Investigations on Automorphism Groups of Quantum Stabilizer Codes

Hanson Hao

Abstract

The stabilizer formalism for quantum error-correcting codes has been, without doubt, the most successful at producing examples of quantum codes with strong error-correcting properties. In this paper, we discuss *strong* automorphism groups of stabilizer codes, beginning with the analogous notion from the theory of classical codes. Two weakenings of this concept, the *weak* automorphism group and *Clifford-twisted* automorphism group, are also discussed, along with many examples highlighting the possible relationships between the types of "automorphism groups". In particular, we construct an example of a $[[10, 0, 4]]$ stabilizer code showing how the Clifford-twisted automorphism groups might be connected to the Mathieu groups. Finally, nonexistence results are proved regarding stabilizer codes with highly transitive strong and weak automorphism groups, suggesting a potential inverse relationship between the error-correcting properties of a quantum code and the transitivity of those automorphism groups.

Contents

1 Introduction

Although quantum computers are thought to be significantly more efficient than classical computers (e.g. Shor's algorithm for integer factorization), they are also inherently more susceptible to noise and breakdown processes. The construction of a *quantum error-correcting code* (QECC) in [10] demonstrated how protection against a single-qubit error was possible, which was a milestone towards the practical realization of quantum computers. However, the problem of finding "good" QECCs is still difficult, and their theory is still in an early stage of development, compared to, say, the theory of classical error-correcting codes. We were interested in this very general question of finding "good" QECCs, with a particular emphasis on investigating how "symmetric" a QECC could be. We were also motivated by the recent paper of Harvey and Moore [7], in which possible connections are found among QECCs, conformal field theories, and the Mathieu moonshine phenomena. Similar theories for classical codes revolve around finding their automorphism groups, which explains the motivations behind some of the definitions and results in Section 2 and Section 3.

This paper is structured as follows: in the remainder of this section we will present a brief background of quantum error correction and *stabilizer codes*, following [5] and Chapter 10 of [8]. We will put more emphasis on the mathematical structures and skip much of the physical intuition behind QECCs. In Section 2 we will introduce three notions of automorphism groups of quantum codes, which we call *strong*, *weak*, and *Clifford-twisted*. We prove some basic results regarding these different types of automorphism groups. The bulk of this section, however, will be devoted to giving examples of all three types of automorphism groups of small QECCs, along with explaining how they might be calculated by hand. Particularly interesting is the example in Section 2.2.6, which gives evidence of a possible connection between Clifford-twisted automorphism groups of stabilizer codes and the Mathieu groups. In Section 3 we will show in certain cases that the strong and weak automorphism group of "good" stabilizer codes cannot be highly transitive, which suggests that the Clifford-twisted automorphism groups might have the most interesting properties. Along the way, we mention some problems that we found interesting, but did not have time to think about.

Acknowledgements

This research was supported by the Stanford Undergraduate Research Institute in Mathematics (SURIM) program during the summer of 2021. The author would like to thank Dr. Pawel Grzegrzolka for coordinating the program. The author also thanks his mentor, Dr. Daniel Bump, for introducing him to the area of quantum error correction, and also for many helpful conversations and insightful suggestions. His idea of investigating "twisted automorphisms" (Definitions 2.6, 2.13) was particularly fruitful.

1.1 Prelude: Classical Error Correction

We give a very brief overview of classical error correction to serve as motivation for the constructions of quantum error correction theory, which will be presented in Section 1.2. First, recall that in classical computing, the basic unit of computation is the *bit*, which takes a state of either 0 or 1; i.e. an element of $\mathbb{F}_2$. Thus an n-bit state is just an element of $\mathbb{F}_2^n$.

The fundamental idea of classical error correction is repetition. In particular, if we want to transmit k bits of information, we would repeat that information in such a way that it would be "hard" to confuse slight moderations of the encoded information with an encoding of a different k bits of information. For instance, if we wanted to send the bits 0 and 1 through a somewhat noisy channel, we could instead send the 3-bit sequences 000 and 111, so that even if there is a 1-bit error, we could use a "majority rules" scheme to correct the error. As an explicit example, if we received 010, which is not a possible codeword, we could conclude with high probability that 000 was the intended message. In this manner, we are essentially embedding one bit into three by the identification of $\mathbb{F}_2^1$ with the 1-dimensional subspace $\{000, 111\}$ in $\mathbb{F}_2^3$, so we call such a scheme a *linear code*. We will soon see how the same ideas apply in the quantum world, although there are some caveats.

1.2 Quantum Error Correction

Definition 1.1. *A* qubit *is the basic unit of quantum computation. It is represented as the complex vector space $\mathbb{C}^2$ with basis vectors $\mathbf{0}$ and $\mathbf{1}$, so the state of a qubit is given by a nonzero linear combination (superposition) $a\mathbf{0} + b\mathbf{1}$.*

Some notational remarks are in order. First, we will write vectors/states in boldface, instead of the bra-ket convention popular in physics. Second, we will not bother with normalizing the state $a\mathbf{0} + b\mathbf{1}$ of a qubit so that $|a|^2 + |b|^2 = 1$, as that does not affect our discussion.

Definition 1.2. *A space of n qubits is represented as the tensor product $(\mathbb{C}^2)^{\otimes n} = \underbrace{\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2}_{n \text{ times}}$, where tensor products are taken over $\mathbb{C}$. A basis for this space is given by*

$$\{\mathbf{0}\ldots\mathbf{0}, \mathbf{0}\ldots\mathbf{01}, \ldots, \mathbf{1}\ldots\mathbf{1}\},$$

the 2^n binary vectors of length n.

We will usually refer to such a space of n qubits as $\mathbb{C}^{2^n}$.

The central idea of quantum error correction is to identify a k-qubit space with some 2^k-dimensional linear subspace, called the *codespace*, of an n-qubit space, called the *ambient space*, where $n > k$. We will call states in the ambient 2^n-dimensional space *physical*, and

states in the 2^k-dimensional codespace *logical*. For example, if we encode a one-qubit space into three qubits by the map $\mathbf{0} \mapsto \mathbf{000}, \mathbf{1} \mapsto \mathbf{111}$, we may refer to the physical state $\mathbf{000}$ as "logical $\mathbf{0}$", denoted $\mathbf{0}_L$.

In this formalism, errors will just be linear operators acting on the codespace. To have error-correcting properties, this codespace should be redundant in some sense, but there is an added difficulty in that we are not allowed to create repetitions of quantum states, as we might do in classical computing. This is the *no-cloning theorem*, as discussed in Theorem 1 of [5]. Therefore, this codespace should consist of highly entangled states—"redundancy without repetition". Moreover, in general physical situations, one may encounter errors randomly affecting a single qubit, or a small number of qubits, of the ambient space. A good code allows such errors to be corrected.

We now state the quantum error-correction conditions as a "black box". A full derivation can be found in Section 10.3 of [8].

Theorem 1.3. *Let $C \subset \mathbb{C}^{2^n}$ be a quantum code, and let P be the orthogonal projection onto C. Then C can correct a set of errors $\mathcal{E} = \{E_i\}$ if and only if there is a complex Hermitian matrix (α_{ij}) such that*

$$PE_i^* E_j P = \alpha_{ij} P, \tag{1}$$

where E_i, E_j run over all operators in $\mathcal{E}$, and * is the conjugate transpose.

Using Theorem 1.3, one can show that if a code C corrects a set of errors $\{E_i\}$, then it also corrects *any linear combination* of the E_i. This is an extremely powerful observation, because instead of possibly having to correct a continuum of errors, it now suffices to focus only on correcting a discrete set of error operations that span the full set of errors we want to correct. Because of this observation, it makes sense to introduce the Pauli matrices:

Definition 1.4. *The four Pauli matrices are defined as follows:*

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

We call X the *bit flip* operator, since it sends $\mathbf{0}$ to $\mathbf{1}$ and vice versa. Similarly, we call Z the *phase flip* operator, since it sends $\mathbf{0}$ to itself, but $\mathbf{1}$ to $-\mathbf{1}$. Finally, Y is a combined bit-phase operator, with an extra factor of i thrown in to make the mathematics easier. The Pauli matrices act on the one-qubit space $\mathbb{C}^2$, but n-fold tensor products of them act on an n-qubit space in the natural way. For example, the tensor product $X \otimes Z \otimes I$ acts on $\mathbb{C}^{2^3}$ by sending $\mathbf{000}$ to $\mathbf{100}$, $\mathbf{111}$ to $-\mathbf{011}$, etc. In the rest of this paper, we will simply write tensor products of Pauli matrices as concatenations, so that the above $X \otimes Z \otimes I$ operator is simply XZI.

The Pauli matrices have remarkable properties that will be fully exploited in the rest of the paper. First, they are all both unitary and Hermitian, and they form a basis of the 2-by-2

complex matrices. So by the discussion after Theorem 1.3, if we wanted to, for instance, correct arbitrary errors occurring on the first qubit of a three-qubit system, we simply need to correct the errors XII, ZII, and YII (along with III, which is automatic). Second, all of the Pauli matrices square to the identity, and they all commute or anticommute. In particular, two Pauli matrices anticommute if and only if they are different nonidentity matrices. This implies that the Pauli matrices form a projective representation of the four-group $\mathbb{Z}_2 \times \mathbb{Z}_2$, which is why they are chosen as our basis of the 2-by-2 complex matrices. Since complex scalars, in general, do not concern us, the fact that the Pauli matrices act like the elements of $\mathbb{Z}_2 \times \mathbb{Z}_2$ is of great utility.

1.3 The Pauli Group and Stabilizer Codes

Now, although the quantum error-correction condition 1 is easy to verify for any particular code and set of errors, it is difficult to actually construct a code correcting a given set of error operations, particularly if that set is large. Indeed, we are usually interested in correcting sets of errors such as "all one-qubit errors," which necessitates correcting an error set of size $3n$ if the ambient space is an n-qubit space (we need to correct an X, Z, and Y error at each physical qubit). The stabilizer formalism introduced by Gottesman [4] provides a convenient workaround to this problem.

Definition 1.5. *The one-qubit* Pauli group*, denoted G_1, is the group of matrices generated by X, Y, and Z. This is a group of order 16:*

$$G_1 = \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}.$$

Note that the elements of G_1 are just Pauli matrices multiplied by some phase factor, which is a fourth root of unity. All elements in G_1 square to plus or minus the identity, and all elements commute or anticommute. This means that G_1 is "almost" an elementary abelian 2-group: its commutator subgroup is $\langle -I \rangle$, and $G_1/[G_1, G_1]$ is indeed an elementary abelian 2-group of order 2^{2n+1}. It will also be useful to speak of G_1 mod its center $Z(G_1) = \langle iI \rangle$, so that we can consider elements without worrying about scalar multiples. We denote $G_1/Z(G_1)$ by P_1.

It is natural to generalize the Pauli group to n-fold tensor products:

Definition 1.6. *The n-qubit Pauli group, denoted G_n, is the group of matrices generated by n-fold tensor products of the Pauli matrices. This is a group of order 4^{n+1}, since there are 4^n possible tensor products, along with 4 possible phases (the fourth roots of unity).*

Again, $[G_n, G_n] = \langle -I \rangle$, $G_n/[G_n, G_n]$ is an elementary abelian 2-group, and $Z(G_n) = \langle iI \rangle$. We denote $G_n/Z(G_n)$ by P_n.

Before moving on to stabilizer codes, we mention an extremely useful way of writing elements of P_n, known as the *check matrix*. We see that an element $s \in P_n$ can uniquely be written as

$$X^{a_1}Z^{b_1} \otimes X^{a_2}Z^{b_2} \otimes \ldots \otimes X^{a_n}Z^{b_n},$$

where the a_i, b_i are 0 or 1. Therefore we represent s as

$$(a|b) = \left[\begin{array}{cccc|cccc} a_1 & a_2 & \ldots & a_n & b_1 & b_2 & \ldots & b_n \end{array}\right], \tag{3}$$

which can be thought of as a vector in $\mathbb{F}_2^{2n}$. For example, the element $IXYZ$ is represented as

$$\left[\begin{array}{cccc|cccc} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right].$$

The utility of this representation becomes clear when we define a symplectic bilinear form over $\mathbb{F}_2^{2n}$ given by

$$\langle (a|b), (c|d) \rangle = a \cdot d + b \cdot c, \tag{4}$$

where $\cdot$ is the usual dot product in $\mathbb{F}_2^n$. Then it is straightforward to check that if $s, s' \in P_n$ have check matrix representations $(a|b)$, $(a'|b')$ respectively, s and s' commute if and only if $\langle (a|b), (a'|b') \rangle = 0$. Since P_n is simply G_n modded out by its center, the same is true for any lifts of s, s' to elements in G_n. The check matrix representation gives us a compact way of writing set of elements in G_n or P_n, which will be useful later on.

We are now ready to define stabilizer codes. The key is to consider the action of G_n on $\mathbb{C}^{2^n}$, and consider the fixed points of a subgroup of G_n.

Definition 1.7. *Let S be an* abelian *subgroup of G_n* not containing $-I$. *Then*

$$C(S) = \{v \in \mathbb{C}^{2^n} : sv = v \text{ for all } s \in S\} \tag{5}$$

is the stabilizer code *corresponding to S. We call S the* stabilizing subgroup *corresponding to $C(S)$.*

It is easy to check that $C(S)$ is a subspace of the ambient space.

It is important to make a few remarks regarding this definition. First, $C(S)$ must contain *all* vectors in $\mathbb{C}^{2^n}$ that are fixed by everything in S. In particular, for a code C' to be called a stabilizer code, there must be some abelian subgroup S of G_n, not containing $-I$, such that $C' = C(S)$. It is not enough for $C' \subset C(S)$. Second, we do not want S to contain $-I$. Otherwise, if $-I \in S$, then for any $v \in C(S)$, we have $v = -Iv \Rightarrow v = 0$, so that $C(S)$ is trivial. Similarly, we require S to be abelian: if $s, s' \in S$ do not commute, then they must anticommute, so that for any $v \in C(S)$, we have $v = (ss')v = (-s's)v = -(s'sv) = -v \Rightarrow v = 0$. Note that because elements in G_n either square to I or $-I$, these two conditions imply that S is an elementary abelian 2-group. Also, elements in S may only have a scalar factor of plus or minus 1, lest they square to $-I$.

Given the construction of stabilizer codes, we would like to somehow connect a stabilizing subgroup $S \subset G_n$ with the dimension of $C(S)$, as well as with the errors that $C(S)$ can correct. Let us first answer the former question. First, note that S can have size at most 2^n. To see why, suppose S has m independent generators (so $|S| = 2^m$), which we write in the check matrix format. We may think of S as an m-dimensional subspace of the $\mathbb{F}_2$-vector space $\mathbb{F}_2^{2n}$. Because these generators all commute, S is self-orthogonal with respect to the bilinear form in 4; that is, $S \subseteq S^\perp$. But $\dim S + \dim S^\perp = 2n$, so that $m = \dim S \leq n$.

With $m \leq n$ in mind, it now makes sense to state the following proposition:

Proposition 1.8. *Let C be the stabilizer code corresponding to a stabilizing subgroup $S \subset G_n$. If S has m independent generators, or equivalently $|S| = 2^m$, then C has dimension 2^{n-m}; that is, it encodes $k \coloneqq n - m$ logical qubits.*

Proof. We claim that the orthogonal projection onto C is given by $P = \frac{1}{|S|}\sum_{s\in S} s$. We must check that

1. $P = P^*$,
2. $Pv = v$ for all $v \in C$,
3. $P^2 = P$.

For (1), we notice that the elements of S are, up to sign, tensor products of the Pauli matrices, which are all Hermitian. Hence any $s \in S$ is Hermitian, so P is as well. For (2), we calculate $Pv = \frac{1}{|S|}\sum_{s\in S} sv = \frac{1}{|S|}\sum_{s\in S} v = v$, since $v \in C$ implies $sv = v$ for all $s \in S$. For (3), we have

$$P^2 = \left(\frac{1}{|S|}\sum_{s\in S} s\right)\left(\frac{1}{|S|}\sum_{t\in S} t\right) = \frac{1}{|S|}\sum_{s\in S}\left(\frac{1}{|S|}\sum_{t\in S} st\right) = \frac{1}{|S|}\sum_{s\in S} P = P,$$

since summing over all $st \in S$, for s fixed, is the same as summing over all $t \in S$.

So because P is an orthogonal projection onto C, we have $\dim C = \mathrm{rk}\ P = \mathrm{tr}P$. Notice that X, Y, and Z are traceless. Using this and the fact that $-I \notin S$, we see that the only term in $\sum_{s\in S} s$ that has nonzero trace is the identity. Hence $\mathrm{tr}P = \frac{1}{|S|}(\mathrm{tr}I) = \frac{2^n}{|S|} = 2^{n-m}$. □

In this situation, we call C an $[[n, k]]$ code, where the double brackets are meant to distinguish the quantum code C from classical codes. From now on, S will always be assumed to an abelian subgroup of G_n not containing $-I$, and we will just write C for $C(S)$ if there is no ambiguity in the stabilizing subgroup S.

We now discuss the error-correcting properties of C. First, we need to define a slight abuse of notation:

Definition 1.9. *Let $N(S)$ be the normalizer of S in G_n. We say that an element $p \in P_n$ is in $N(S) - S$ if there is some lift $g \in G_n$ of p in $N(S) - S$.*

The idea behind this definition is that we can and should disregard the scalar phase of any element of G_n, since we know that if C corrects some error E, it corrects any scalar multiple of E. Notice that $N(S)$ is equal to the centralizer of S, since two elements of G_n either commute or anticommute, and $-I \notin S$.

Along with this definition, we must reinterpret the quantum error-correction conditions, Theorem 1.3, in the language of stabilizer codes.

Theorem 1.10. *Let C be the stabilizer code corresponding to a subgroup S. If $\{E_i\}$ is a set of error operators in G_n such that $E_i^* E_j \notin N(S) - S$ for all i and j, then the $\{E_i\}$ are correctable.*

Proof. See Theorem 10.8, [8]. □

The criterion in Theorem 1.10 is much easier to use than that in Theorem 1.3. Instead of having to construct the projection matrix onto our code and do various matrix calculations, the error-correcting properties of a stabilizer code can be computed only using knowledge of G_n. Because of this, we introduce a few more definitions regarding elements of G_n and P_n:

Definition 1.11. *The* weight *of an element $p \in P_n$ is the number of non-identity tensor factors in p.*

For instance, the element $XZIIZ \in P_5$ has weight 3, since it has three non-identity tensor factors.

Definition 1.12. *Let C be an $[[n, k]]$ stabilizer code corresponding to a subgroup $S \subset G_n$, where $k \geq 1$. Then the* distance *d of C is defined as*

$$d = \min\{\text{weight}(p) : p \in P_n, p \in N(S) - S\}. \tag{6}$$

Recall that we write $p \in N(S) - S$ as in the sense of Definition 1.9. In this case, we call C a $[[n, k, d]]$ code.

Remark 1.13. *In the degenerate case $k = 0$, where C is 1-dimensional (i.e. a single state), $N(S) - S$ is empty, so the above definition does not make sense. We follow the convention of [3] and say that the distance is*

$$d = \min\{\text{weight}(p) : p \in S, p \neq I\}. \tag{7}$$

From these definitions, we see that the product of any two weight w operators has weight at most $2w$, and that the conjugate transpose does not change the weight of an operator. Then it follows from Theorem 1.10 that any code with distance greater than $2w$ can correct errors affecting any w qubits.

Example 1.14. *Consider a subgroup $S \subset G_5$ generated by*

$$\{XZZXI, IXZZX, XIXZZ, ZXIXZ\}.$$

These elements all pairwise commute, and S does not contain $-I$, so the stabilizer code C corresponding to S encodes $5 - 4 = 1$ logical qubit. Moreover, one can directly verify that the distance of S is 3—for instance, $XIZIX \in N(S) - S$, but there are no elements $p \in P_n$ of lesser weight in $N(S) - S$. This shows that C can correct any error affecting one qubit, since $3 > 2 \cdot 1$.

Using the projection matrix of a stabilizer code as in Proposition 1.8, we can find a basis for C:

$$\begin{aligned}
\mathbf{0}_L &= \mathbf{00000} + \mathbf{10010} + \mathbf{01001} + \mathbf{10100} \\
&\quad + \mathbf{01010} - \mathbf{11011} - \mathbf{00110} - \mathbf{11000} \\
&\quad - \mathbf{11101} - \mathbf{00011} - \mathbf{11110} - \mathbf{01111} \\
&\quad - \mathbf{10001} - \mathbf{01100} - \mathbf{10111} + \mathbf{00101} \\
\mathbf{1}_L &= (XXXXX)\mathbf{0}_L \\
&= \mathbf{11111} + \mathbf{01101} + \mathbf{10110} + \mathbf{01011} \\
&\quad + \mathbf{10101} - \mathbf{00100} - \mathbf{11001} - \mathbf{00111} \\
&\quad - \mathbf{00010} - \mathbf{11100} - \mathbf{00001} - \mathbf{10000} \\
&\quad - \mathbf{01110} - \mathbf{10011} - \mathbf{01000} + \mathbf{11010}.
\end{aligned}$$

Remark 1.15. *The aforementioned stabilizer code is the smallest able to correct one error on any qubit, in terms of the number n of physical qubits.*

More examples of small stabilizer codes, as well as more details about their theory, can be found in Section 10.5 of [8]. Many more examples of stabilizer codes, including codes with larger parameters $[[n, k, d]]$, can be found at [6]. We will reference those tables of quantum codes in the below sections.

2 Automorphism Groups of Codes

2.1 Definitions and Basic Results

Just like in the theory of classical codes, we can define the automorphism group of a QECC:

Definition 2.1. *Let C be a quantum code. There is a natural action of S_n on the n-qubit ambient space $\mathbb{C}^{2^n}$. We define the* strong automorphism group *of C as*

$$\mathrm{Aut}_{\mathrm{strong}}(C) = \{\sigma \in S_n : \sigma(C) = C\}. \tag{8}$$

We are particularly interested in the case where C is a stabilizer code with stabilizing subgroup S. Indeed, in this case we may also consider the action of S_n on G_n given by permuting tensor factors. This action is equivalently given by $\sigma(g) = M_\sigma g M_\sigma^{-1}$, where $g \in G_n$ and M_σ^{-1} is the permutation matrix associated to σ in the natural basis of $\mathbb{C}^{2^n}$. Then the above condition for $\mathrm{Aut}_{\mathrm{strong}}(C)$ can be rephrased in terms of S:

Proposition 2.2. *Let C be a nontrivial (i.e. nonzero) stabilizer code corresponding to a subgroup $S \subset G_n$. Then $\sigma \in \mathrm{Aut}_{\mathrm{strong}}(C)$ if and only if $\sigma(S) = S$.*

Proof. Notice that $\sigma(S)$ fixes $\sigma(C)$ pointwise. Hence if $\sigma(S) = S$, it follows that $\sigma(C) \subseteq C$. But σ acts as an invertible linear map $C \to \sigma(C)$, which implies $\sigma(C) = C$. This proves the "if" direction. Conversely, if $\sigma(C) = C$, then $\sigma(C)$ is fixed pointwise by S. It is also fixed pointwise by $\sigma(S)$, so the same is true for the join $\langle S, \sigma(S)\rangle$ of the two subgroups. Because C is nontrivial, $\langle S, \sigma(S)\rangle$ must be abelian and must not contain $-I$. Then by Proposition 1.8, $\langle S, \sigma(S)\rangle$ must have the same size as S, which implies $\sigma(S) \subseteq S$. But σ is bijective, so we have equality, proving the reverse direction. □

Since all $\sigma \in S_n$ act as invertible linear maps on S, we can simplify the result of Proposition 2.2 to a form that is more suitable for actually computing strong automorphism groups.

Corollary 2.3. *Let C be a nontrivial stabilizer code corresponding to a subgroup $S \subset G_n$. Let S be generated by $g_1, \ldots, g_m$. Then $\sigma \in \mathrm{Aut}_{\mathrm{strong}}(C)$ if and only if $\sigma(g_i) \in S$ for each i.*

Example 2.4. *Let $S \subset G_3$ be generated by XZZ and ZXZ, so that $S = \{III, XZZ, ZXZ, YYI\}$. Using the Kronecker product for matrices, we can calculate the projection onto $C = C(S)$ as*

$$P = \frac{1}{|S|}\sum_{s\in S} s = \frac{1}{4}\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 & 0 & -1 \\ 1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & 1 \\ -1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix},$$

so that C is spanned by $\mathbf{0}_L = \frac{1}{2}(\mathbf{000} + \mathbf{010} + \mathbf{100} - \mathbf{110})$ and $\mathbf{1}_L = \frac{1}{2}(\mathbf{001} - \mathbf{011} - \mathbf{101} - \mathbf{111})$. Checking each of the permutations in S_3, we obtain $\mathrm{Aut}_{\mathrm{strong}}(C) = \{(1), (12)\} \cong \mathbb{Z}_2$. Indeed, these are also the only permutations σ that satisfy $\sigma(S) = S$, as is easily seen.

Remark 2.5. *At this point, it is worth mentioning that the strong automorphism group of a stabilizer code cannot be characterized by the parameters $[[n, k, d]]$. For instance, let $S = \{IIII, XXXX, ZZZZ, YYYY\}$ and $S' = \{IIII, XXZZ, YYXX, ZZYY\}$ be subgroups of G_4. Then the corresponding stabilizer codes C and C' both have parameters $[[4, 2, 2]]$. But $\mathrm{Aut}_{\mathrm{strong}}(C) = S_4$, while $\mathrm{Aut}_{\mathrm{strong}}(C') = \{(1), (12), (34), (12)(34)\} \cong \mathbb{Z}_2 \times \mathbb{Z}_2$.*

It turns out that the notion of "strong automorphism group" does not produce many interesting examples for stabilizer codes with small n; in particular, it seems that for most codes, the strong automorphism group is usually small compared to the full symmetric group. We will see more examples in Section 2.2, and we will give some reasons as to why this general heuristic might be true in Section 3. Therefore we now introduce an extension of the automorphism group concept.

Definition 2.6. *Let C be a quantum code. We define the* weak automorphism group *of C as*

$$\mathrm{Aut}_{\mathrm{weak}}(C) = \{\sigma \in S_n : \sigma(C) = \gamma_\sigma \cdot C \text{ for some } \gamma_\sigma \in G_n\}. \tag{9}$$

That is, $\sigma(C)$ is "almost" C, where we allow a twist of the vectors in C by some element γ_σ of the Pauli group (which is allowed to vary depending on σ, and is not necessarily unique). We should check that $\mathrm{Aut}_{\mathrm{weak}}(C)$ is actually a group. Indeed this is the case, since we can view $\mathrm{Aut}_{\mathrm{weak}}(C)$ as a semidirect product of sorts: if $\sigma, \tau \in \mathrm{Aut}_{\mathrm{weak}}(C)$ correspond to $\gamma_\sigma, \gamma_\tau \in G_n$, then

$$\sigma\tau(C) = (\sigma(\gamma_\tau) \cdot \gamma_\sigma) \cdot C, \tag{10}$$

as can be easily verified. Recall that the action of σ on G_n by permuting tensor factors is the same as the matrix action by conjugation, $g \mapsto M_\sigma g M_\sigma^{-1}$. Hence $\sigma\tau \in G_n$, since we can set $\gamma_{\sigma\tau} = \sigma(\gamma_\tau) \cdot \gamma_\sigma$.

We now want to somewhat justify the labels "strong automorphism" and "weak automorphism". In Proposition 2.2, we were able to give a characterization of strong automorphisms in terms of the stabilizing subgroup of a stabilizer code. We can do the same for weak automorphisms:

Proposition 2.7. *Let C be a nontrivial stabilizer code corresponding to a subgroup $S \subset G_n$. Then $\sigma \in \mathrm{Aut}_{\mathrm{weak}}(C)$ if and only if $\sigma(S) \subset S \cup -S$, where $-S = \{-s : s \in S\}$.*

Corollary 2.8. *Since σ is bijective, the following conditions are equivalent to the one in Proposition 2.7:*

1. *If S is generated by $g_1, \ldots, g_m$, then $\sigma \in \mathrm{Aut}_{\mathrm{weak}}(C)$ if and only if $\sigma(g_i)$ or $-\sigma(g_i)$ is in S for each i.*

2. *If π is the projection $G_n \to G_n/\langle -I \rangle$, then $\pi(\sigma(S)) = \pi(S)$; that is, $\sigma(S)$ and S are the same subgroup up to sign.*

Proof of Proposition 2.7. First, if $\sigma \in \mathrm{Aut}_{\mathrm{weak}}(C)$, then $\sigma(C) = \gamma_\sigma \cdot C$ for some $\gamma_\sigma \in G_n$. Because C and $\sigma(C)$ have the same dimension, it follows from dimension considerations (Proposition 1.8) that $\sigma(S)$ is exactly the set of elements stabilizing $\sigma(C)$. Similarly, because

γ_σ is invertible, it follows that $\gamma_\sigma S \gamma_\sigma^{-1}$ is exactly the set of elements stabilizing $\sigma(C)$, so $\sigma(S) = \gamma_\sigma S \gamma_\sigma^{-1}$. But γ_σ commutes or anti-commutes with everything in S, so $\sigma(S) = \gamma_\sigma S \gamma_\sigma^{-1} \subset S \cup -S$.

Conversely, suppose $\sigma(S) \subset S \cup -S$. σ is an isomorphism of abelian groups, and $\sigma(S)$ does not contain $-I$ (otherwise $\sigma^{-1}(-I) = -I \in S$, contradiction). Since $S \cup -S = \{\pm I, \pm a_1, \pm a_2, \ldots\}$ for an enumeration of the elements of S, by counting and the fact that $-I \notin \sigma(S)$, we see that plus or minus s is in $\sigma(S)$ for any $s \in S$. In particular, if $s_1, \ldots, s_m$ are an independent set of generators for S, we may find unique $r_1, \ldots, r_m \in S$ such that $\sigma(r_i)$ equals plus or minus s_i. It follows that the $\{\sigma(r_i)\}$ generate $\sigma(S)$.

Set $e_i = \sigma(r_i) s_i$, so $e_i \in \{\pm 1\}$. Renumber the r's, s's, and e's so that $e_1 = \ldots = e_a = 1$, and $e_{a+1} = \ldots = e_m = -1$. We would like to produce an element $\gamma \in G_n$ such that γ commutes with $s_1, \ldots, s_a$, and anticommutes with $s_{a+1}, \ldots, s_m$ (if $a = m$, then take $\gamma = I$, so we can assume $a < m$). It suffices to find a γ that commutes with $\{s_1, \ldots, s_a, s_{a+1}s_{a+2}, s_{a+1}s_{a+3}, \ldots, s_{a+1}s_m\}$, and anticommutes with s_{a+1}.

Recall the check matrix representation of elements in G_n (where we disregard elements of the center $\langle iI \rangle$), so we may consider S as an m-dimensional subspace of $\mathbb{F}_2^{2n}$. It follows that the elements in S commuting with all of $\{s_1, \ldots, s_a, s_{a+1}s_{a+2}, s_{a+1}s_{a+3}, \ldots, s_{a+1}s_m\}$ form a subspace V of dimension $2n - (m-1)$, which is the orthogonal complement with respect to the symplectic bilinear form 4. Similarly, the elements in S commuting with all of $\{s_1, \ldots, s_a, \mathbf{s_{a+1}}, s_{a+1}s_{a+2}, s_{a+1}s_{a+3}, \ldots, s_{a+1}s_m\}$ form a subspace W of dimension $2n - m$. Therefore we may find some element in V but not in W, and the corresponding $\gamma \in G_n$ (taking the phase scalar factor to be 1) is the desired element.

It follows that $\gamma S \gamma^{-1} \cong S$ is generated by

$$\{s_1, \ldots, s_a, -s_{a+1}, \ldots, -s_m\} = \{\sigma(r_1), \ldots, \sigma(r_m)\},$$

so that $\gamma S \gamma^{-1} = \sigma(S)$. $\gamma S \gamma^{-1}$ is the stabilizer for $\gamma \cdot C$, and $\sigma(S)$ is the stabilizer for $\sigma(C)$, so $\gamma \cdot C = \sigma(C)$. Setting $\gamma_\sigma \coloneqq \gamma$, we have $\sigma \in \mathrm{Aut}_{\mathrm{weak}}(C)$. □

This somewhat justifies the "strong" and "weak" labels, since Propositions 2.2 and 2.7 show that a strong automorphism of $C(S)$ sends S to itself, while a weak automorphism sends S to itself but "disregarding signs". Of course the strong automorphism group is a subgroup of the weak automorphism group.

Example 2.9. *Let $S \subset G_3$ be generated by XXX, YYI, and ZXZ, so that $S = \{III, XXX, YYI, ZXZ, -ZZX, -YIY, XZZ, -IYY\}$. Then Corollaries 2.3 and 2.8 show that the strong automorphism group of $C = C(S)$ is $\{(1), (12)\} \cong \mathbb{Z}_2$, while the weak automorphism group is all of S_3.*

This example shows that this weaker notion of automorphism can enlarge the strong automorphism group, and we will see in some later examples that $\mathrm{Aut}_{\mathrm{weak}}(C)$ can be quite

a bit larger than $\mathrm{Aut}_{\mathrm{strong}}(C)$ for certain stabilizer codes C. This example also shows that $\mathrm{Aut}_{\mathrm{strong}}(C)$ is not necessarily normal in $\mathrm{Aut}_{\mathrm{weak}}(C)$. So we may pose the following (somewhat vague) question:

Question 2.10. *Can we describe the relationship between* $\mathrm{Aut}_{\mathrm{strong}}(C)$ *and* $\mathrm{Aut}_{\mathrm{weak}}(C)$ *for a given stabilizer code* $C = C(S)$*?*

We extend the notion of an automorphism group one last time. For this, we need to introduce the (one-qubit) *Clifford group* L_1, following the terminology of [1] and [2].

Definition 2.11. *The (one-qubit)* Clifford group L_1 *is the subgroup of the normalizer of* G_1 *in* $U(2)$ *that contains entries from* $\mathbb{Q}\left(\frac{1+i}{\sqrt{2}}\right) = \mathbb{Q}(e^{\frac{\pi i}{4}})$*. It is generated by* G_1*,* $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$*, and* $S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$*.*

In the terminology of quantum logic gates, H is the *Hadamard gate*, and S is the *phase shift gate* where the shift is by $\frac{\pi}{4}$. Note that the conjugation actions of H and S on X, Y, and Z are as follows:

$HXH^{-1} = Z$	$SXS^{-1} = Y$
$HZH^{-1} = X$	$SZS^{-1} = Z$
$HYH^{-1} = -Y$	$SYS^{-1} = -X$

.

We can extend the action of L_1 on G_1 to n qubits as follows, which gives us our second weakening of the strong automorphism group:

Definition 2.12. *We define the* n*-qubit* diagonal Clifford group L_n *as*

$$L_n = \{A_1 \otimes \ldots \otimes A_n : A_i \in L_1\}. \tag{11}$$

L_n has a natural conjugation action on G_n, which motivates the following definition.

Definition 2.13. *Let* C *be a stabilizer code corresponding to the subgroup* $S \subset G_n$*. We define the* Clifford-twisted automorphism group *of* C *as*

$$\mathrm{Aut}_{\mathrm{Clif}}(C) = \{\sigma \in S_n : \sigma(S) = \lambda_\sigma S \lambda_\sigma^{-1} \text{ for some } \lambda_\sigma \in L_n\}. \tag{12}$$

Since L_n contains G_n, we have the chain of inclusions $\mathrm{Aut}_{\mathrm{Clif}}(C) \supseteq \mathrm{Aut}_{\mathrm{weak}}(C) \supseteq \mathrm{Aut}_{\mathrm{strong}}(C)$ for a stabilizer code C. We are somewhat justified in calling $\mathrm{Aut}_{\mathrm{Clif}}(C)$ an automorphism group: it is a group for the same reason that $\mathrm{Aut}_{\mathrm{weak}}(C)$ is a group, and because the elements of L_n act as automorphisms of G_n, it is not hard to show that if $\sigma \in \mathrm{Aut}_{\mathrm{Clif}}(C)$, then C and $\sigma(C)$ have the same $[[n, k, d]]$ parameters.

To aid computation, we make some useful observations involving Definition 2.13. First, if $g_1, \ldots, g_m$ generate S, then for σ to be in $\mathrm{Aut}_{\mathrm{Clif}}(C)$ it suffices to verify that $\sigma(g_i) \in \lambda_\sigma S \lambda_\sigma^{-1}$

for each g_i. This can be further weakened: it suffices that plus or minus $\sigma(g_i)$ is in $\lambda_\sigma S \lambda_\sigma^{-1}$, since if we are only off by a sign, we can simultaneously correct for that by conjugating by an appropriate element of G_n, as in the proof of Proposition 2.7. Finally, because we can disregard signs, we can interpret the action of L_1 on G_1 (or each tensor factor of G_n) as S_3 acting on the non-identity Pauli matrices. For instance, the above table shows that H induces the permutation (XZ) and S induces the permutation (XY) on the set $\{X, Y, Z\}$, and various combinations of H and S induce the other permutations of S_3. So we can rephrase Definition 2.13 in a more verbose, but more intuitive, manner:

Corollary 2.14. *Let C be a stabilizer code corresponding to the subgroup $S \subset G_n$, and consider the componentwise action of $(S_3)^n := \underbrace{S_3 \times \ldots \times S_3}_{n \text{ times}}$ on elements of G_n (i.e. on each tensor factor, S_3 acts on the three non-identity Pauli matrices). Then $\sigma \in \mathrm{Aut}_{\mathrm{Clif}}(C)$ if and only if there is some $\rho_\sigma \in (S_3)^n$ such that for each generator g_i of S, $\rho_\sigma \cdot (\sigma(g_i)) \in S \cup -S$.*

Note that we use $\cdot$ to emphasize the fact that σ and ρ_σ have very different actions: σ permutes the tensor factors in each element, while ρ_σ permutes X, Y, and Z in each tensor factor.

Let us use Corollary 2.14 in an example. Recall from Remark 2.5 that if C is the stabilizer code corresponding to the subgroup $S \subset G_4$ generated by $XXZZ$ and $YYXX$, then $\mathrm{Aut}_{\mathrm{strong}}(C) = \{(1), (12), (34), (12)(34)\} \cong \mathbb{Z}_2 \times \mathbb{Z}_2$. Using Corollary 2.8, it is easy to see that $\mathrm{Aut}_{\mathrm{weak}}(C)$ is the same group. However, we now show that $\mathrm{Aut}_{\mathrm{Clif}}(C)$ is the full S_4.

Example 2.15. *We want to show that $\mathrm{Aut}_{\mathrm{Clif}}(C) = S_4$, and we already know $(12) \in \mathrm{Aut}_{\mathrm{Clif}}(C)$, so it suffices to show that $(1234) \in \mathrm{Aut}_{\mathrm{Clif}}(C)$. Applying this permutation to the generators $XXZZ$ and $YYXX$ gives $ZXXZ$ and $XYYX$. The latter two elements can be obtained from the former two by applying the permutation (XZY) in the first tensor factor, and (XYZ) in the third. Hence $(1234) \in \mathrm{Aut}_{\mathrm{Clif}}(C)$.*

2.2 Examples of Automorphism Groups

In this section we give many examples of $\mathrm{Aut}_{\mathrm{strong}}(C)$, $\mathrm{Aut}_{\mathrm{weak}}(C)$, and $\mathrm{Aut}_{\mathrm{Clif}}(C)$ for various small stabilizer codes C with large distance. Most codes will be taken from [6]. The majority of these examples were calculated by hand and then verified using a SAGE program (see Appendix A). These hand calculations are somewhat tedious *ad hoc* processes, so we will only try to give a general outline of how they were done. We will also make some observations about the more noteworthy automorphism groups. However, we should again remark that the automorphism groups of a stabilizer code cannot be completely determined by the parameters $[[n, k, d]]$, so these examples should be taken as interesting specific cases rather than showcasing a general phenomenon.

2.2.1 A $[[5, 1, 3]]$ code

Recall from Example 1.14 that the subgroup $S \subset G_5$ generated by

$$XZZXI$$
$$IXZZX$$
$$XIXZZ$$
$$ZXIXZ$$

gives a $[[5, 1, 3]]$ stabilizer code C, and it is the smallest able to correct one error on any qubit in terms of the number n of physical qubits. We find that $\mathrm{Aut}_{\mathrm{strong}}(C) = \mathrm{Aut}_{\mathrm{weak}}(C) = \langle (12345), (25)(34) \rangle \cong D_{10}$, the dihedral group of order 10. One could compute these groups by writing out the elements of S in full:

$$\begin{array}{cccc} XZZXI & XYIYX & ZIZYY & ZYYZI \\ IXZZX & IZYYZ & YXXYI & YIYXX \\ XIXZZ & YYZIZ & IYXXY & ZZXIX \\ ZXIXZ & XXYIY & YZIZY & IIIII \end{array}$$

Then it is clear that $\mathrm{Aut}_{\mathrm{strong}}(C) = \mathrm{Aut}_{\mathrm{weak}}(C)$, since all of the elements have the same phase $+1$, and moreover any strong automorphism must take the four generators of S to a cyclic permutation of $XZZXI$. It should follow pretty easily that $\mathrm{Aut}_{\mathrm{strong}}(C) = \langle (12345), (25)(34) \rangle$. We call stabilizer codes C with $(12 \ldots n) \in \mathrm{Aut}_{\mathrm{strong}}(C)$ (or more generally, any n-cycle in place of $(12 \ldots n)$) *cyclic*, and they are of particular interest (see [4]).

It is much harder to come up with a systematic approach for calculating the Clifford-twisted automorphism group, so we must resort to "guessing" elements in $\mathrm{Aut}_{\mathrm{Clif}}(C)$. It is a good idea to begin with testing (12) and $(12 \ldots n)$, since those two permutations generate S_n. In this case, we already have $(12345) \in \mathrm{Aut}_{\mathrm{Clif}}(C)$, so we try (12). This permutation sends the generators to

$$ZXZXI$$
$$XIZZX$$
$$IXXZZ$$
$$XZIXZ,$$

and twisting these by the permutations (YZ), (YZ), (XZ), (XY), and (XZ) in the respective tensor factors gives

$$YXXYI$$
$$XIXZZ$$
$$IXZZX$$
$$XYIYX,$$

all of which are in S. Hence $(12) \in \mathrm{Aut}_{\mathrm{Clif}}(C) \Rightarrow \mathrm{Aut}_{\mathrm{Clif}}(C) = S_5$.

2.2.2 A $[[6, 0, 4]]$ code

Recall the $\mathbf{0}_L$ and $\mathbf{1}_L$ states from Example 1.14, which form a basis for the $[[5, 1, 3]]$ code mentioned above. We may consider the state $\mathbf{0} \otimes \mathbf{0}_L + \mathbf{1} \otimes \mathbf{1}_L \in \mathbb{C}^{2^6}$, which defines a (degenerate) $[[6, 0, 4]]$ code C. This "code" is important as an example of a *maximally entangled state*, following the terminology of [7].

It is not hard to check that the stabilizing subgroup $S \subset G_6$ corresponding to C is generated by

$$\begin{gathered} IXZZXI \\ IIXZZX \\ IXIXZZ \\ IZXIXZ \\ XXXXXX \\ ZZZZZZ. \end{gathered}$$

It turns out that while $\mathrm{Aut}_{\mathrm{strong}}(C)$ is again $\langle (23456), (36)(45) \rangle \cong D_{10}$, $\mathrm{Aut}_{\mathrm{weak}}(C)$ is in fact $\langle (23456), (135)(264) \rangle \cong \mathrm{PSL}(2, 5) \cong A_5$. For instance, $(135)(264)$ sends $IXZZXI$ to $XZIIZX$, which is not in S, but rather in $-S$.

Note that this is another example where $\mathrm{Aut}_{\mathrm{strong}}(C)$ is not normal in $\mathrm{Aut}_{\mathrm{weak}}(C)$.

To calculate these groups, we adopt the procedure from Section 2.2.1 and write out all the $2^{6-0} = 64$ elements of S. Since any permutation fixes $XXXXXX$ and $ZZZZZZ$, we can focus on the first four generators of S. It is a good strategy to do casework on the image of the first tensor factor, since we would then need to find four elements in S with I's in that slot (perhaps with the proper sign, depending on whether we are calculating the strong or weak automorphism group). We also use the constraint that the first four generators must be sent to elements of S (or $-S$) that have 2 I's, 2 X's, and 2 Z's as tensor factors. All of these types of criteria—location of the I's in the tensor product, weights of elements, and the types of Pauli matrices used in each tensor product—are highly useful when performing hand computations.

Finally, we claim that $\mathrm{Aut}_{\mathrm{Clif}}(C) = S_6$. First, we check that $(12) \in \mathrm{Aut}_{\mathrm{Clif}}(C)$: it sends

the generators to

$$XIZZXI$$
$$IIXZZX$$
$$XIIXZZ$$
$$ZIXIXZ$$
$$XXXXXX$$
$$ZZZZZZ,$$

and twisting these by the permutations (XY), (XY), (XZ), (YZ), (YZ) and (XZ) in the respective tensor factors gives

$$YIXYXI$$
$$IIZYYZ$$
$$YIIXYX$$
$$ZIZIXX$$
$$YYZXXZ$$
$$ZZXYYX,$$

all of which are in S. It remains to check that $(123456) \in \mathrm{Aut}_{\mathrm{Clif}}(C)$. This sends the generators to

$$IIXZZX$$
$$XIIXZZ$$
$$ZIXIXZ$$
$$ZIZXIX$$
$$XXXXXX$$
$$ZZZZZZ,$$

and twisting these by the permutations (XY), (XY), (XZ), (YZ), (YZ) and (XZ) in the respective tensor factors gives

$$IIZYYZ$$
$$YIIXYX$$
$$ZIZIXX$$
$$ZIXXIZ$$
$$YYZXXZ$$
$$ZZXYYX,$$

all of which are in S. Again, for this computation, it is important to exploit the position of I's as tensor factors, since conjugation by elements of L_6 cannot change I.

2.2.3 The $[[7, 1, 3]]$ Steane code

The $[[7, 1, 3]]$ Steane code C corresponds to a subgroup $S \subset G_7$ that has generators given by

$$IIIXXXX$$
$$IXXIIXX$$
$$XIXIXIX$$
$$IIIZZZZ$$
$$IZZIIZZ$$
$$ZIZIZIZ.$$

C is derived from the classical $[7, 4]$ Hamming code—this is apparent if one writes out the check matrix representations of these generators, and compares that matrix to the parity-check matrix of the Hamming code. It is of primary interest as an example of a *CSS code* (see Section 10.4.2 of [8] for the full details). Since it is derived from the Hamming code, which has an automorphism group of $\mathrm{PGL}(3, 2) \cong \mathrm{PSL}(2, 7)$, it should be of no surprise that $\mathrm{Aut}_{\mathrm{strong}}(C) = \mathrm{Aut}_{\mathrm{weak}}(C) = \langle (46)(57), (124)(365) \rangle \cong \mathrm{PGL}(3, 2)$. In general, it is at least true that the strong automorphism group of a CSS code is isomorphic to the automorphism group of the classical code it is derived from.

By writing out all of the $2^{7-1} = 64$ elements of S, one can check that $\mathrm{Aut}_{\mathrm{Clif}}(C)$ is also $\mathrm{PGL}(3, 2)$. This provides a nontrivial example where all three types of automorphism group are equal.

2.2.4 An $[[8, 3, 3]]$ code

The tables at [6] give us the generators for the stabilizing subgroup $S \subset G_8$ of an $[[8, 3, 3]]$ code C:

$$XIZIYZXY$$
$$IXZZYXYI$$
$$IZXIYYZX$$
$$IZIYZXXY$$
$$ZZZZZZZZ.$$

This code is significant because it is the smallest-sized code encoding 3 logical qubits that can correct any single-qubit error. We can calculate that

$$\begin{aligned}\mathrm{Aut}_{\mathrm{strong}}(C) = \{&(1), (12)(35)(68)(47), (13)(25)(48)(67), (14)(27)(38)(56), (15)(23)(46)(78),\\ &(16)(28)(37)(45), (17)(24)(36)(58), (18)(26)(34)(57)\} \cong \mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2,\end{aligned}$$

$$\mathrm{Aut}_{\mathrm{weak}}(C) = \mathrm{Aut}_{\mathrm{strong}}(C) \rtimes \langle (2453876) \rangle \cong (\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2) \rtimes \mathbb{Z}_7.$$

We may alternatively identify $\mathrm{Aut}_{\mathrm{weak}}(C)$ as $\mathrm{AGL}(1,8)$, the 1-dimensional affine general linear group over $\mathbb{F}_8$, which acts sharply 2-transitively on 8 points. This is a group of order 56 with a unique Sylow 2-subgroup isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_2$.

To calculate these groups, one can use the following remarkable description of the elements of S: each of the $2^{8-3} - 4 = 28$ elements in S that are not $IIIIIIII$, $XXXXXXXX$, $ZZZZZZZZ$, or $YYYYYYYY$, contain exactly 2 I's, 2 X's, 2 Z's, and 2 Y's as tensor factors. Moreover, for any pair of integers $1 \leq i < j \leq 8$, exactly one of those 28 elements has I's as tensor factors in slots i and j.

To calculate $\mathrm{Aut}_{\mathrm{Clif}}(C)$, we use the fact that $\mathrm{Aut}_{\mathrm{Clif}}(C) \supseteq \mathrm{Aut}_{\mathrm{weak}}(C)$ is 2-transitive, so it suffices to find the stabilizer of slots 1 and 2. A tedious check by casework (using the positions of I tensor factors, as well as the above observations about S, as our guide) shows that this stabilizer is of order 3 and generated by $(367)(458)$. Hence $\mathrm{Aut}_{\mathrm{Clif}}(C)$ is a group of order $56 \cdot 3 = 168$. We may identify $\mathrm{Aut}_{\mathrm{Clif}}(C)$ as the 1-dimensional affine *semilinear* group $\mathrm{A\Gamma L}(1,8)$, which is a semidirect product $\mathrm{AGL}(1,8) \rtimes \mathrm{Aut}(\mathbb{F}_8)$.

2.2.5 An $[[8,2,3]]$ code

It turns out that the $[[8,3,3]]$ code mentioned in 2.2.4 has an $[[8,2,3]]$ subcode C, which is the smallest-sized code encoding 2 logical qubits that can correct any single-qubit error. Its stabilizing subgroup S is generated by

$$\begin{aligned}
&XIZIYZXY\\
&IXZZYXYI\\
&IZXIYYZX\\
&IZZYZXZZ\\
&IIZIIIYX\\
&ZZZZZZZZ.
\end{aligned}$$

Note that the product of the fourth and fifth generators of S give the fourth generator of the aforementioned $[[8,3,3]]$ code.

Now, any element of $\mathrm{Aut}_{\mathrm{strong}}(C)$ or $\mathrm{Aut}_{\mathrm{strong}}(C)$ sends the generator $IIZIIIYX$ to another weight-3 element in S (up to sign). By analyzing the weight-3 elements in S (there are only 8 of them), we can conclude that the only nontrivial element in both $\mathrm{Aut}_{\mathrm{strong}}(C)$ and $\mathrm{Aut}_{\mathrm{weak}}(C)$ is $(13)(25)(48)(67)$. So although the strong automorphism group of the above $[[8,3,3]]$ code was sharply 1-transitive, and the weak automorphism group of the same code was sharply 2-transitive, neither $\mathrm{Aut}_{\mathrm{strong}}(C) \cong \mathbb{Z}_2$ nor $\mathrm{Aut}_{\mathrm{weak}}(C) = \mathrm{Aut}_{\mathrm{strong}}(C)$ are transitive.

2.2.6 A $[[10, 0, 4]]$ code

Consider the following generators of a stabilizing subgroup $S \subset G_{10}$:

$$XIIZXZXZII$$
$$IXIIZXZXZI$$
$$IIXIIZXZXZ$$
$$ZIIXIIZXZX$$
$$XZIIXIIZXZ$$
$$ZXZIIXIIZX$$
$$XZXZIIXIIZ$$
$$ZXZXZIIXII$$
$$IZXZXZIIXI$$
$$IIZXZXZIIX$$

These ten elements are cyclic permutations of each other. It can be checked that S determines a degenerate $[[10, 0, 4]]$ code C, and from [6] we see that 4 is the best possible distance for a $[[10, 0]]$ code. Note that this is not the same $[[10, 0, 4]]$ code given at [6].

Now, we will later see (Theorems 3.3 and 3.5) that there are no nondegenerate 12-qubit (resp. 11-qubit) stabilizer codes with strong or weak automorphism group M_{12} (resp. M_{11}). This suggests that the Clifford-twisted automorphism group might be the "correct" automorphism group to investigate, if we are trying to find some connections between stabilizer codes and the small Mathieu groups. This code provides some evidence of such a connection: we have

$$\mathrm{Aut}_{\mathrm{Clif}}(C) = \langle (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10), (8\ 9)(4\ 10)(5\ 6) \rangle \cong M_{10}.2,$$

a 3-transitive group of order 1440. This group is also isomorphic to the 2-dimensional projective semilinear group $\mathrm{P\Gamma L}(2, 9)$, as well as to the automorphism group of the unique (up to isomorphism) $(3, 4, 10)$ Steiner system.

It is evident that $(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10) \in \mathrm{Aut}_{\mathrm{Clif}}(C)$, since it is even a strong automorphism. To see that $(8\ 9)(4\ 10)(5\ 6) \in \mathrm{Aut}_{\mathrm{Clif}}(C)$, one applies this permutation to the above ten generators and then twists by (XZ) in the 5th, 6th, 8th, and 9th tensor factors (in the other tensor factors, we do nothing). Recall from the discussion surrounding Definition 2.11 that this (XZ) twist is, up to sign, the Hadamard gate. The resulting elements of G_{10} will all be in S. Since $M_{10}.2$ is a maximal subgroup of S_{10}, it remains to show that some transposition, say (13), is not in $\mathrm{Aut}_{\mathrm{Clif}}(C)$. This is not too hard to do using the program in Appendix A: one way is to apply (13) to the first and third generators listed above (so they become $IIXZXZXZII$ and $XIIIIZXZXZ$, respectively), and show that the resulting two elements of G_{10} cannot be simultaneously twisted into elements of S.

The general heuristic for the construction of this code is as follows: somehow the code should be related to the unique $(3, 4, 10)$ Steiner system, since as mentioned above $M_{10}.2$ is the automorphism group of that Steiner system. From [11] we obtain a description of the blocks of such a system, and we place I's in tensor factors corresponding to ten of those blocks, which will be cyclic permutations of each other. In some way, this makes sense to try, since the identity matrices act as "distinguished elements": X's, Y's, and Z's can all be twisted into each other, but that is not true with I's. The rest of the construction consists of playing around with the remaining six tensor factors and eventually coming upon the "right ones".

As a final remark, the Clifford-twisted automorphism group of this code is indeed the most interesting: both the strong and weak automorphism groups of C are
$\langle (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10), (2\ 10)(3\ 9)(4\ 8)(5\ 7) \rangle \cong D_{20}$.

2.2.7 An $[[11, 1, 5]]$ code

Table 8.5 of [4] gives the generators for the stabilizing subgroup S of an $[[11, 1, 5]]$ code C as follows:

$$
\begin{array}{l}
ZZZZZZIIIII \\
XXXXXXIIIII \\
IIIZXYYYYXZ \\
IIIXYZZZZYX \\
ZYXIIIZYXII \\
XZYIIIXZYII \\
IIIZYXXYZII \\
IIIXZYZXYII \\
ZXYIIIZZZXY \\
YZXIIIYYYZX
\end{array}
$$

This code is significant because it is the smallest-sized code that can correct arbitrary errors on any two qubits (see [6]). S is slightly too large to analyze using the aforementioned techniques, or via brute-force using the program in Appendix A; however, we strongly suspect that both $\mathrm{Aut}_{\mathrm{strong}}(C)$ and $\mathrm{Aut}_{\mathrm{weak}}(C)$ are trivial.

Question 2.16. *Are the strong and weak automorphism groups of C trivial?*

3 Stabilizer Codes with Highly Transitive Automorphism Groups

In this section, we try to investigate the general question of "*how symmetric can a good stabilizer code be?*" We will focus on stabilizer codes that have distance at least 3 (so they can at least correct one error on any qubit, hence "good"), and we will try to find those codes with multiply transitive strong and weak automorphism groups (hence "symmetric").

The examples in Section 2.2 suggest that in general, the strong and weak automorphism groups of a good code cannot be "too symmetric". For instance, the highest degree of transitivity we observed was 2-transitive, such as in the $[[7, 1, 3]]$ Steane code with strong and weak automorphism group $\mathrm{PGL}(3, 2)$ (Section 2.2.3), and in a $[[8, 3, 3]]$ code with weak automorphism group $\mathrm{AGL}(1, 8)$ (Section 2.2.4). We believe that this is a good heuristic, and we now provide some results supporting this point of view.

Theorem 3.1. *An $[[n, k, d]]$ stabilizer code C with S_n or A_n as its strong or weak automorphism group has $d \leq 2$.*

We make a useful definition:

Definition 3.2. *Let T be a subset of E_n. Then the **complexity** of T is the number of different Pauli matrices (I, X, Y, or Z) that appear as tensor factors in elements of T (so the complexity is an integer in $\{1, 2, 3, 4\}$). We define the complexity of an element similarly. For example, the subset $T = \{IXX, YYZ\}$ has complexity 4, even though both elements IXX and YYZ have complexity 2.*

Proof of Theorem 3.1. Let C be the stabilizer code in question with parameters $[[n, k, d]]$, and let $S \subset G_n$ be its stabilizing subgroup. Note that any stabilizer code with distance at least 3 must be able to correct an arbitrary error on a single qubit, which means it must be at least 5 physical qubits large (see, for example, the quantum Hamming bound discussed in Section 10.3.4 of [8]). So we may assume that $n \geq 5$.

We prove the theorem in the case $\mathrm{Aut}_{\mathrm{weak}}(C) \cong S_n$ or A_n, and the proof for $\mathrm{Aut}_{\mathrm{strong}}(C)$ is essentially the same.

Now, S has complexity 1, 2, 3, or 4. If S has complexity 1, then it is the trivial subgroup, so we can disregard this case. If S has complexity 2, then without loss of generality every element $s \in S$ is a tensor product of I's and X's. Then $XI \ldots I$ is in the normalizer $N(S)$, so either it is in S or $d = 1$. The same applies for $IXI \ldots I$, $IIXI \ldots I$, etc., so either S contains them all or $d = 1$. In the former case, S will have size 2^n, so that $k = 0$. In this case, d still equals 1, applying the distance convention for zero-dimensional stabilizer codes as discussed in Remark 1.13.

If S has complexity 3, then without loss of generality, every element $s \in S$ is a tensor product of I's, X's, and Z's. Suppose $s_1 \in S$ has an X in slot i (the ith tensor factor) and

$s_2 \in S$ has a Z in slot j. Then we may choose some permutation $\sigma \in \mathrm{Aut}_{\mathrm{weak}}(C)$, which is either S_n or A_n, such that $\sigma(i) = j$. So plus or minus $\sigma(s_1)$ is in S, and taking the positive sign (without loss of generality), we see that $\sigma(s_1)s_2 \in S$. But this element has a Y in slot j, contradicting the complexity of S.

If S has complexity 4, we first claim that no element in S can have complexity 3 or 4. If $s \in S$ had complexity 3 or 4, then it has I, X, and Z as tensor factors somewhere, say $s = \ldots I \ldots X \ldots Z \ldots$ (without loss of generality, since it will be apparent that it does not matter where these tensor factors are located). Then applying an appropriate 3-cycle $\sigma \in A_n \subseteq \mathrm{Aut}_{\mathrm{weak}}(C)$, we get $\sigma(s) = \ldots Z \ldots I \ldots X \ldots$, so that plus or minus $\sigma(s)$ is in S. In either case, the point is that $\sigma(s)$ and s do not commute, a contradiction.

So elements in S have complexity at most 2. Suppose we had some complexity 2 element s, which is, without loss of generality, a tensor product of I's and X's only, with at least 1 copy of each. Let s have an I in slot i and an X in slot j. Now, because we assume $n \geq 5$, there are at least 3 more slots, so either I or X appears in the remaining slots at least twice. Without loss of generality, suppose I appears in slots k and l of s, where i, j, k, l are pairwise distinct. Consider the permutation $\sigma = (ij)(kl) \in A_n \subseteq \mathrm{Aut}_{\mathrm{weak}}(C)$, so that $\sigma(s)$ has the effect of only switching the I in slot i and the X in the slot j (the I's in slots k and l are invisibly switched). Then plus or minus $\sigma(s)$ is in S, so taking the positive sign, we see that $t \coloneqq \sigma(s)s \in S$ has an I in every slot of the tensor product, except in slots i and j, where it has X's.

Then because n is at least 5, by applying elements in $\mathrm{Aut}_{\mathrm{weak}}(C)$, it follows that any tensor product of I's and exactly two X's is in S, up to sign. Then the elements $\{\pm XXI \ldots I, \pm IXXI \ldots I, \ldots, \pm I \ldots IXX\}$ are independent in S, so S has size at least 2^{n-1}. But the complexity 4 hypothesis means that S contains some term with a Y or Z, and that element is certainly independent from the above set, so S must have size exactly 2^n. So $k = 0$, and by the distance convention for zero-dimensional stabilizer codes, $d \leq 2$.

The only case we have not considered is where all elements in S have complexity 1. Then S is a subgroup contained in $\{\pm I \ldots I, \pm X \ldots X, \pm Z \ldots Z, \pm Y \ldots Y\}$, and $XXI \ldots I$ is a weight 2 element in $N(S) - S$, so $d = 2$. □

The classification of finite simple groups tells us that the only k-transitive groups for $k \geq 6$ are S_n and A_n for large enough n. We also know that the only 5-transitive groups are the Mathieu groups M_{24} and M_{12}, and the only 4-transitive groups are M_{23} and M_{11}. Therefore it makes sense to discuss these groups next. We will now deal with the small Mathieu groups, starting with M_{12}.

Theorem 3.3. *There is no $k \geq 1$ stabilizer code with ambient space $\mathbb{C}^{2^{12}}$ that has strong or weak automorphism group M_{12}.*

Proof. We prove the statement in the $\mathrm{Aut}_{\mathrm{strong}}$ case, since the $\mathrm{Aut}_{\mathrm{weak}}$ case only gives us increased freedom to make sign changes in elements of the stabilizer group, which doesn't

make a difference. In particular, in all that follows, we will mean "equal up to a sign $\pm$" when we say "equal", unless stated (we will only need to deal with signs in a few places). The only fact about M_{12} we will need is that it is 5-transitive as a permutation group on 12 points.

Let C be a stabilizer code (encoding at least 1 logical qubit) corresponding to the stabilizer subgroup $S \subset E_{12}$ with $\mathrm{Aut}_{\mathrm{strong}}(C) \supseteq M_{12}$. We prove that this inclusion must be strict.

Disregarding the trivial case, the complexity of S must be 2, 3, or 4. First, if the complexity of S is 2, then without loss of generality, all elements are tensor products of X's and I's. Pick some $s \in S$ not equal to I or $X \dots X$ (if there is no such element, then $\mathrm{Aut}_{\mathrm{strong}}(C) \cong S_{12}$), so that s has at most 6 X's as tensor factors. Note that we could change the X's and I's and the argument still works. We can assume without harm that the X's are located consecutively in the first slots, so s is one of the following, up to sign:

$$\{XI \dots I, XXI \dots I, XXXI \dots I, XXXXI \dots I, XXXXXI \dots I, XXXXXXIIIIII\}.$$

It does not actually matter where the X's are located, since the argument can be easily modified for any position of the X's.

Because M_{12} is 5-transitive, we may produce one of the elements in the below list by applying a permutation from $\mathrm{Aut}_{\mathrm{strong}}(C)$ to s. The dots mean that we do not exactly know the order of the remaining tensor factors, since we may only specify the images of 5 points (qubits).

$$\{IXI \dots, XIXI \dots, XXIXI \dots, XXXIX \dots, XXXXI \dots, XXXXI \dots\}.$$

Let $s' \in S$ be the corresponding element to s. Notice that in each case, ss' is an element with weight 2, except in the last case, where ss' could have weight 4. But in the last case we can just repeat this procedure to produce some element of weight 2. Let us call this process **reduction to a weight 2 element**, since we will want to refer to it later. We have essentially shown that if S contains an element of weight at most 6, then it contains an element of weight 2.

So S contains some element of weight 2. By 5-transitivity of $\mathrm{Aut}_{\mathrm{strong}}(C) \supseteq M_{12}$, S contains all tensor products of I's and X's of weight 2, of which the 11 elements $\{XXI \dots I, IXXI \dots I, \dots, I \dots IXX\}$ are independent. Since $k \geq 1$, S is generated by exactly these elements. Now, notice that if we apply the appropriate element in the 5-transitive group $\mathrm{Aut}_{\mathrm{strong}}$ to $IXXI \dots I$, we get $XIXI \dots I \in S$ *with the same sign as* $IXXI \dots I$. Multiplying these, we see that $+XXI \dots I \in S$ (with the positive sign added for emphasis). Then $XIXI \dots I \in S$, the product of the first two generators, has the same sign as $IXXI \dots I \in S$, so the transposition (12) is in $\mathrm{Aut}_{\mathrm{strong}}$ (it fixes the rest of the generators). Therefore $\mathrm{Aut}_{\mathrm{strong}}(C) \neq M_{12}$, since M_{12} does not contain any transposition. This finishes the discussion of the complexity 2 case.

As in the proof regarding S_n or A_n automorphism group, transitivity of $\mathrm{Aut}_{\mathrm{strong}}(C)$ shows that S cannot have complexity 3.

It remains to discuss the complexity 4 case. First, we show that there cannot be an element in S with complexity 2. This argument essentially follows the above work where S has complexity 2: given an element $s \in S$ with complexity 2, we may reduce to a weight 2 element $t \in S$ that only has I's and one other Pauli matrix as tensor factors (say, X). Then the same argument shows that S would have to be generated by $\{XXI \dots I, IXXI \dots I, \dots, I \dots IXX\}$ (by the $k \geq 1$ hypothesis). This is a contradiction to S being complexity 4.

So there is an element in S with complexity at least 3 (if all elements in S had complexity 1, then $\mathrm{Aut}_{\mathrm{strong}}(C) \cong S_{12} \neq M_{12}$). We claim that there is in fact an element in S with complexity 4. Suppose the first three slots of $s \in S$ are XIZ (as usual, it doesn't matter what order they are in, what slots they are in, or what Pauli matrices they are, as long as they are pairwise distinct). There are four possibilities:

1. s, interpreted as a string of Pauli matrices, begins $XIZI \dots$. By 5-transitivity of $\mathrm{Aut}_{\mathrm{strong}}(C)$ there is some $s' \in S$ that begins $ZXII \dots$. Then their product ss' begins $YXZI \dots$, the desired element of complexity 4.

2. s begins $XIZX \dots$. By 5-transitivity of $\mathrm{Aut}_{\mathrm{strong}}(C)$ there is some $s' \in S$ that begins $IZXX \dots$. Then their product ss' begins $XZYI \dots$, the desired element of complexity 4.

3. s begins $XIZZ \dots$. By 5-transitivity of $\mathrm{Aut}_{\mathrm{strong}}(C)$ there is some $s' \in S$ that begins $IZXZ \dots$. Then their product ss' begins $XZYI \dots$, the desired element of complexity 4.

4. s begins $XIZY \dots$. This already has complexity 4.

Therefore we may assume without loss of generality that s has complexity 4, and that the first four tensor factors of s are $XZYI$.

Now, we look at elements of $N(S) - S$. By the bounds found at [6], $N(S) - S$ contains an element of weight at most 6. We turn to looking at the possibilities case by case:

1. Weight 1:

   (a) $XI \dots I \notin N(S)$ because it does not commute with $ZIXY \dots$. A similar argument shows that $YI \dots I$, $ZI \dots I \notin N(S)$.

2. Weight 2:

   (a) $XXI \dots I \notin N(S)$ because it does not commute with $ZIXY \dots$. A similar argument shows that $YYI \dots I$, $ZZI \dots I \notin N(S)$.

   (b) $XZI \dots I \notin N(S)$ because it does not commute with $ZIXY \dots$. A similar argument shows that $ZXI \dots I, XYI \dots I, YXI \dots I, YZI \dots I, ZYI \dots I \notin N(S)$.

3. Weights 3 through 6: it is easily seen that if $g \in N(S)$, then $\sigma(g) \in N(\sigma(S)) = N(S)$ for $\sigma \in \mathrm{Aut}_{\mathrm{strong}}(C)$ (the same argument works for $\mathrm{Aut}_{\mathrm{weak}}(C)$, since in that case $\sigma(S)$ only differs from S by signs). Then if $g \in N(S)$ has weight 3, 4, 5, or 6, we may reduce to an element $g' \in N(S)$ of weight 2. This falls into the above case.

In all cases, we get a contradiction, so we are done.

□

Remark 3.4. *This method fails with M_{23} and M_{24}. We know that there is a $[[23, 1, 7]]$ stabilizer code that at least has strong automorphism group M_{23}; this is the CSS code generated from the classical $[23, 12, 7]$ Golay code (see section 7.15.4 of [9]. There should also be a $[[24, 0, 8]]$ "code" that is the CSS code generated from the classical $[24, 12, 8]$ extended Golay code.*

One can use the same method to show the following:

Theorem 3.5. *There is no $k \geq 1$ stabilizer code with ambient space $\mathbb{C}^{2^{11}}$ that has strong or weak automorphism group M_{11}.*

The main difference is that reduction to a weight 2 element is now only valid for elements of weight at most 5. But luckily this does not pose a problem: $\lfloor \frac{11}{2} \rfloor = 5$, so the arguments for complexity 2 elements still hold, and the bounds at [6] show that $N(S) - S$ for any stabilizing subgroup $S \subset G_{11}$ contains an element of weight at most 5, so the last casework step can be repeated. Every other step in the proof of Theorem 3.3 can be done using only 4-transitivity.

We end this paper by presenting some questions that naturally follow from topics discussed above.

Question 3.6. *Are Theorems 3.3 and 3.5 true if the $k \geq 1$ hypothesis is dropped?*

Question 3.7. *Describe stabilizer codes C with $\mathrm{Aut}_{\mathrm{Clif}}(C) \cong S_n$. Replace S_n by A_n and the Mathieu groups, if possible.*

Question 3.8. *Describe stabilizer codes C with 3-transitive $\mathrm{Aut}_{\mathrm{strong}}(C)$ or $\mathrm{Aut}_{\mathrm{weak}}(C)$. Replace "3-transitive" by "2-transitive", "transitive", and "cyclic" if possible.*

Question 3.9. *Can a stabilizer code with trivial strong (resp. weak) automorphism group be arbitrarily good? That is, are there stabilizer codes with trivial strong (resp. weak) automorphism groups having arbitrarily large distance? Replace "trivial" with "cyclic" if possible.*

References

[1] Beverley Bolt, T. G. Room, and G. E. Wall. On the clifford collineation, transform and similarity groups. i. *Journal of the Australian Mathematical Society*, 2(1):60–79, 1961.

[2] Beverley Bolt, T. G. Room, and G. E. Wall. On the clifford collineation, transform and similarity groups. ii. *Journal of The Australian Mathematical Society*, 2:80–96, 1961.

[3] A.R. Calderbank, E.M. Rains, P.W. Shor, and N.J.A. Sloane. Quantum error correction via codes over GF(4). pages 292–, 1997.

[4] Daniel Gottesman. *Stabilizer codes and quantum error correction*. PhD thesis, California Institute of Technology, May 1997.

[5] Daniel Gottesman. An introduction to quantum error correction and fault-tolerant quantum computation. 2009.

[6] Markus Grassl. Quantum error-correcting code tables. `http://codetables.de/`, 2019. Accessed: 2021-08-08.

[7] Jeffrey A. Harvey and Gregory W. Moore. Moonshine, superconformal symmetry, and quantum error correction. *Journal of High Energy Physics*, 2020(5), May 2020.

[8] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition, 2011.

[9] John Preskill. Chapter 7 of notes on quantum computation. `http://theory.caltech.edu/~preskill/ph229/notes/chap7.pdf`.

[10] Peter W. Shor. Scheme for reducing decoherence in quantum computer memory. *Phys. Rev. A*, 52:R2493–R2496, Oct 1995.

[11] Eric W. Weisstein. Steiner quadruple system. `https://mathworld.wolfram.com/SteinerQuadrupleSystem.html`.

Appendix A SAGE Code for Computing Automorphism Groups

Here is some SAGE code using a brute-force method to calculate the strong and weak automorphism groups of a stabilizer code. Thanks to Dr. Daniel Bump for his major assistance in writing this program.

```
"""
SAGE Code for looking for automorphisms of stabilizer quantum error
correcting codes.
"""

# For reference, x,y,z = Pauli matrices
# h,s = Hadamard and Phase gates.

X = Matrix([[0,1],[1,0]])
Y = Matrix([[0,-i],[i,0]])
Z = Matrix([[1,0],[0,-1]])
H = Matrix([[1,1],[1,-1]])
S = Matrix([[1,0],[0,i]])

# The Pauli error group is defined by generators and relations:

FG.<x,y,z,j> = FreeGroup()
PE = FG/{j^4,x^2,y^2,z^2,x*y/x/y*j^2,y*z/y/z*j^2,z*x/z/x*j^2,x*j/x/j,y*j/y/j,
z*j/z/j,x*y/j/z,y*x*j/z,y*z/j/x,z*y*j/x,z*x/j/y,x*z*j/y}
PE.inject_variables()

def ca(a, split=False):
    """
    Canonical form for the Pauli Error group elements
    """
    for b in [PE.one(),x,y,z]:
        for k in [PE.one(),j,j^2,j^3]:
            if a == k*b:
                if split:
                    return [k,b]
                else:
                    return k*b
    return a
```

```
def emul(A,B,debug=False):
    """
    Multiplication for Pauli error group elements in canonical form
    """
    a0 = A[0]
    b0 = B[0]
    r0 = a0*b0
    rt = []
    for [t,u] in zip(A[1:],B[1:]):
        [p,q]=ca(t*u,split=True)
        r0 *= p
        if debug:
            print (t,u,p)
        rt.append(q)
    ret = [ca(r0)]
    for t in rt:
        ret.append(t)
    return ret

def eprod(M,n):
    """
    M is a subset of the stabilizer group S
    Returns the product of the elements of M.
    """
    ret = tuple((n+1)*[PE.one()])
    for m in M:
        ret = emul(ret,m)
    return list(ret)

def unpack(st):
    """
    Pauli error group elements may be represented by strings
    and unpacked by this function.

    sage: unpack("XYZZY")
    (1, x, y, z, z, y)
    """
    ret = [PE.one()]
    for c in st:
```

```
        if c == "X":
            ret.append(x)
        elif c == "Y":
            ret.append(y)
        elif c == "Z":
            ret.append(z)
        elif c == "I":
            ret.append(PE.one())
    return tuple(ret)

def pack(w):
    ret = ""
    for t in w[1:]:
        if t==PE.one():
            ret += "I"
        elif t==x:
            ret += "X"
        elif t==y:
            ret += "Y"
        elif t==z:
            ret += "Z"
    return ret

def Stabilizer(S):
    """
    Return the stabilizer group generated by a set of generators
    of the Pauli error group in string notation. The generators
    must commute and the generated group may not contain -I.
    """
    n = len(S[0])
    return [eprod(M,n) for M in Set(unpack(s) for s in S).subsets()]

def GenToArray(S):
    """
    Takes an array of generator strings and returns the same Pauli elements
    in standard form.
    """
    return [list(unpack(s)) for s in S]

StabTest = ["XXX", "YYI", "ZXZ"]
```

```
Stab513 = ["XZZXI","IXZZX","XIXZZ","ZXIXZ"]
Stab604 = ["IXZZXI","IIXZZX","IXIXZZ","IZXIXZ","XXXXXX","ZZZZZZ"]
Stab713 = ["IIIXXXX", "IXXIIXX", "XIXIXIX", "IIIZZZZ", "IZZIIZZ", "ZIZIZIZ"]
Stab833 = ["XIZIYZXY", "IXZZYXYI", "IZXIYYZX", "IZIYZXXY", "ZZZZZZZZ"]
Stab823 = ["XIZIYZXY", "IXZZYXYI", "IZXIYYZX", "IZZYZXZZ", "IIZIIIYX", "ZZZZZZZZ"]
Stab933 = ["XIZIYZXYI", "IXZZYXYII", "IZXIYYZXI", "IZIYZXXYI", "ZZZZZZZZI",
 "IIIIIIIIX"]
Stab1004 = ["XIIZXZXZII", "IXIIZXZXZI", "IIXIIZXZXZ", "ZIIXIIZXZX", "XZIIXIIZXZ",
 "ZXZIIXIIZX", "XZXZIIXIIZ", "ZXZXZIIXII", "IZXZXZIIXI", "IIZXZXZIIX"]
Stab1115 = ["ZZZZZZIIIII", "XXXXXXIIIII", "IIIZXYYYYXZ", "IIIXYZZZZYX",
 "ZYXIIIZYXII", "XZYIIIXZYII", "IIIZYXXYZII", "IIIXZYZXYII", "ZXYIIIZZZXY",
 "YZXIIIYYYZX"]

"""
StabTest should have Autweak=S3 and Autstrong=S2. Try using ListPerms function.
"""

def ListPerms(Gen):
    """
    Prints out permutations in weak and strong automorphism groups, as well as
    the sizes of the groups.
    Weak automorphism group is calculated first to improve efficiency.
    Input: List of strings with generators for stabilizing subgroup; e.g. Stab513
    """
    genList = GenToArray(Gen)
    C = Stabilizer(Gen)
    short_gen = [w[1:] for w in genList]
    short_code = [w[1:] for w in C]
    size = len(short_code[0])
    weakgroup = []
    stronggroup = []
    for p in Permutations(size):
        check = true
        for w in short_gen:
            if check == true:
                check = [w[p[i]-1] for i in range(size)] in short_code
            else:
                break
        if check == true:
            print(p)
```

```
        weakgroup.append(p)
print("Size of weak automorphism group: " + str(len(weakgroup)))
for p in weakgroup:
    check = true
    for w in genList:
        if check == true:
            check = ([w[0]]+[w[p[i]] for i in range(size)]) in C
        else:
            break
    if check == true:
        print(p)
        stronggroup.append(p)
print("Size of strong automorphism group: " + str(len(stronggroup)))
```