Pauli word is its number of nonidentity coordinates. Write $C^{\perp_s}$ for the symplectic dual of C. The isotropic and coisotropic weights are the least nonzero weights in C and $C^{\perp_s}$, respectively. The coisotropic weight is the classical minimum distance of the additive code $C^{\perp_s}$. For $k > 0$, the quantum distance instead excludes the stabilizer words from the dual,

$$\begin{aligned} d_{\mathrm{iso}} &= \min_{0 \neq c \in C} \mathrm{wt}(c), \\ d_{\mathrm{coiso}} &= \min_{0 \neq v \in C^{\perp_s}} \mathrm{wt}(v), \\ d_Q &= \min_{v \in C^{\perp_s} \setminus C} \mathrm{wt}(v). \end{aligned} \tag{9.1}$$

The isotropic weight concerns all stabilizer words, including products of the chosen generators. Since $C \subseteq C^{\perp_s}$, we have $d_{\mathrm{coiso}} = \min(d_{\mathrm{iso}}, d_Q)$ for $k > 0$. At $k = 0$ the space equals its dual. The first two weights agree, and d_Q is left undefined.

These weights can differ greatly. The rank-one code $\mathrm{span}\{X^{10}\}$ has $(d_{\mathrm{iso}}, d_{\mathrm{coiso}}, d_Q) = (10, 1, 1)$, since a single X commutes with its generator and lies outside C. Table IV also contains fifteen $[[10,1]]$ classes with weights $(2,2,4)$. Their weight-two dual words are stabilizers, while their least nontrivial logical operators have weight four.

These quantities can be computed without listing the dual. Let a_i count weight-i words in C, and let b_j count weight-j

words in $C^{\perp_s}$. The additive MacWilliams identity gives [3]

$$b_j = 2^{-r} \sum_{i=0}^{n} K_j(i) a_i,$$
$$K_j(i) = \sum_{t=0}^{j} (-1)^t 3^{j-t} \binom{i}{t} \binom{n-i}{j-t}. \tag{9.2}$$

Binomial coefficients outside their valid range are zero. We obtain the a_i by enumerating the 2^r stabilizer words, at most 1 024 at length ten. Among $j \geq 1$, the first positive a_j gives d_{iso}, the first positive b_j gives d_{coiso}, and the first positive $b_j - a_j$ gives d_Q. The postprocessor uses exact integer arithmetic and checks divisibility and $b_j \geq a_j$ for each transformed coefficient it uses.

The program `classify_joint_statistics.py` checks the input generators for length, independence, and commutation, then increments the corresponding $(k, d_{\text{iso}}, d_{\text{coiso}}, d_Q)$ entry. The resulting `joint_statistics.csv` has 106 nonzero entries. This calculation assumes a complete list with one representative per equivalence class. It does not repeat the equivalence search, so its completeness rests on the underlying enumeration and its checks.

10. CONCLUSION

We have extended the classification of Cross and Vandeth to ten qubits, obtaining 58 217 368 equivalence classes across all stabilizer ranks. Of these, 56 743 701 are indecomposable. We also computed their joint distribution by isotropic weight, coisotropic weight, and quantum distance.

The main idea of our search is to eliminate redundant descriptions before completing a generator matrix. We construct matrices one row at a time, reject partial rows that cannot be lexicographically first, and discard comparisons that can no longer lead to a rejection. Graph canonical labeling identifies the remaining equivalences, while canonical RREF matrices provide compact exact certificates. We use symmetry-mass identities to check the enumeration and adapt the same methods to Hermitian self-orthogonal $\mathbb{F}_4$-linear codes and binary self-orthogonal codes.

At eleven qubits, the number of candidates and memory requirements grow substantially. We are pursuing this extension with a parallel C++ implementation.

ACKNOWLEDGMENTS

We are grateful to Markus Grassl, who pointed us to Cross and Vandeth's work, shared partial enumerations, and urged us to run symmetry-mass checks. I also thank Cross and Vandeth for their reference classification through nine qubits.

[1] Andrew Cross and Drew Vandeth. Small binary stabilizer subsystem codes. arXiv:2501.17447, 2025. 1, 7

[2] Lars Eirik Danielsen and Matthew G. Parker. On the classification of all self-dual additive codes over GF(4) of length up to 12. *Journal of Combinatorial Theory, Series A*, 113(7):1351–1367, 2006. 1

[3] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. Quantum error correction via codes over GF(4). *IEEE Transactions on Information Theory*, 44(4):1369–1387, 1998. 1, 6, 10

[4] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014. 5

[5] Xiang-dong Hou. On the number of inequivalent binary self-orthogonal codes. *IEEE Transactions on Information Theory*, 53(7):2459–2479, 2007. 7

[6] Kaushik Kulkarni. pybliss: Python bindings for the bliss tool. https://github.com/kaushikcfd/pybliss. Software. Accessed October 22, 2026. 8