Quantum Subsystem Codes

Their Theory and Use

Nikolas P. Breuckmann - September 26, 2010

Bachelor thesis under supervision of
Prof. Dr. B. M. Terhal

Fakultät für Mathematik, Informatik und Naturwissenschaften - Fakultät 1
IQI - Institute for Quantum Information

RWTH AACHEN UNIVERSITY

Contents

Notations

$\lvert abc...\rangle$	The tensor product $\lvert a\rangle \otimes \lvert b\rangle \otimes \lvert c\rangle \otimes ...$
$\mathcal{H}$	A complex Hilbert space
$V^{\otimes n}$	The n-fold tensor product of V
$\underline{n}$	The set of natural numbers $\{1, ..., n\}$
$\mathcal{P}_n$	The Pauli group acting on n qubits
$\mathcal{G}$	The gauge group
$\mathcal{S}$	The stabilizer group
$\mathcal{L}_b$	The bare logical group
$\mathcal{Z}(G)$	The center of group G
$C(G)$	The centralizer of group G in $\mathcal{P}_n$
$\mathcal{N}(G)$	The normalizer of group G in $\mathcal{P}_n$
$\leqslant$	depending on the context: subspace of a vector space or subgroup
$M(\mathcal{S})$	The check matrix of the stabilizer code $\mathcal{S}$
$wt(P)$	The weight of the operator P
K_e	The operator induced by the edge e
$W(M)$	The operator induced by the cycle M

1 Introduction

In recent years it has been shown that by exploiting the properties of quantum physics we might be able to build computers that can outperform any computer from today. Until now this exponentially speed up has been most famously demonstrated by Peter W. Shor's factoring algorithm and Lov Grover's searching algorithm[1]. However this gain in computing power comes at a price. Since we need quantum systems to perform quantum computing we have to deal with very delicate structures. The fail-safe storage of information on a quantum computer is of course indispensable and relies on a good physical implementation. But also on a more abstract level we can fight errors by introducing error correction schemes. In fact it has been shown that if the occurrence of errors stays below a certain threshold it is possible to protect the data which makes reliable quantum computing practicable.

Chapter 2 of this thesis presents the basic ideas behind error correction on quantum computers and the first example for a error correcting quantum code is constructed along the way. Chapter 3 introduces the stabilizer formalism which is a powerful tool to describe the states inside the quantum computer. This leads to the definition of stabilizer codes which will be generalized to subsystem codes in chapter 4. In chapter 5 the geometrical subsystem codes are introduced. The Bacon-Shor code serves as an example to show how subsystem codes can improve the error correction. Afterwards the topological codes are defined and their practical advances explained. The topological subsystem codes have good properties concerning the error detection. One of the main results of this thesis will be that by mapping topological codes the error detection can be done locally.

2 General quantum error-correction

Before error correction can be discussed it is crucial to know how information is stored on a quantum computer and therefore how it can be corrupted. Just as on a classical computer from today we encode our information into bits which have two states - either 0 or 1. They are referred to as qubits and they will be written as $|0\rangle$ and $|1\rangle$. As the Dirac notation indicates these states are elements of a (2-dimensional) Hilbert space $\mathcal{H} := \langle |0\rangle, |1\rangle \rangle_{\mathbb{C}} \cong \mathbb{C}^2$. Therefore not only $|0\rangle$ and $|1\rangle$ are valid qubits, but also arbitrary linear combinations $\alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbb{C}$. Since we want to encode more information than just one bit we need to implement bit strings. This is done by composing the state spaces of the qubits

[1]Grover's algorithm "only" provides a quadratic speed up which is still noticeable for large databases.

with the tensor product according to the postulates of quantum mechanics. For example the bit string **010** is realized by the state $|010\rangle := |0\rangle \otimes |1\rangle \otimes |0\rangle \in \mathcal{H}^{\otimes 3}$.[2]

To develop quantum error correcting schemes it is of course essential to know what kind of errors we are dealing with. This question is easily answered in the classical case, where the only type of error is the flip of one or several bits in a bit string. But qubits have more inner structure than classical bits. Since they are elements of a quantum mechanical system every operation that is allowed by quantum mechanics is a candidate for an error. These *quantum operations* - also called *quantum channels* - are completely positive trace preserving linear mappings acting on the density operator of the system we encode our information in. For simplicity we will only consider errors described by actions of the Pauli group $\mathcal{P}_n$ which is generated by the the Pauli operators acting on n qubits and an overall phase:[3]

$$\mathcal{P}_n = \{\gamma P_1 \otimes ... \otimes P_n \mid \gamma \in \{\pm 1, \pm i\}, P_i \in \{I, X, Y, Z\}\} \tag{2.1}$$

The Pauli operators can be represented by the matrices from figure 1.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Figure 1: The identity and the three Pauli matrices

The weight $wt(P)$ of a Pauli group element P is the number qubits P acts on non-trivially i.e. as X, Y or Z. One can easily check that all single qubit Pauli operators either commute or anti-commute. Thus we have $[P, P'] = 0$ or $\{P, P'\} = 0$ for all $P, P' \in \mathcal{P}_n$.

If we think of $|0\rangle$ and $|1\rangle$ as the standard base vectors of $\mathbb{C}^2$ then the action of X and Z on the basis states $|0\rangle$ and $|1\rangle$ is:

$$\begin{aligned} X|0\rangle &= |1\rangle \quad & Z|0\rangle &= |0\rangle \\ X|1\rangle &= |0\rangle \quad & Z|1\rangle &= -|1\rangle \end{aligned}$$

This is why we will refer to X as bit-flip error and to Z as phase-flip error.

Since a qubit has more inner structure than a classical bit we have to deal with some difficulties which at first sight seem to make the task of error correction

[2]We will from now on only use $\mathcal{H}$ to refer to an n-fold tensor product of 2-dimensional Hilbert spaces i.e. the state space of n qubits.

[3]We will often only consider X and Z since $Y = iXZ$.

impossible. We will illustrate these problems while at the same time construct the first example for a quantum code which protects a single qubit against errors.

The key idea behind encoding data to protect it against errors is to add redundancy. A simple example would be a so called repetition code which just takes the information and makes several copies. Classically we can encode the bit **0** into **000** and the bit **1** into **111**. If only one bit-flip occurred we can still reconstruct the original bit string by majority voting e.g. **010** would be corrected to **000**. The following theorem shows that this can not be done with qubits.

Theorem 2.1 (No-cloning theorem)**.** *There is no quantum operation that takes a state $|\psi\rangle \otimes |s\rangle$ to $|\psi\rangle \otimes |\psi\rangle$ for all states $|\psi\rangle$ where $|s\rangle$ is some normalized standard pure state.*

Proof. Suppose there is a unitary U which provides us the desired mapping

$$U\left(|\psi\rangle \otimes |s\rangle\right) = |\psi\rangle \otimes |\psi\rangle$$
$$U\left(|\phi\rangle \otimes |s\rangle\right) = |\phi\rangle \otimes |\phi\rangle$$

Taking the inner product of these two equations gives us

$$\langle\psi\,|\,\phi\rangle = (\langle\psi| \otimes \langle s|)U^\dagger U(|\phi\rangle \otimes |s\rangle) = (\langle\psi| \otimes \langle\psi|)(|\phi\rangle \otimes |\phi\rangle) = (\langle\psi\,|\,\phi\rangle)^2$$

But from this follows that $\langle\psi\,|\,\phi\rangle \in \{0, 1\}$. Thus $|\psi\rangle$ and $|\phi\rangle$ are either orthogonal or identical. ∎

Therefore we can not apply this method directly to some qubit $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ because there is in general no way to encode it into the state $|\psi\rangle^{\otimes 3}$. And to make things even worse there is a second obstacle which has to do with another intrinsic feature of quantum mechanics: Since the recovery of the initial state depends on what error happened we need to make a measurement. But in quantum mechanics such a measurement disturbs our state and we might loose our encoded information.

As it turns out we are still able construct a code which protects against single qubit flips. Let us assume we have a single qubit which is either in the state $|0\rangle$ or $|1\rangle$ and two other qubits in the $|0\rangle$ state which are *ancilla qubits*. There exists a unitary evolution U (see figure 2) which has the following effect on this 3-qubit state:

$$U(|x\rangle \otimes |0\rangle \otimes |0\rangle) = U(|x\rangle \otimes |x\rangle \otimes |x\rangle) \tag{2.2}$$

where $x \in \{0, 1\}$. Since U is linear we can map $\alpha\,|0\rangle + \beta\,|1\rangle$ to $\alpha\,|000\rangle + \beta\,|111\rangle$ and thereby avoid the limitations given by theorem 2.1.

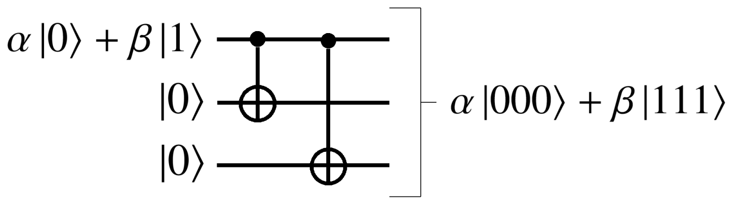
$\alpha|0\rangle + \beta|1\rangle$
$|0\rangle$
$|0\rangle$
$\alpha|000\rangle + \beta|111\rangle$

Figure 2: The quantum circuit for encoding the state $\alpha|0\rangle + \beta|1\rangle$ into the encoded state $\alpha|000\rangle + \beta|111\rangle$ by introducing two ancilla states. The unitary evolution U is realized by two CNOT gates.

To detect on which qubit an error occurred we define the following projection operators:

$$P_0 = |000\rangle\langle 000| + |111\rangle\langle 111| \tag{2.3}$$

$$P_1 = |100\rangle\langle 100| + |011\rangle\langle 011| \tag{2.4}$$

$$P_2 = |010\rangle\langle 010| + |101\rangle\langle 101| \tag{2.5}$$

$$P_3 = |001\rangle\langle 001| + |110\rangle\langle 110| \tag{2.6}$$

The projector P_0 is measured with probability one if no error happened and P_i for $i = 1, 2, 3$ is measured with probability one if the i-th qubit was flipped. It is important that each measurement always tells us exactly which error happened and that after the measurement we still have the exact same state as before. Such a measurement is called a *syndrome measurement* and the measurement result is called the *error syndrome*. Knowing the error syndrome we can do a recovery operation by flipping the according qubit back and thereby recover our original encoded state.

A phase flip error takes a qubit $\alpha|0\rangle + \beta|1\rangle$ to $\alpha|0\rangle - \beta|1\rangle$. If we choose $\alpha = \beta = \frac{1}{\sqrt{2}}$ than these two states behave under the action of Z just as $|0\rangle$ and $|1\rangle$ do under the action of X. These states are denoted $|+\rangle$ and $|-\rangle$. By performing a base change and following the same strategy as for the bit-flip code we can construct a code which protects against single phase-flip errors.

Since a overall phase has no physical meaning and $Y = iXZ$ we can now correct X-, Y- and Z-errors.

To protect the encoded qubit against both types of errors we simply concatenate both codes and get the *Shor code* which encodes a single qubit into 9 physical qubits to protect it against bit flip and phase flip errors. The codewords of the Shor code are the following:

$$|\bar{0}\rangle := \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}} \tag{2.7}$$

$$|\bar{1}\rangle := \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}} \tag{2.8}$$

The states $|\bar{0}\rangle$ and $|\bar{1}\rangle$ are the *logical states* of the code i.e. the qubit string before the encoding. Note that because quantum codes rely on the formalism of quantum mechanics they will always be linear. Therefore the *code space*[4] C of the Shor-Code is given by $C = \langle |\bar{0}\rangle, |\bar{1}\rangle\rangle_{\mathbb{C}}$.

A nice property of the Shor code is that we can see from the states how error correction works in detail: There is an inner layer which comes from the 3-bit flip code and an outer layer from the 3-bit phase flip code. In the inner layer single qubit flip errors are corrected by majority vote within each set of three. For example if in one of the three blocks the second bit flips we correct:

$$|010\rangle \pm |101\rangle \to |000\rangle \pm |111\rangle \tag{2.9}$$

The outer layer does the same majority vote with respect to the signs. Assume that the sign in the second block flipped. Then the state

$$\frac{(|000\rangle + |111\rangle)(|000\rangle - |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}} \tag{2.10}$$

is corrected to

$$\frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}}. \tag{2.11}$$

Both corrections are independent. Therefore the encoded qubit is protected from single bit *and* phase flips at the same time.

We will now see that only being able to correct errors from the Pauli group is not a restriction. Since we are now dealing with continuous states there might be continuous changes of the relative phase and not only phase-flips. Let us assume for example that we are starting with a single qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ and after some error happened our state has an additional relative phase: $E|\psi\rangle = \alpha|0\rangle + e^{i\delta}\beta|1\rangle$. To correct this error we might suspect that we would have to get to know $e^{i\delta}$ up to infinite precision which is a practically impossible task.

Fortunately the Pauli matrices and the identity matrix span $\mathbb{C}^{2\times 2}$ and thus we can decompose an error as follows:

$$E|\psi\rangle = (e_0 I + e_1 X + e_2 Y + e_3 Z)|\psi\rangle \text{ with } e_i \in \mathbb{C} \tag{2.12}$$

[4]The vector space of all possible encoded states.

Measuring the error syndrome collapses this superposition into either $|\psi\rangle$, $X|\psi\rangle$, $Y|\psi\rangle$ or $Z|\psi\rangle$ and our initial state $|\psi\rangle$ can be recovered by applying the appropriate inversion operation. Thus we only need to correct X and Z errors on each qubit to recover our initial state since $Y = iXZ$ and a global phase shift has no physical effect.

3 The stabilzer formalism

In this chapter we will introduce an alternative description of quantum codes based on the *stabilizer formalism*. In more complicated codes the states of the code space may become quite complex and unpleasant to deal with since the number of coefficients grows exponentially in the number of qubits. Instead of bothering with the states from $\mathcal{H}$ directly we rely on the use of group theory to describe them and in particular on actions of the Pauli group on the state space of the qubits $\mathcal{H}$. We will from now on always assume that we are dealing with *n physical qubits* i.e. the qubits we will encode our data on.

Let P be an operator from $\mathcal{P}_n$. The set of all states which are invariant under the action of P

$$V_P := \{|\psi\rangle \in \mathcal{H} \mid P|\psi\rangle = |\psi\rangle\} \tag{3.1}$$

is a subspace of $\mathcal{H}$. Equivalently we can think of it as the eigenspace to the eigenvalue $+1$ of P.

For example for $n = 3$ the operator Z_1Z_2 stabilizes the space

$$\langle |00x\rangle, |11x\rangle \mid x \in \{0,1\}\rangle_{\mathbb{C}}. \tag{3.2}$$

To stay within the formalism of group action, which allows us to analyze the action of a whole group on a vector space, this should be thought of as the subspace stabilized by the group generated by P namely $\langle P\rangle$. This can be taken further by introducing new generators into the so called *stabilizer group* $\mathcal{S}$. The stabilizer group of the previous example can be extended by introducing the operator Z_2Z_3. The stabilizer group becomes

$$\mathcal{S} = \langle Z_1Z_2, Z_2Z_3\rangle = \{I, Z_1Z_2, Z_2Z_3, Z_1Z_3\} \tag{3.3}$$

and the stabilized vector space is reduced to

$$V_{\mathcal{S}} := \langle |000\rangle, |111\rangle\rangle_{\mathbb{C}}. \tag{3.4}$$

The interesting result here is that we obtained the code space of the 3-qubit bit-flip code from the previous section. This means that we are able to describe the code space C with the help of the stabilizer group $\mathcal{S}$.

To see how we can also describe the error detection from the bit-flip code over the stabilizer elements we assume that our system is in one of the stabilizer states, say $|000\rangle$. Now a single bit-flip error occurs on the first qubit, i.e. a X_1 operator acts on the state. The resulting state $|100\rangle$ then is a -1 eigenstate of Z_1Z_2 but it still is in the $+1$ eigenspace of Z_2Z_3. This comes from the fact that X_1 anti-commutes with Z_1Z_2 but commutes with Z_2Z_3. Hence measuring the stabilizer generators is equivalent to do parity checks (see table 1).

(Z_1Z_2, Z_2Z_3)	Type of X-error
$(1, 1)$	no error
$(-1, 1)$	error on the first qubit
$(-1, -1)$	error on the second qubit
$(1, -1)$	error on the third qubit

Table 1: Possible results of the syndrome measurement for the bit-flip code

This procedure is similar to the error detection for the bit-flip code because measuring the generators of the stabilizers gives us the error syndrome for a single qubit-flip. Note that measuring the product of both stabilizers also does not give us any information about the coefficients of the state.

The stabilizer formulation is not only possible for the simple qubit-flip code but for a wide range of quantum error correcting codes. The main strategy stays the same: We introduce a stabilizer group $\mathcal{S}$ and regard the stabilized subspace $V_\mathcal{S}$ as our code space. If an error occurs which anti-commutes with some of the stabilizer generators then our state is taken to the -1 eigenspace of these generators. Measuring the stabilizer generators S_i then gives us a distinctive error syndrome $m_i \in \{\pm 1\}$ from which we can deduce the type of error. Recovery can then be done by applying the according inverse operation.

Properties of stabilizer groups

Before we investigate further how error detection and correction works in detail we will first have a look at some theory regarding the stabilizer group $\mathcal{S}$. From what we have seen so far it is not clear if any subgroup of $\mathcal{P}_n$ qualifies to be a stabilizer group. We also need to know how the choice of stabilizer generators effects the properties of the code such as the number of qubits which can be encoded and against what errors it protects our encoded data.

Let us first analyze how the choice of $\mathcal{S}$ effects the stabilized vector space $V_\mathcal{S}$. Because we are interested in encoding information in $V_\mathcal{S}$ we want $V_\mathcal{S}$ to be non-trivial. A necessary and sufficient condition for this is given in the following theorem:

Theorem 3.1. *For a group $\mathcal{S} \leq \mathcal{P}_n$ to stabilize a non-trivial vector space it has to be abelian and $-I$ can not be one of its elements.*

Proof. Assume there are two non-commuting elements A and B in $\mathcal{S}$ then we have:

$$|\psi\rangle = AB\,|\psi\rangle = -BA\,|\psi\rangle = -\,|\psi\rangle \tag{3.5}$$

From this follows $|\psi\rangle = 0$ and $V_{\mathcal{S}} = \{0\}$.
If $-I \in \mathcal{S}$ a similar argument holds because we get the equality:

$$-I\,|\psi\rangle = |\psi\rangle \tag{3.6}$$

■

It is easy to see that therefore $\pm iI \notin \mathcal{S}$ as well. We also get the following corollary.

Corollary 3.2. *All elements of $\mathcal{S}$ square to the identity I. Therefore they have eigenvalues* ± 1.

At this point it is convenient to introduce the so called *symplectic notation*, because it allows us to prove some following theorems with the help of linear algebra. The main observation is that multiplication in the Pauli group can be described by the addition of exponents for each X_i and Z_i. Thus we can use the vector space $\mathbf{Z}_2^{2n}$ to describe multiplications in $\mathcal{P}_n$.[5]

Definition 3.3. *Let $r : \mathcal{P}_n \rightarrow \mathbf{Z}_2^{2n}$ be a function that maps an operator P on a row-vector v by setting the j-th entry of v to 1 if P acts on the j-th qubit as X and setting the n+j-th entry of v to 1 if P acts on the j-th qubit as Z. A presence of a 1 on both sides indicates that P acts as Y and a 0 that P acts as the identity.*

For example if $n = 4$ and we take $X_1 Y_2 Z_3 \in \mathcal{P}_4$ then

$$r(X_1 Y_2 Z_3) = (1, 1, 0, 0 \mid 0, 1, 1, 0). \tag{3.7}$$

The mapping r is a homomorphism of groups[6] with $ker(r) = \langle iI \rangle$, thus $r(P)$ does not contain any information about the multiplicative factor of P. If we have a set of operators $\{P_1, ..., P_l\} \subseteq \mathcal{P}_n$ then we will call these operators *linearly independent* if $\{r(P_1), ..., r(P_l)\}$ is a linearly independent set of vectors. The mapping r establishes a close correspondence between stabilizer groups and subspaces of $\mathbf{Z}_2^{2n}$.

With the symplectic notation there is an easy way to check whether a set of generators for a stabilizer group is independent or not.

[5] $\mathbf{Z}_2$ denotes the field with two elements.
[6] When thinking of $\mathbf{Z}_2^{2n}$ as an additive group.

Lemma 3.4. *Let $S_1, ..., S_r$ be the generators of a stabilizer group $\mathcal{S}$. Then the operators $S_1, ..., S_r$ are independent if and only if $r(S_1), ..., r(S_r)$ are linearly independent vectors.*

Proof. We proof the contrapositive. Assume that $r(S_1), ..., r(S_r)$ are linearly dependent. Then there are coefficients $\alpha_1, ..., \alpha_{r-1} \in \mathbf{Z}_2$ such that

$$r(S_r) = \sum_{i \in \underline{r-1}} \alpha_i \cdot r(S_i)$$

This is equivalent to saying that $\prod_{i \in \underline{r}} S_i^{\alpha_i}$ equals S_r modulo some element from the kernel of $r|_{\mathcal{S}}$. Since $ker(r|_{\mathcal{S}}) = \{I\}$ we have

$$S_r = \prod_{i \in \underline{r-1}} S_i^{\alpha_i}$$

which shows that $S_1, ..., S_r$ are dependent. ■

The symplectic notation can also be used to check if two elements of $\mathcal{P}_n$ commute. This is done by a symplectic inner product defined by the following matrix:

$$\Lambda := \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} \tag{3.8}$$

Where the off-diagonals are $n \times n$ identity matrices. Two elements of the Pauli group P and P' commute if and only if they act as different elements of $\{X, Y, Z\}$ on an even number of qubits. This condition is easily seen to be equivalent to

$$r(P)\Lambda r(P')^{tr} = 0. \tag{3.9}$$

A useful way of presenting the generators of a stabilizer code is the so called *check matrix* which is defined over the mapping r.

Definition 3.5. *Suppose $\mathcal{S}$ is generated by $S_1, ..., S_r$. Then the rows of the check matrix $M(\mathcal{S})$ are defined as the rows of $\mathbf{Z}_2^{2n}$ the stabilizer generators are mapped to, i.e.*

$$M(\mathcal{S})_{i,-} := r(S_i) \text{ for all } i \in \underline{r}. \tag{3.10}$$

Thus we have $M(\mathcal{S}) \in \mathbf{Z}_2^{r \times 2n}$.

Lemma 3.6. *Let $\mathcal{S}$ be a subgroup of $\mathcal{P}_n$ generated by linearly independent generators $S_1, ..., S_r$. Then for all $i \in \underline{r}$ there exists a $P \in \mathcal{P}_n$ such that $PS_iP^\dagger = -S_i$ and $PS_jP^\dagger = S_j$ for all $j \neq i$.*

Proof. We want P to be an element of $\mathcal{P}_n$ which commutes with all generators of $\mathcal{S}$ except for S_i which can be expressed as

$$r(S_j)\Lambda r(P) = \delta_{ij} \text{ for all } j \in \underline{r}$$

We can now use the mapping r and the check matrix $M(\mathcal{S})$ to show that such an operator P exists since $r(P)^{tr}$ must be a solution of the following system of linear equations:

$$M(\mathcal{S})\Lambda x = e_i$$

where $x \in Z_2^{2n}$. Since the generators are linearly independent $M(\mathcal{S})$ has full rank and $M(\mathcal{S})\Lambda$ is invertible. Because r is surjective there exists a $P \in \mathcal{P}_n$ such that

$$r(P) = (M(\mathcal{S})\Lambda)^{-1}e_i$$

which concludes the proof. ■

Lemma 3.6 provides the tool to prove that for each independent generator, which is added to the stabilizer $\mathcal{S}$, the dimension of the stabilized space $V_{\mathcal{S}}$ is divided by 2. This result is intuitively clear since each stabilizer has a +1 and -1 eigenspace which cuts the total Hilbert space into two subspaces of the same dimension.

Theorem 3.7. *Let $\mathcal{S}$ be a stabilizer group operating on n qubits and with independent generators $S_1, ..., S_r$. The vector space stabilized by $\mathcal{S}$ has the dimension 2^{n-r}.*

Proof. Let $x = (x_1, ..., x_r)^{tr}$ be an element of Z_2^r. We define the operator

$$P^x := \frac{\prod_{j\in\underline{r}}(I + (-1)^{x_j}S_j)}{2^r} \tag{3.11}$$

Notice that $P^{(0,...,0)}$ is the projector onto $V_{\mathcal{S}}$. We have shown in lemma 3.6 that there exists a $T_x \in \mathcal{P}_n$ such that

$$T_x P^{(0,...,0)} T_x^\dagger = P^x \tag{3.12}$$

Therefore the dimension of the +1-eigenspace of P^x is the same for all $x \in Z_2^r$ namely $dim(V_{\mathcal{S}})$. It is also easy to see that all these eigenspaces are orthogonal and since

$$\sum_{x\in Z_2^r} P^x = \frac{1}{2^r}\sum_{x\in Z_2^r}\prod_{j\in\underline{r}}(I + (-1)^{x_j}S_j) = \frac{1}{2^r}\sum_{x\in Z_2^r} I = I \tag{3.13}$$

we can decompose the 2^n-dimensional Hilbert space $\mathcal{H}$ into 2^r orthogonal vector spaces of the dimension $dim(V_{\mathcal{S}})$. Hence $dim(V_{\mathcal{S}}) = 2^{n-r}$. ■

Stabilizer codes

From theorem 3.7 follows that the code space is isomorphic to a Hilbert space with dimension 2^{n-r} or equivalently to the Hilbert space of $n-r$ qubits. The qubits of this isomorphic Hilbert space are called *virtual qubits* since they have no physical representation. Nevertheless they play an important role since they are be used to carry the encoded information. From now on we will use the letter k to refer to the number of virtual qubits i.e. the number of qubits we can encode. We can also find operators in $\mathcal{P}_n$ which act as Pauli X and Z on virtual qubits. They are called *logical operators*. The following theorem is taken from [10] without a proof:

Theorem 3.8. *Let $\mathcal{S}$ be a stabilizer group in $\mathcal{P}_n$ and $S_1, ..., S_r$ independent generators of $\mathcal{S}$. Then there are independent operators $\overline{X}_1, ..., \overline{X}_{n-r}, \overline{Z}_1, ..., \overline{Z}_{n-r} \in \mathcal{P}_n$ such that*

$$C(\mathcal{S}) = \langle iI, S_1, ..., S_r, \overline{X}_1, ..., \overline{X}_{n-r}, \overline{Z}_1, ..., \overline{Z}_{n-r} \rangle \tag{3.14}$$

and $\overline{X}_i^2 = \overline{Z}_j^2 = I$ for all $i, j \in \underline{n-r}$. Furthermore they satisfy $[S_l, \overline{X}_i] = [S_l, \overline{Z}_j] = 0$ and $[\overline{X}_i, \overline{Z}_j] = \delta_{i,j}$ for all $l \in \underline{r}$ and $i, j \in \underline{n-r}$.

Theorem 3.8 has some interesting implications concerning the Hilbert space of the physical qubits. Because $\{S_1, ..., S_r, \overline{Z}_1, ..., \overline{Z}_{n-r}\}$ is a complete set of commuting observables there is a basis of eigenvectors for $\mathcal{H}$:

$$S_i \left| s_1, ..., s_r, z_1, ..., z_{n-r} \right\rangle = (-1)^{s_i} \left| s_1, ..., s_r, z_1, ..., z_{n-r} \right\rangle \tag{3.15}$$

$$Z_i \left| s_1, ..., s_r, z_1, ..., z_{n-r} \right\rangle = (-1)^{z_i} \left| s_1, ..., s_r, z_1, ..., z_{n-r} \right\rangle \tag{3.16}$$

Note that the s_i and z_i are not the eigenvalues which could be ± 1 but the according exponents of -1. This notation is convenient since

$$Z \left| 0 \right\rangle = (-1)^0 \left| 0 \right\rangle \tag{3.17}$$

and

$$Z \left| 1 \right\rangle = (-1)^1 \left| 1 \right\rangle . \tag{3.18}$$

This shows that the physical qubit space $\mathcal{H}$ can be decomposed into a tensor product of the 2^{n-r}-dimensional logical space $\mathcal{H}_L$ and the 2^r-dimensional space $\mathcal{H}_{syn}$ in which the information about the syndrome measurement is kept.

We want to use the states stabilized by $\mathcal{S}$ to encode information. Therefore the states of the code space C have the following form:

$$\left| 0, ..., 0, z_1, ..., z_{n-r} \right\rangle =: \overline{\left| z_1, ..., z_{n-r} \right\rangle} \tag{3.19}$$

on which the logical operators act in the following way:

$$\overline{X}_i\, \overline{|z_1, ..., z_{n-r}\rangle} = \overline{|z_1, ..., z_i + 1 \bmod 2, ..., z_{n-r}\rangle} \tag{3.20}$$
$$\overline{Z}_i\, \overline{|z_1, ..., z_{n-r}\rangle} = (-1)^{z_i}\, \overline{|z_1, ..., z_{n-r}\rangle} \tag{3.21}$$

What we have accomplished is to split up the Hilbert space of physical qubits into a direct sum of different subspaces where C is the subspace of $\mathcal{H}$ which is stabilized by a stabilizer group $\mathcal{S}$. The strategy to protect the encoded information is that errors take the encoded state from C into another subspace. We will discuss the general error detection and correction schemes in the following chapter for a more general class of quantum codes but we can already give a indicator for the error correction qualities of stabilizer codes called *distance*. The distance tells us how many single qubit changes are needed to get from one codeword to another. This notion can be captured with the help of the logical operators:

Definition 3.9. *The distance of a stabilizer code is defined as the minimum weight of all elements of* $C(\mathcal{S}) - \mathcal{S}$ *i.e.*

$$d := \min_{P \in C(\mathcal{S}) - \mathcal{S}} wt(P). \tag{3.22}$$

The Shor code in the stabilizer formalism

In chapter 2 we introduced the Shor code which encodes a single qubit into 9 physical qubits and protects it against single qubit errors. We will now translate this code in the stabilizer formalism. From theorem 3.7 follows that the stabilizer formulation of the Shor code should have 8 generators. In fact we can find a set of independent stabilizer generators by thinking back to the parity checks of the bit- and phase-flip codes. The parity of the bits is fixed by operators of the form Z_iZ_{i+1} whereas the phases are fixed by the stabilizers $X_1X_2X_3X_4X_5X_6$ and $X_4X_5X_6X_7X_8X_9$.

Having the states of the Shor code in mind (see eqations (2.7) and (2.8)) we observe that the operator $X_1X_2X_3X_4X_5X_6X_7X_8X_9$ takes the logical state $|\overline{1}\rangle$ to $-|\overline{1}\rangle$ and leaves $|\overline{0}\rangle$ invariant. Consequently

$$\overline{Z} := X_1X_2X_3X_4X_5X_6X_7X_8X_9 \tag{3.23}$$

is the logical Z operator. The logical X operator is given by

$$\overline{X} := Z_1Z_2Z_3Z_4Z_5Z_6Z_7Z_8Z_9. \tag{3.24}$$

Notice that the logical operators have nothing in common with the X and Z operators on the physical qubits. In other codes the logical operators might have an even more complicated form.

We will from now on refer to stabilizer codes which encode k logical qubits onto n physical qubits with distance d as $[[n, k, d]]$-codes. For example the Shor code is a $[[9, 1, 3]]$-code.

$$Z \otimes Z \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I$$
$$I \otimes Z \otimes Z \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I$$

$$I \otimes I \otimes I \otimes Z \otimes Z \otimes I \otimes I \otimes I \otimes I$$
$$I \otimes I \otimes I \otimes I \otimes Z \otimes Z \otimes I \otimes I \otimes I$$

$$I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes Z \otimes Z \otimes I$$
$$I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes Z \otimes Z$$

$$X \otimes X \otimes X \otimes X \otimes X \otimes X \otimes I \otimes I \otimes I$$
$$I \otimes I \otimes I \otimes X \otimes X \otimes X \otimes X \otimes X \otimes X$$

Figure 3: The eight generators for the Shor code. The first three pairs of generators check the parity of the the qubits in the three blocks and the last two check the phase between the three blocks.

4 Subsystem codes

The notion of stabilizer codes can be extended into a more general form called *subsystem codes*. In this chapter we will first discuss the defining properties of the underlying formalism and how it relates to the stabilizer codes from the preceding chapter. Later we will discuss how these new codes can be constructed.

Definition of subsystem codes

Until now we decomposed the Hilbert space of the qubits into a direct sum $\mathcal{H} = C \oplus C^{\perp}$ where C is the code space in which the information is encoded. Subsystem codes do not use the complete subspace C. Instead the code space has a subsystem structure i.e. C is a tensor product of subspaces or subsystems[7] which we will call A and B for the rest of this chapter. The complete physical Hilbert space $\mathcal{H}$ then looks as follows:

$$\mathcal{H} = A \otimes B \oplus C^{\perp} \tag{4.1}$$

We will only use the subsystem A to store information whereas errors on B will be ignored and therefore contribute gauge degrees of freedom. One might argue if it would not suffice to take a smaller code space to obtain the same result, but we will see that this approach brings with it some advantages not only for the theory but also for practical realizations.

[7]Hence the name.

The decomposition in (4.1) is the most general case to analyze. If we wanted to store information somewhere else within $\mathcal{H}$ we would have to decompose A further into $A \otimes A'$ or $A \oplus A'$ and store data only in A. Then we can define $B' := A' \otimes B$ respectively $C^{\perp} := A' \otimes B \oplus C^{\perp}$ and we obtain the same structure for $\mathcal{H}$ as in (4.1).

The mathematical description of the code space C is equivalent to the one of stabilizer codes i.e. in subsystem codes we also have a stabilizer group $\mathcal{S}$ such that

$$C = \{|\psi\rangle \in \mathcal{H} \mid S\,|\psi\rangle = |\psi\rangle \text{ for all } S \in \mathcal{S}\}. \tag{4.2}$$

To the definition of a subsystem code also belong two other groups acting on the code space which we call the *logical group* $\mathcal{L}_b$ and the *gauge group* $\mathcal{G}$. Both groups are subgroups of $\mathcal{P}_n$ and $\mathcal{S}$ will always be a subgroup of $\mathcal{G}$ (see the section on the construction of subsystem codes). Moreover do the operators of $\mathcal{L}_b$ and $\mathcal{G}$ commute.

$$[G, L] = 0 \text{ for all } G \in \mathcal{G}, L \in \mathcal{L}_b \tag{4.3}$$

This property of $\mathcal{L}_b$ and $\mathcal{G}$ is sufficient to induce the desired tensor structure on C (see [5]). That means that we will define the subsystems A and B such that operators from $\mathcal{L}_b$ only act non-trivially on A and operators from $\mathcal{G}$ only act non-trivially on B. The logical group $\mathcal{L}_b$ is equivalent to the logical group known from stabilizer codes. That means that $\mathcal{L}_b$ is generated by operators which act as Pauli X and Z on the virtual qubits in A. The novelty in subsystem codes in contrast to stabilizer codes is the gauge group $\mathcal{G}$. The actions of the gauge group on C are also called *gauge transformations* and since G is a group they induce an equivalence relation on C. Two states $|\psi\rangle, |\psi'\rangle \in C$ will be regarded as equivalent $|\psi'\rangle \sim |\psi\rangle$ if there is a $G \in \mathcal{G}$ such that $|\psi'\rangle = G\,|\psi\rangle$. Since equivalent states carry by definition the same information we can expand our notion of logical operators. Suppose $G \in \mathcal{G}$ and $P \in \mathcal{L}_b$ then the operator $P \otimes G$ manipulates our data in the same way as P. Hence P and $P \otimes G$ are in some sense also equivalent. We will refer to operators from $\mathcal{L}_b$ which do not perform any gauge transformations as *bare logical operators*. Accordingly products of operators from $\mathcal{L}_b$ and $\mathcal{G} - \{I\}$ will be called *dressed logical operators*.

Construction of subsystem codes

Following [4] we construct a subsystem code by choosing $2n$ elements of $\mathcal{P}_n$ called X_i and Z_i with $i \in \underline{n}$. As the notation indicates we want them to follow the same commutation rules as X_i and Z_i

$$[X_i, Z_j] = 2X_i Z_j \delta_{i,j}. \tag{4.4}$$

Keep in mind that this is the only restriction we set. In particular the primed operators may act non-trivially on several physical qubits. However we can imagine them acting on *virtual qubits*. To construct the stabilizer group we choose $r < n$ and define $\mathcal{S} := \{Z'_1, ..., Z'_r\}$. This group is abelian and does not contain $-I$. Hence by theorem 3.1 we can use it as a stabilizer group which defines a code space with $n - r = k$-qubits (see theorem 3.7). We will from now on refer to Z'_i as S_i for all $i \in \underline{r}$. It is easy to see that the centralizer of $\mathcal{S}$ is

$$C(\mathcal{S}) = \langle iI, S_1, ..., S_r, X'_{r+1}, ..., X'_n, Z_{r+1}, ..., Z_n \rangle \tag{4.5}$$

The gauge group $\mathcal{G}$ should leave the code space C invariant. Hence we put iI and all elements of $\mathcal{S}$ into $\mathcal{G}$. Then we pick a number q which defines the number of gauge qubits or equivalently the number of gauge degrees of freedom. Of course it only makes sense to choose q such that $r + q < n$. Thus the gauge group $\mathcal{G}$ can be generated by the stabilizer group $\mathcal{S}$, q pairs of primed X and Z operators X'_j and Z'_j with $j > r$ and the phase iI. We can w.l.o.g. choose the pairs X_i and Z_i with i from $r + 1$ to $r + q$ and get

$$\mathcal{G} := \langle iI, \mathcal{S}, X'_{r+1}, ..., X'_{r+q}, Z'_{r+1}, ..., Z'_{r+q} \rangle. \tag{4.6}$$

The group of bare logical operators $\mathcal{L}_b$ is generated by the left $n - r - q = k$ pairs of X and Z operators.

$$\mathcal{L}_b = \langle X'_{r+q+1}, ..., X'_n, Z'_{r+q+1}, ..., Z'_n \rangle \tag{4.7}$$

Since we have $[\mathcal{G}, \mathcal{L}_b] = 0$ we have split up C into a tensor product of the 2^k dimensional subsystem A and the 2^q dimensional subsystem B. This result is obvious if we think back to the virtual qubits. What we essentially did was taking the logical operators of q virtual qubits to define the gauge transformations and kept the other k qubits as logical qubits. Note that the non-trivial bare operators are the elements of $C(\mathcal{G}) - \mathcal{G}$ whereas the dressed operators are elements of $C(\mathcal{S}) - \mathcal{G}$ and that the stabilizer group $\mathcal{S}$ is up to phase factors the center of the gauge group, i.e. $\mathcal{Z}(\mathcal{G}) := \mathcal{G} \cap C(\mathcal{G}) = \langle iI, \mathcal{S} \rangle$. Hence the subsystem code is already uniquely defined by $\mathcal{G}$.

Error detection and correction

Since we do not care about errors on the subsystem B we have to reconsider the error correcting conditions. Then for an error to be correctable we would want it to do only gauge transformations on our state. Suppose P is the projector onto the code space. Then the error correcting condition is given by

$$PE_aE_bP = I_A \otimes G_B^{a,b} \text{ for all a,b} \tag{4.8}$$

Note that the gauge transformation may depend on the error. In [6] Kribs et al. proved that this condition is necessary for error correction and in [7] Nielsen and Poulin proved that this condition is also sufficient.

Theorem 4.1. *Let $\mathcal{G}$ be a $\mathcal{S}$ be the stabilizer subgroup and $C \leq \mathcal{H}$ the stabilized code space. Suppose $\{E_i \mid i \in J\}$ is a set of operators from $\mathcal{P}_n$. Then $\{E_i \mid i \in J\}$ is a correctable set of errors for the subsystem code $\mathcal{G}$ if and only if $E_i^\dagger E_j \notin C(\mathcal{S}) - \mathcal{G}$ for all $i, j \in J$.*

Proof. Suppose P is the projector from $\mathcal{H}$ onto C. For some fixed $i, j \in J$ we have either $E_i^\dagger E_j \in \mathcal{P}_n - C(\mathcal{S})$, $E_i^\dagger E_j \in C(\mathcal{S}) - \mathcal{G}$ or $E_i^\dagger E_j \in \mathcal{G}$.
Consider the first case. There must be a $S_1 \in \mathcal{S}$ which anti-commutes with $E_i^\dagger E_j$. We can find elements from $\mathcal{S}$ such that $\mathcal{S} = \langle S_1, ..., S_r \rangle$. This allows us to write P as

$$P = \frac{\prod_{l=1}^r (I + S_l)}{2^r} \tag{4.9}$$

from which follows that

$$PE_i^\dagger E_j P = P(I - S_1) E_i^\dagger E_j \frac{\prod_{l=2}^r (I + S_l)}{2^r}. \tag{4.10}$$

Since $(I + S_1)(I - S_1) = 0$ we get $P(I - S_1) = 0$ and therefore $PE_i^\dagger E_j P = 0$ which satisfies the error correction conditions.
For the second case we observe that

$$C(\mathcal{S}) - \mathcal{G} \cong \mathcal{L} - \{I\} \times \mathcal{G}. \tag{4.11}$$

Hence $E_i^\dagger E_j = L_A \otimes G$ with some non-trivial operator on A L_A. Therfore we can not correct errors from $C(\mathcal{S}) - \mathcal{G}$.
Finally for $E_i^\dagger E_j \in \mathcal{G}$ the error is only a gauge transformation which only takes us to an equivalent state. ∎

Theorem 4.1 divides all possible errors from $\mathcal{P}_n$ into three categories:

- $E_a E_b \in \mathcal{P}_n - C(\mathcal{S})$ - These errors anti-commute with at least one element of the stabilizer.
- $E_a E_b \in C(\mathcal{S}) - \mathcal{G}$ - Errors from this group commute with all stabilizers ($E_a E_b \in C(\mathcal{S})$) and act non-trivially on A ($E_a E_b \notin \mathcal{G}$).
- $E_a E_b \in G$ - These errors only perform gauge transformations.

The definition for the distance for subsystem codes is similar to the one given in 3.9 except that not only elements of $\mathcal{S}$ are not harmful to states of the code space but also elements of $\mathcal{G}$ only map to states in C which we consider as equivalent. Thus the distance for subsystem codes is defined as

$$d := \min_{P \in C(\mathcal{S}) - \mathcal{G}} wt(P) \tag{4.12}$$

We will refer to a subsystem code that encodes k logical qubits into n physical qubits with distance d as a $[[n, k, q, d]]$ subsystem code.

Every subsystem code can be turned into a stabilizer code by extending the stabilizer group with Z operators acting on the gauge qubits[8] and thereby stabilize all gauge bits. The reverse is true as well i.e. turning a $[[n, k, d]]$ stabilizer code into a $[[n, k, q, d]]$ subsystem code. Though it is not always clear what the largest possible value for q is.

The Shor-subsystem code

Let us have a look at the Shor-code again. The parity checks defined by the first 6 stabilizer generators (see figure 3) can be united into two stabilizers: We are

$$\begin{gathered}
Z \otimes Z \otimes I \otimes Z \otimes Z \otimes I \otimes Z \otimes Z \otimes I \\
I \otimes Z \otimes Z \otimes I \otimes Z \otimes Z \otimes I \otimes Z \otimes Z \\
\\
X \otimes X \otimes X \otimes X \otimes X \otimes X \otimes I \otimes I \otimes I \\
I \otimes I \otimes I \otimes X \otimes X \otimes X \otimes X \otimes X \otimes X
\end{gathered}$$

Figure 4: The reduced set of stabilizers for the Shor code.

now checking the parity of 6 physical qubits at once which leads to less syndrome measurements. By doing this our stabilized code space has become bigger. Before we only had 1 qubit which lived in a 2-dimensional Hilbert space. By reducing the number of stabilizer generators the stabilized space is now 2^5-dimensional which means that there are 5 virtual qubits. But by doing so there are now errors which can not be detected anymore and thus not all of the qubits are protected against errors. We can find 4 pairs of anti-commuting operators (see figure 4) which generate the errors which now can not be detected anymore. These are the generators of the gauge group which operates on 4 gauge qubits leaving one qubit which is protected just as in the known stabilizer formulation.

[8]Or equivalently the X operators.

$$I \otimes Z \otimes Z \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I$$
$$I \otimes I \otimes X \otimes I \otimes I \otimes I \otimes I \otimes I \otimes X$$

$$I \otimes I \otimes I \otimes I \otimes Z \otimes Z \otimes I \otimes I \otimes I$$
$$I \otimes I \otimes I \otimes I \otimes I \otimes X \otimes I \otimes I \otimes X$$

$$Z \otimes Z \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I$$
$$X \otimes I \otimes I \otimes I \otimes I \otimes I \otimes X \otimes I \otimes I$$

$$I \otimes I \otimes I \otimes Z \otimes Z \otimes I \otimes I \otimes I \otimes I$$
$$I \otimes I \otimes I \otimes X \otimes I \otimes I \otimes X \otimes I \otimes I$$

Figure 5: Gauge operators of the $[[9,1,4,3]]$ Shor-code

Subsystem codes compared to stabilizer codes

In conclusion does the generalization into subsystem codes provide an advantage over stabilizer codes. By introducing the gauge transformations we archived passive fault tolerance. We have also seen that we can reduce the number of stabilizer which decreases the complexity of the active error correction procedure. Another advantage follows from the fact that the stabilizers can be deconstructed into gauge operators. This leads to the possibility of doing a syndrome measurement by only measuring gauge operators. This will be further analyzed in the next chapter.

5 Geometrical subystem codes

We continue with codes where we envision the qubits in a certain position in a space. This approach has two advantages: First the operators of the code can be visualized which makes it easier and more intuitive to work with. The second advantage is that the codes get a more physical interpretation since the operators correspond to interactions between neighboring qubits. Therefore locality will become a new property we can analyze. We say a set of operators acts *spatially local* if each operator acts only on a constant number of qubits located within constant distance from each other. A good code should have local stabilizer generators to simplify the syndrome measurement and non-local logical operators because non-local interactions are less likely to happen.

In this chapter we will consider two examples for geometrical codes. At first we will consider a code which is related to the Shor code called the Bacon-Shor code. It is a subsystem code defined on a square lattice with qubits on each site.

Afterwards we will define the family of topological subsystem codes. We will introduce a way to construct these codes which was provided in [2] and explicitly construct the square-octagon code as an example. At the end of this chapter we will provide a theorem from which follows that syndrome measurement using only local operators is always possible.

The Bacon-Shor code

The Bacon-Shor code is a $[[n^2, 1, (n-1)^2, n]]$ subsystem code defined on a $n \times n$-lattice of qubits. It was introduced by Dave Bacon in [12] and will be referred to as $C_{BS}^{(n)}$. The code space is stabilized by operators which act as X on two neighboring rows and as Z on two neighboring columns of qubits. More formally we have a stabilzer group

$$\mathcal{S} := \langle X_{i,-}X_{i+1,-}; Z_{-,i}Z_{-,i+1} \mid i \in \underline{n-1} \rangle \tag{5.1}$$

where $P_{i,-}$ means that the operator acts as P on each qubit in the i-th row of the lattice (equivalently $P_{-,i}$ for columns). Note that all stabilizers commute and $-I \notin \mathcal{S}$ so that we have a proper stabilizer group. We have $r = 2(n-1)$ independent generators of $\mathcal{S}$ and therefore the code space C is $2^{n^2-2(n-1)} = 2^{(n-1)^2+1}$ dimensional. Thus we have $(n-1)^2 + 1$ virtual qubits.

We define $\overline{X} := X_{1,-}$ and $\overline{Z} := Z_{-,1}$ as the logical operators for one of the logical qubits. The gauge group $\mathcal{G}$ only acts on the left $(n-1)^2$ qubits and is generated by two-qubit operators of the form $X_{j,i}X_{j+1,i}$ respectively $Z_{i,j}Z_{i,j+1}$ for all $i \in \underline{n}$ and $j \in \underline{n-1}$. Note that all of the gauge group operators commute with the logical operators for the encoded qubit. The codespace is decomposed into $C = A \otimes B$ where A is two-dimensional and B is $2^{(n-1)^2}$-dimensional.

From the choice of the logical operators for A follows that the distance of the code is n. The error recovery in A works in the same way as introduced in section 4. We can measure $X_{i,-}X_{i+1,-}$ to correct phase-flip errors on up to $\left\lfloor \frac{n}{2} \right\rfloor$ rows respectively $Z_{-,i}Z_{-,i+1}$ to correct bit-flip errors on up to $\left\lfloor \frac{n}{2} \right\rfloor$ columns. This seems like a lot of effort for the protection of just one qubit but note that only the parity of the physical qubits is relevant since errors on two qubits in one row (for Z errors) or one column (for X errors) are elements of the gauge group $\mathcal{G}$. These errors only affect the virtual gauge qubits from B and not the protected qubit from A.

But now we have a problem concerning the active error correction: The stabilizer generators which have to be measured grow with the size of the lattice. This leads to a practical problem since it is harder to do measurements with many qubits involved. Fortunately each stabilizer is composed of gauge operators

$$X_{i,-}X_{i+1,-} = \bigotimes_{k \in \underline{n}} X_{i,k}X_{i+1,k} \tag{5.2}$$

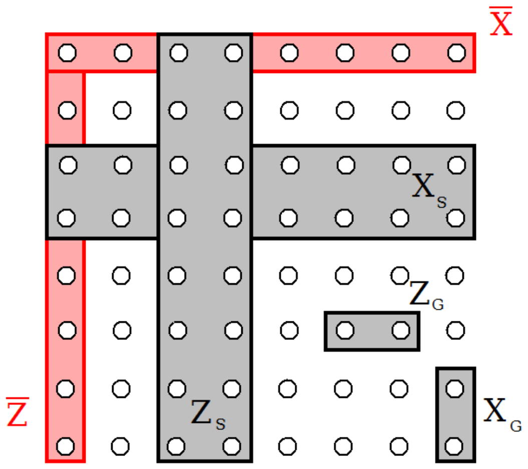

Figure 6: The Bacon Shor code $C_{BS}^{(8)}$ encoding a single qubit into 64 physical qubits on a square lattice.

and

$$Z_{-,j}Z_{-,j+1} = \bigotimes_{k\in\underline{n}} Z_{k,j}Z_{k,j+1}. \tag{5.3}$$

Since all of these factors commute they can all be measured separately. The product of all 'gauge measurements' then gives us the eigenvalue of the stabilizer generator. The measurement becomes easier because the gauge generators only act non-trivially on two qubits. Thus we will only need one $|0\rangle$ or $|+\rangle$ ancillary state to realize a syndrome measurement (see figure 7).[9]

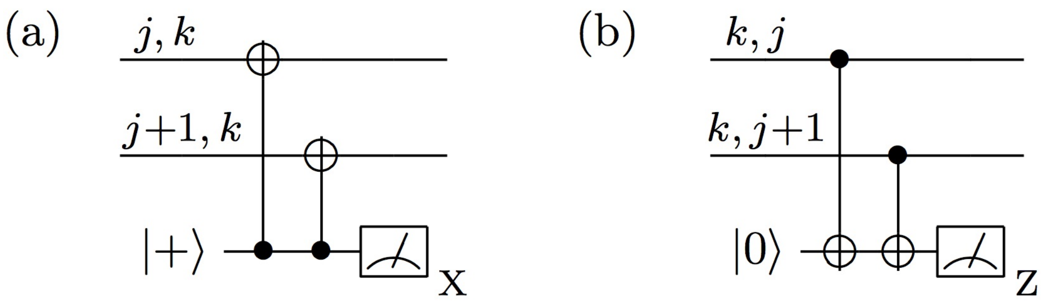

Figure 7: (a) Circuit for measuring the gauge operator $X_{j,k}X_{j+1,k}$. (b) Circuit for measuring the gauge operator $Z_{k,j}Z_{k,j+1}$.

[9]Figure taken from [9].

Topological subsystem codes

2D topological subsystem codes are a family of codes defined on a lattice which is embedded into a plane or a torus. That means that the qubits are either located on a confined 2-dimensional space or that the space is periodic in both directions (in loose terms: if we start at a qubit on the lattice and walk in one direction we will arrive at the same qubit we started at). All topological subsystem codes should have the following properties:

- The stabilizer group $\mathcal{S}$ has spatially local generators which can be identified with closed homologically trivial loops on the lattice.
- The syndrome measurement can be done by measuring the eigenvalues of operators acting solely on two qubits.
- The non-trivial undetectable bare errors can be identified with closed homologically non-trivial loops on the lattice.

Note that from the last property follows that the non-trivial undetectable errors are non-local and grow with the size of the lattice. This gives us an extra advantage since non-local errors occur with a far smaller probability than local errors. But the main advantage of topological subsystem codes over topological stabilizer codes is the second property. The syndrome measurement of all known stabilizer codes involves at least four-qubit measurements which are more difficult to realize.

Instead of dealing with the lattice of the code directly we will use graphs to define topological codes. The locations of the qubits will be referred to as *sites* and the set of all sites of a code is called V. All sites are connected via *edges* which can connect either two or three sites. The set of all edges E is therefore the union of E_2 and E_3 which denote the set of edges connecting two and three sites. This means we are dealing with a 3-valent hypergraph. The edges also have a property: They can be of either X-,Y- or Z-type. We will have the following restrictions on the edges:

- Each site has exactly three incident edges,
- Any pair of edges share at most one site,
- There are no sites shared by two triangles.

Each edge gives rise to an operator. If we have an edge $e = (u, v) \in E_2$ connecting two sites u and v and e is of X-type then its corresponding operator is $K_e := X_u X_v$ (equivalently for Y- and Z-type). These two-qubit interactions are called *link operators*. The following theorem gives a necessary and sufficient condition on when two of those operators commute.

Lemma 5.1. *Suppose we have two edges $e, f \in E_2$. Then their according link operators commute if and only if they do not share exactly one end point.*

Proof. If they do not share any end-points the operators act on different qubits and thereby commute. If they act on the same two qubits than the operators are identical which also lets them commute. If they only share one end-point u then by construction the act as different Pauli-operators on u and therefore they anti-commute. ■

Edges connecting three qubits give rise to the *triangle operators*. Suppose the Z-type edge $e = (u, v, w)$ connects the sites u,v and w then the triangle operator is $Z_u Z_v Z_w$ (equivalently for Y- and Z-type). If we assume that X- ,Y- and Z-errors occur with the same probability[10] then types of the link and triangle operators are specified up to a permutation. Thus without loss of generality we can set all edges connecting three sites to be of Z-type and the edges connecting two sites to be of X- respectively Y-type.

Theorem 5.2. *Suppose $e, f \in E$ are two edges. Then for their respective operators K_e and K_f we have*

$$K_e K_f = (-1)^{\eta(e,f)} K_f K_e \tag{5.4}$$

with

$$\eta(e, f) = \begin{cases} 0, & \textit{if e and f share an even amount of sites or } e, f \in E_3, \\ 1, & \textit{otherwise.} \end{cases} \tag{5.5}$$

Proof. The case where $e, f \in E_2$ follows directly from lemma 5.1. Suppose $e, f \in E_3$ then they are both Z-type and therefore K_e and K_f commute. If e is in E_2 and f in E_3 then they share by construction of the graph either one or no sites. If they share one site $u \in V$ than K_e acts on the according qubit as either X_u or Y_u and K_f as Z_u. Thus K_e and K_f anti-commute. If they do not share any sites at all they trivially commute. ■

We can use the link and triangle operators as components of bigger operators by considering sets of edges. The subsets which will turn out to be useful are cycles on the graph.

Definition 5.3. *A subset of edges $M \subseteq E$ is called a cycle if and only if each site of the lattice has an even number of incident edges from M.*

[10]These types of quantum channels are called *depolarizing channels*.

We will denote the set of all edges of a closed loop with M which is a subset of E. These closed loops give rise to the so called *loop operators*. If M is a closed loop of edges then $W(M)$ is the corresponding loop operator. One reason why we consider loop operators is that we have a general commutation rule similar to theorem 5.1.

Theorem 5.4. *Suppose $e \in E$ is an edge and $M \subseteq E$ is a cycle. Their according operators commute if and only if e is not a triangle contained in M, i.e.*

$$[W(M), K_e] = 0 \leftrightarrow e \notin M \cap E_3 \tag{5.6}$$

Proof. Suppose $e \in E_2$ then we must consider the case where e is an element of M or it shares a site with edges laying in M (if not K_e and $W(M)$ would act on different qubits). If $e \in M$ then let $u, v \in V$ be the sites connected by e. Both have by definition 5.3 an even amount of edges incident from M from which we can conclude that K_e and $W(M)$ commute. The same argument holds for the case where e incides on a site $u \in V$ which also has incident edges from M.
Now let e be in E_3. From the restrictions we put on the graph it follows that if e is not in M than K_e and $W(M)$ do not share any qubits. If on the other hand $e \in M$ then each site of e has an odd amount of incident edges from M which are not equal to e. Thus for the odd amount of edges connected by e we have an odd amount of incident edges from M which each give rise to an operator anti-commuting with K_e and therefore K_e anti-commutes with $W(M)$. ■

Corollary 5.5. *Suppose $M, M' \subseteq E$ are two cycles then*

$$W(M)W(M') = (-1)^{\Delta(M,M')}W(M')W(M) \tag{5.7}$$

where $\Delta(M, M') := |M \cap M' \cap E_3|$ is the number of triangles shared by M and M'.

Let $\mathcal{G}_{loop} := \{W(M) \mid M \subseteq E \text{ is a cycle}\}$ be the group of all loop operators. Instead of dealing with individual links we will define a topological code over the cycles of the graph. Nevertheless the actual gauge group $\mathcal{G}$ of the code is the group of all edge operators. $\mathcal{G}$ can always be reconstructed from $\mathcal{G}_{loop}$ using the following identity:

$$\mathcal{G} = C(\mathcal{G}) \tag{5.8}$$

We also have

$$\mathcal{S} = \mathcal{G}_{loop} \cap C(\mathcal{G}_{loop}) = \mathcal{Z}(\mathcal{G}_{loop}). \tag{5.9}$$

The advantage of using $\mathcal{G}_{loop}$ will become clear when we explicitly construct a topological code. The bare logical operators as well as the stabilizers then can be easily seen by considering the commutation rules given above.

To transform the hypergraph (E, V) into an ordinary graph $(\tilde{E}, \tilde{V})$ we can simply set $\tilde{V} := V$. Furthermore every link $e \in E_2$ can be put into $\tilde{E}$. Triangle edges have to be split up into a triple of links, i.e. if we have an edge $e = (u, v, w) \in E_3$ then $\tilde{E}$ has elements (u, v), (v, w) and (w, u). To make a distinction between edges in $\tilde{E}$ which arise from links and triangles we will refer to the first type as *solid links* and to the latter as *dashed links*.

The square-octagon code

Let us look at an example for a 3-valent graph that suffices the restraints which were given in the preceding section. In figure 8 we can see such a hypergraph. Because of the alignment of the sites and edges the code which is defined by this hypergraph is called *square-octagon code*.

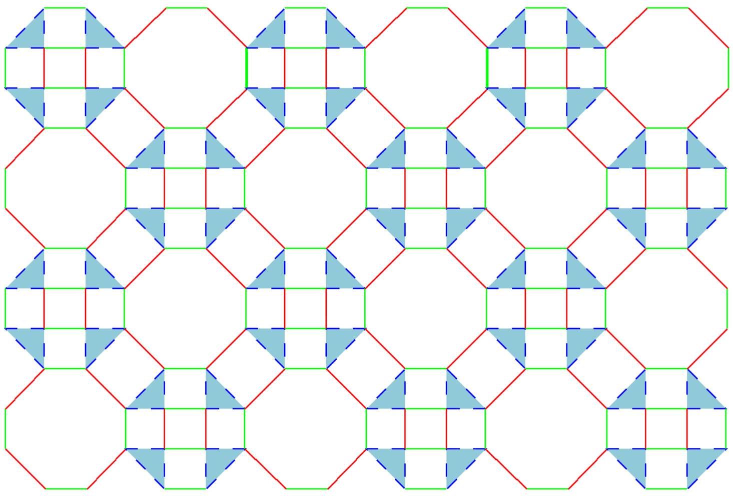

Figure 8: The 3-valent hypergraph of the square-octagon code. The graph has three types of edges: The links which can be of X- or Y-type (green and red solid lines) and triangles which we chose as Z-type operators (blue trangles with dashed lines). Each site of the graph has two incident links and one incident triangle.

To understand the codes properties let us first analyze the loops in the hypergraph. There are essentially two different kinds: Those which give rise to operators that have spatially local support and those that 'grow' with the size of the lattice. The existence of these cycles can be explained by the topology of the lattice on which we locate the sites of the graph: Recall that we have the open-boundary condition and therefore we can imagine the graph being located on the surface of a torus. Thus we can construct loops which can be tightened to a point (homologically trivial loops) and those who are located around the torus and therefore can not be contracted (homologically non-trivial loops).

Let us first consider the homologically trivial ones. They can be associated with the elementary squares and octagons of the lattice. Each octagon contributes two basis cycles called A and C (see figure 9). The cycle A consists of the 8 link operators that form the boundary of an octagon which does not have any sites in it. The cycle B is made up of 12 links and 8 triangles around the octagon. Each square contributes two basic loops B and D, where D consists of 4 links that form a square and B consists of 4 triangles and 6 links.

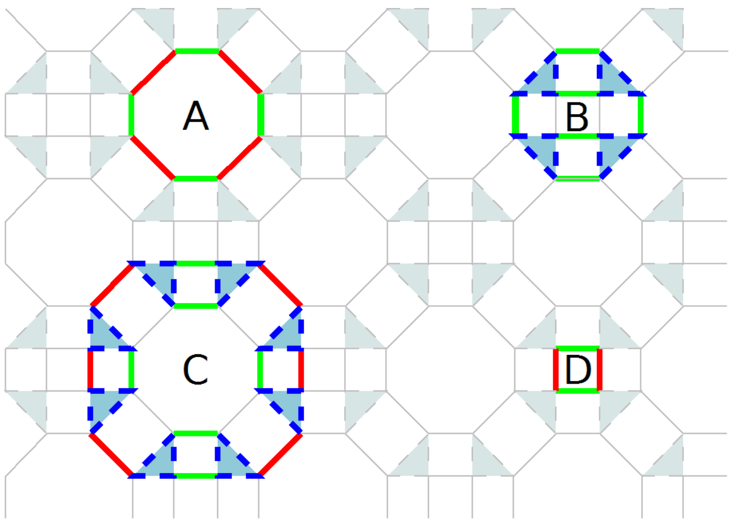

Figure 9: The four elementary cycles of the hypergraph. Edges contributing to the cycles are colored while others are grayed out.

The homologically non-trivial loops can be seen in figure 10. The basis cycles which correspond to the vertical loops on the lattice are called Z_1 and Z_2 and those which correspond to horizontal loops are called X_1 and X_2. Note that despite the suggestive notation these are not operators but cycles in the cycle space of the hypergraph.

The cycles A and D do not contain any triangles from which follows with corollary 5.5 that $W(A)$ and $W(D)$ will commute with all other loop operators, i.e. $W(A), W(D) \in C(\mathcal{G}_{loop}) \cap \mathcal{G}_{loop}$. By inspection it is also easy to see that the cycles B and C share either zero or two triangles with all other basis cycles. Again we can conclude with corollary 5.5 that $W(B)$ and $W(C)$ commute with all the other loop operators.

The cycles X_1, X_2, Z_1 and Z_2 do not share an even amount of triangles with all cycles. As the notation indicates do $W(X_1)$ and $W(Z_1)$ anti-commute since X_1 and Z_1 share one triangle. The same is true for $W(X_2)$ and $W(Z_2)$. All other pairings of the cycles X_1, X_2, Z_1 and Z_2 share no triangles which again leads to commuting operators. Because A, B, C, D and X_1, X_2, Z_1, Z_2 form basis of the cycle

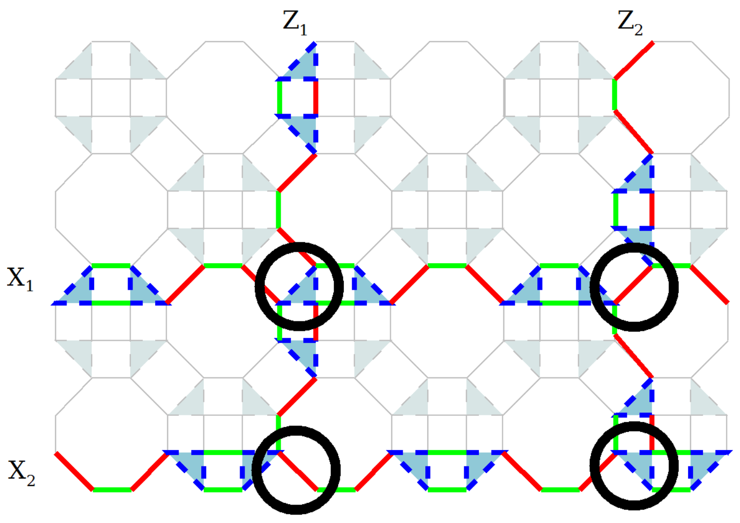

Figure 10: The basis cycles of the hypergraph which correspond to homologically non-trivial loops on the lattice. The black circles mark edges that are shared by two cycles. Note that X_1 and Z_1 (respectively X_2 and Z_2) share one triangle and therefore give rise to anti-commuting operators $W(X_1)$ and $W(Z_1)$ while X_1 and Z_2 (respectively X_2 and Z_1) only share one link operator.

space we can conclude that the stabilizer group $\mathcal{S}$ is generated by the operators corresponding to A, B, C and D.

$$\mathcal{S} = \langle W(A), W(B), W(C), W(D)\rangle \tag{5.10}$$

The following theorem enables us to do a syndrome measurement by only measuring elements of the gauge group i.e. link operators.

Theorem 5.6 (Syndrome measurement with link operators)**.** *The eigenstate of a stabilizer $S \in \mathcal{S}$ can be measured by a set of link operators if and only if S can be written as a product of these link operators*

$$S = K_m \cdots K_1 \tag{5.11}$$

and K_j commutes with the ordered product of all preceding link operators, i.e.

$$[K_j, K_{j-1} \cdots K_1] = 0 \;\; \textit{for all } j \in \{2, ..., m\} \tag{5.12}$$

Proof. Let us first proof that the conditions (5.11) and (5.12) are necessary. For doing so we analyze how such an adaptive eigenvalue measurement works in general. Assume that we already measured generators $K_1, ..., K_j$ with measurement results $m_1, ..., m_j \in \{\pm 1\}$. Then

$$P_j := \frac{I + m_j K_j}{2} \tag{5.13}$$

is the projector into the m_j-eigenspace of K_j and can therefore be used to describe the measurement of K_j. Now define

$$R_j := P_j \cdots P_1 \tag{5.14}$$

as the operator which describes the ordered measurements of the sequence K_1 to K_j for the configuration of outcomes $m := (m_1, ..., m_j)$. We say that this procedure simulates the eigenvalue measurement of S if there exists a function $\sigma : \{\pm 1\}^l \rightarrow \{\pm 1\}$ such that

$$R_l S = S R_l = \sigma(m) R_l \text{ for all } m \in \{\pm 1\}^l \tag{5.15}$$

This means that the state after the measurements of the K_i must be an eigenvector of S with eigenvalue $\sigma(m)$.
What we have to prove is that this is only possible if S can be decomposed as in (5.11) while satisfying (5.12). To do so we define are abelian subgroups of $\mathcal{P}_n$ in the following way: First we set $\mathcal{S}_0 := \{I\}$. For $j \in \{1, ..., l\}$ the groups are defined inductively by

$$\mathcal{S}_j := \langle m_i K_i, \mathcal{S}'_{j-1} \rangle \tag{5.16}$$

where $\mathcal{S}'_j$ is defined as the set of all elements of $\mathcal{S}_j$ which commute with K_{j+1}, i.e.

$$\mathcal{S}'_j := \{P \in \mathcal{S}_j \mid [P, K_{j+1}] = 0\} \tag{5.17}$$

From (5.15) follows that $\sigma(m)S \in \mathcal{S}_m$ which allows us to write

$$\sigma(m)S = (m_l K_l)^{\alpha_l} \cdot S_{l-1} \tag{5.18}$$

for some $S_{l-1} \in \mathcal{S}_{l-1}$ and $\alpha_l \in \{0, 1\}$. Since $\mathcal{S}'_j \subseteq \mathcal{S}_j$ we can apply this method again until we arrive at the decomposition

$$\sigma(m)S = (m_l K_l)^{\alpha_l} \cdots (m_1 K_1)^{\alpha_1} \tag{5.19}$$

where $\alpha_l, ..., \alpha_1 \in \{0, 1\}$. By construction we have $(m_{j-1} K_{j-1})^{\alpha_{j-1}} \cdots (m_1 K_1)^{\alpha_1} \in \mathcal{S}'_j$ for all $j \in \underline{l}$. Thus we get $[K_j, K_{j-1}^{\alpha_{j-1}} \cdots K_1^{\alpha_1}] = 0$. By omitting all factors with $\alpha_i = 0$ we obtain the equations (5.11) and (5.12).

Now let us proof that (5.11) and (5.12) are also sufficient. For doing so we proof by induction over j that after measuring K_j our state is an eigenvector of $S_j := K_j \cdots K_1$ with eigenvalue $\sigma_j = m_j \cdots m_1$. Suppose $j = 2$ then equation (5.12) implies that K_1 and K_2 commute and therefore their eigenvalues m_1 and m_2 can be measured simultaneously. From equation (5.11) we get $S = K_2 K_1$. Therefore the eigenvalue of S is $\sigma = m_2 \cdot m_1$ which proves the basis case. For the inductive step assume that the eigenvalue of S_j can be measured as described above. From (5.12) follows that K_{j+1} commutes with S_j. Hence their eigenvalues can be measured independently. After measuring K_j the eigenvalue of S_j is $\sigma_{j+1} = m_{j+1}\sigma_j$. ■

An example for syndrome extraction by measuring link operators

Let us consider the square-octagon code C_{SO} as an example. Recall that the stabilizers of C_{SO} were loop operators generated by $W(A)$, $W(B)$, $W(C)$, and $W(D)$ which can be seen in figure 9. For $W(D)$ we can check the conditions straightforward. We first take all X-type link operators K_1 and K_3[11] which obviously commute among each other and also commute with the Y-type operators K_2 and K_4 because they share an even amount of qubits with them. Hence we can decompose $W(D)$ into

$$W(D) = K_4 K_2 K_3 K_1 \tag{5.20}$$

which suffices equations (5.11) and (5.12). Note that the decomposition is not unique. The decomposition of $W(A)$ follows completely analogous:

$$W(A) = K_8 K_6 K_4 K_2 K_7 K_5 K_3 K_1 \tag{5.21}$$

Actually the syndrome measurement can be done in only two steps in both cases since we can measure all X- and Y-type operators at once.

For the stabilizer $W(B)$ the definition of the cycles helps us to find decompositions that suffice (5.12): Because every site has an odd number of incident edges and the corresponding link operators anti-commute we can take the link operators that make up the stabilizer and group them after their respective type (X, Y or Z). Hence $W(B)$ can be measured by first measuring the dashed links because each of those commute with every other link operator from $W(B)$ and second all solid link operators. The decomposition has the following form:

$$W(B) = \left[\prod_{e \in E_X \cap C} K_e\right] \cdot \left[\prod_{e \in E_Z \cap C} K_e\right] \tag{5.22}$$

The decomposition for $W(C)$ follows in a similar way. We use the fact that after measuring all dashed links all left solid link operators commute and thus we can write:

$$W(C) = \left[\prod_{e \in E_X \cap C} K_e\right] \cdot \left[\prod_{e \in E_Y \cap C} K_e\right] \cdot \left[\prod_{e \in E_Z \cap C} K_e\right] \tag{5.23}$$

Again we can see that the syndrome extraction can be done in a two way measurement because after the measurement of the dashed link operators all solid link operators commute.

[11] Link operators are numbered beginning at the top and continuing clockwise

The conditions of theorem 5.6 would be easily fulfilled if we knew that the gauge operators were all of one type since then they would trivially commute. In [1] was proven by the use of so-called Majorana fermions that stabilizer codes can be mapped such that the stabilizer group becomes a direct product of X- and Z-type operators. Here we extend that proof to subsystem codes and use a more direct way without the use of Majorana fermions.

Theorem 5.7 (Code mapping)**.** *Every $[[n, k, q, d]]$ subsystem code can be mapped onto a $[[4n, 2k, 2q, 2d]]$ subsystem code. The gauge group $\tilde{\mathcal{G}}$ of the new code is a direct product of $\tilde{\mathcal{G}}(X)$ and $\tilde{\mathcal{G}}(Z)$ which only contain X- respectively Z-type operators. This mapping also preserves the geometric locality of a code up to a constant factor.*

Proof. First define the mapping $\sigma : \mathcal{P}_n \to \mathbf{Z}_2^{4n}$:
Let be P be an element of $\mathcal{P}_n$.

1. If P acts as X on the jth qubit then $\sigma(P)_j = \sigma(P)_{j+n} = 1$
2. If P acts as Y on the jth qubit then $\sigma(P)_j = \sigma(P)_{j+2n} = 1$
3. If P acts as Z on the jth qubit then $\sigma(P)_j = \sigma(P)_{j+3n} = 1$

All other entries are set to 0. If we think of $\mathbf{Z}_2^{4n}$ as divided into 4 blocks of length n then the jth entry of block 1 marks if P acts non-trivial on the jth qubit and blocks 2, 3 and 4 if P acts as X, Y or Z.
The other two mappings we need for the proof are

$$\tau_X : \mathbf{Z}_2^{4n} \to \mathcal{P}_{4n},\ (c_1, ..., c_{4n})^{tr} \mapsto \prod_{j \in \underline{4n}} X_j^{c_j} \tag{5.24}$$

and

$$\tau_Z : \mathbf{Z}_2^{4n} \to \mathcal{P}_{4n},\ (c_1, ..., c_{4n})^{tr} \mapsto \prod_{j \in \underline{4n}} Z_j^{c_j}. \tag{5.25}$$

Furthermore we define the vector space

$$D := \langle e_j + e_{j+n} + e_{j+2n} + e_{j+3n} =: d_j \mid j \in \underline{n} \rangle \leqslant \mathbf{Z}_2^{4n} \tag{5.26}$$

where e_j denotes the jth standard base vector of $\mathbf{Z}_2^{4n}$.

Let $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{G}}$ denote the stabilizer respectively the gauge group. To get the generators of $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{G}}$ we map each generator of $\mathcal{S}$ and $\mathcal{G}$ under $\tau_X \circ \sigma$ and add the images of D under τ_X and τ_Z, i.e.

$$\tilde{\mathcal{G}} := \langle(\tau_X \circ \sigma)(G), (\tau_Z \circ \sigma)(G), \tau_X(D), \tau_Z(D)\rangle \tag{5.27}$$

$$\tilde{\mathcal{S}} := \langle(\tau_X \circ \sigma)(S), (\tau_Z \circ \sigma)(S), \tau_X(D), \tau_Z(D)\rangle \tag{5.28}$$

The set of non-trivial logical operators of the new code is defined as

$$\tilde{\mathcal{L}} := C(\tilde{\mathcal{S}}) - \tilde{\mathcal{G}}. \tag{5.29}$$

Let us proof first of all that following this procedure we end up with a proper subsystem code. Let P and P' be operators from $\mathcal{P}_n$.

Claim. *P and P' commute if and only if $\sigma(P) \cdot \sigma(P') = 0$.*

To see why this is the case let us first assume that $[P, P'] = 0$. Then there is an even number of qubits on which they act with different elements from $\{X, Y, Z\}$. If we take the inner product $\sigma(P) \cdot \sigma(P')$ then entries where P and P' act as the same Pauli cancel each other out leaving only an even amount of ones from block one. Therefore $\sigma(P) \cdot \sigma(P')$ is equal to 0. If $[P, P'] \neq 0$ then there will be consequently an odd amount of anti-commuting operations and thus with a similar argument as above $\sigma(P) \cdot \sigma(P') \neq 0$.

Moreover if $\sigma(P) \cdot \sigma(P') = 0$ then $(\tau_X \circ \sigma)(P)$ and $(\tau_Z \circ \sigma)(P')$ have even overlap and therefore they commute. This shows that commuting generators still commute after being mapped onto X- respectively Z-type operators. Furthermore do the elements of D have zero inner product with all vectors of the image of σ. Thus we can conclude that operators in $\tilde{\mathcal{S}}$ commute with operators in $\tilde{\mathcal{G}}$ and that up to phase factors we have $\mathcal{Z}(\tilde{\mathcal{G}}) = \tilde{\mathcal{S}}$. From the construction of $\tilde{\mathcal{S}}$ and the fact that it is an abelian group it follows that $-I \notin \tilde{\mathcal{S}}$.
Consequently $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{G}}$ define a subsystem code.

Obviously our new code encodes on $\tilde{n} = 4n$ physical qubits. To show that the other parameters of the code also have the desired properties we need to show the following claim.

Claim. *Linearly independent operators stay linearly independent after the mapping and all operators from $\tau_{X/Z}(D)$ are linearly independent from all operators of the image of $\tau_{X/Z} \circ \sigma$.*

We proof the contrapositive. For this we take operators $P_0, ..., P_m$ from $\mathcal{P}_n$ such that $\sigma(P_0), ..., \sigma(P_m)$ are linearly dependent. There is a subset of indices $I \subseteq \underline{m}$ such that

$$\sigma(P_0) = \sum_{i \in I} \sigma(P_i) \tag{5.30}$$

Because $ker(\sigma) = \langle iI \rangle$ we can conclude

$$P_0 \sim \prod_{i \in I} P_i \tag{5.31}$$

from which immediately follows that

$$r(P_0) = \sum_{i \in I} r(P_i) \tag{5.32}$$

and therefore $P_0, ..., P_m$ are linearly dependent. The rest follows from the fact that the d_j are not in $Im(\sigma)$.

This allows us to simply count the generators in the new code: Since every generator from the old code was mapped onto two generators in the new code and we added $2n$ independent generators to the stabilizer we get $\tilde{r} = 2r + 2n$ and $\tilde{q} = 2q$. Consequently the new code encodes $\tilde{k} = \tilde{n} - \tilde{r} - \tilde{q} = 2k$ qubits.

We can also see that the gauge group $\tilde{\mathcal{G}}$ can be represented as the direct product of two subgroups, $\tilde{\mathcal{G}} = \tilde{\mathcal{G}}(X) \cdot \tilde{\mathcal{G}}(Z)$, where $\tilde{\mathcal{G}}(X)$ and $\tilde{\mathcal{G}}(Z)$ contain only X-respectively Z-type elements of $\mathcal{P}_n$. Of course the same is true for $\tilde{\mathcal{S}}$ as well.
For the distance of the new code we show:

Claim. *The minimum weight of the logical operators in the new code has doubled.*

First let us proof that a non-trivial logical operator L from $C(\mathcal{S}) - \mathcal{G}$ is mapped onto a non-trivial logical operator $\tilde{L}$ in $\tilde{\mathcal{L}}$. We set w.l.o.g. $\tilde{L} := (\tau_X \circ \sigma)(L)$ and show that $\tilde{L}$ is in $C(\tilde{\mathcal{S}}) - \tilde{\mathcal{G}}$.[12]

It has been already shown that $\tilde{L}$ commutes with every element in $\tilde{\mathcal{S}}$. We show that $\tilde{L} \notin \tilde{\mathcal{G}}$ by contradiction and assume that

$$\tilde{L} \in \tilde{\mathcal{G}}(X) = \langle (\tau_X \circ \sigma)(G_i), \tau_X(d_j) \mid i \in \underline{r+q}, j \in \underline{n} \rangle \tag{5.33}$$

which is eqivalent to

$$\sigma(L) \in \langle \sigma(G_i), d_j \mid i \in \underline{r+q}, j \in \underline{n} \rangle. \tag{5.34}$$

Thus there is a $G \in \mathcal{G}$ and a $d \in D$ such that $\sigma(L) = \sigma(G) + d \Leftrightarrow d = \sigma(L) + \sigma(G)$. The vector d can not be 0 because then we would have $L \in \mathcal{G}$. But also for $d \neq 0$ we get a contradiction since G and L can only act as either X, Y or Z on each qubit. Therefore $\tilde{L}$ must be an element of $C(\tilde{\mathcal{S}}) - \tilde{\mathcal{G}}$. Note that by mapping L we double the weight since for each qubit that L acts on non-trivially $\tilde{L}$ acts on two: Namely one in block 1 and one in block 2, 3 or 4.

[12] The argument for $\tilde{P} = (\tau_Z \circ \sigma)(P)$ is the same.

Suppose $\tilde{L}$ is a minimum weight logical operator from $\tilde{\mathcal{L}}$. Then w.l.o.g. it is a pure X-type operator which means that there must be a $L \in \mathcal{L}$ and a $v \in D$ such that $\tilde{L} = \tau_X(\sigma(L) + v)$. The vector $\sigma(L)$ has at least $2d$ non-zero entries and adding v does not change this number. Therefore $\tilde{L}$ has at least the weight $2d$.

We can think of this mapping as splitting up one qubit into four qubits. If $\mathcal{S}$ is generated by geometrically local operators then these are mapped onto operators which themselves still act locally. Therefore the new code is geometrically local as well. ■

Remark 5.8. *Note that τ_X and τ_Z are homomorphisms while σ is not, for example we have for $n = 1$*

$$\sigma(XYZ) = \sigma(iI) = 0 \neq e_1 + e_2 + e_3 + e_4 = \sigma(X) \cdot \sigma(Y) \cdot \sigma(Z). \tag{5.35}$$

However we have

$$\sigma(P_1P_2) - \sigma(P_1)\sigma(P_2) \in D \text{ for all } P_1, P_2 \in \mathcal{P}_n. \tag{5.36}$$

Therefore $\sigma : \mathcal{P}_n \to \mathbf{Z}_2^{4n}/D$ is a homomorphism.[13]

It is of course important that the locality of the code is preserved because it is our goal to do local measurements. The following theorem provides exactly this.

Corollary 5.9. *Every topological subsystem code can be mapped such that the error syndrome can be extracted by measuring gauge group elements.*

Proof. We need to show that for all generators $\tilde{S}_i$ of $\tilde{\mathcal{S}}$ there exist $\tilde{G}_1, ..., \tilde{G}_m$ such that

$$\tilde{S} = \tilde{G}_m \cdots \tilde{G}_1 \tag{5.37}$$

and

$$[\tilde{G}_j, \tilde{G}_{j-1} \cdots \tilde{G}_1] = 0 \text{ for all } j \in \underline{m}. \tag{5.38}$$

Equation (5.38) is easily seen to be true since $\tilde{\mathcal{S}}$ is a direct product of $\tilde{\mathcal{S}}(X)$ and $\tilde{\mathcal{S}}(Z)$. To show that (5.37) holds:

$$\begin{aligned}
\tilde{S}_i &= (\tau_X \circ \sigma)(S_i) && (5.39)\\
&= (\tau_X \circ \sigma)(K_m \cdots K_1) && (5.40)\\
&= \tau_X\left(\sum_{i \in \underline{m}} \sigma(K_i) + d\right) && (5.41)\\
&= \prod_{i \in \underline{m}} (\tau_X \circ \sigma)(K_i) \cdot \tau_X(d) && (5.42)
\end{aligned}$$

The term in (5.42) is a product of generators of $\tilde{\mathcal{G}}(X)$ which shows that $\tilde{\mathcal{S}}_i$ can be decomposed gauge operators. The statement now follows from theorem 5.6. ■

[13]The vector space $\mathbf{Z}_2^{4n}/D$ is the quotient space.

Mapping of the SO-code

In figure 11 we see how the stabilizers of the SO-code get mapped. Since it is difficult to visualize all blocks at once we can see how the stabilizers look on each block. As mentioned in the proof we can think of this mapping as splitting up a single qubit into four qubits laying next to each other. Therefore the local operators of the new code stay local in the new code. The blocks are not independent but they are linked by the elements from $\tau_{X/Z}(D)$.

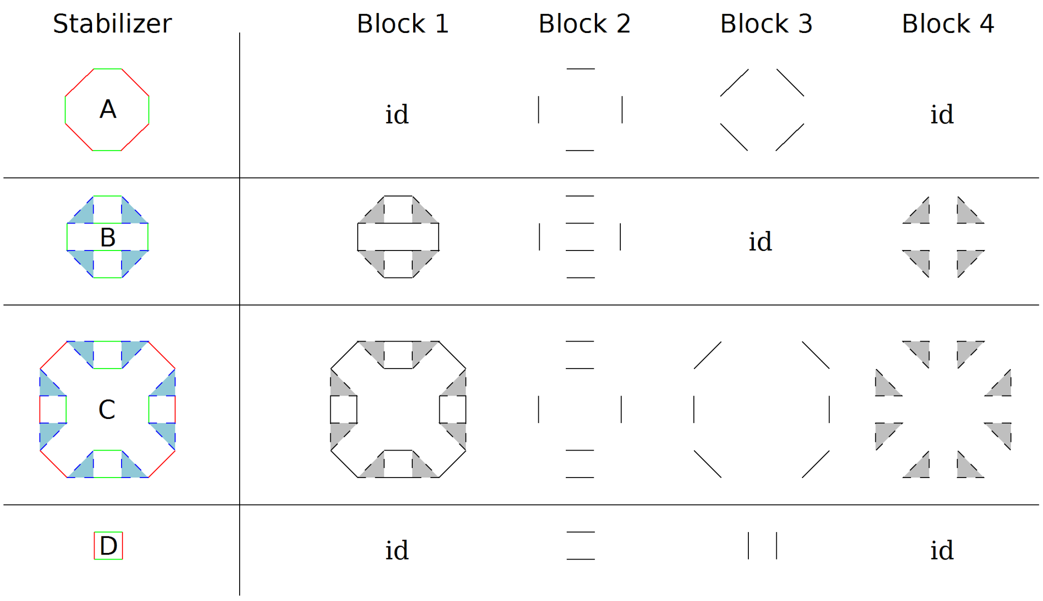

Figure 11: The stabilizer generators of the SO-code before and after being mapped. Note that the mapped elements of the vector space D are not visible here. They provide parity checks among the corresponding qubits of each block i.e. the i-th qubit of block 1,2,3 and 4 for all $i = 1, ..., n$.

6 Conclusion

Subsystem codes provide good properties for active error correction while combining them with passive error avoiding. We have seen how these class of error correcting codes is constructed and also analyzed how they can protect information from being corrupted. It turned out that the syndrome measurement could be done in a different way than for stabilizer codes - namely by measuring gauge operators. This property in combination with a geometric interpretation then led to a simplification of the error detection process.

We also introduced the topological codes and gave a way to construct them. At the end we provided a mapping onto subsystem codes which have certain properties. This allowed us to show that any topological subsystem code can be mapped such that the error syndrome can be extracted by local gauge measurements.

7 Appendix

Standard form for stabilizer codes

Suppose we are given a set of generators for a stabilizer group of a code which encodes k logical qubits on n physical qubits. In general these generators do not have to be linearly independent. Then the check matrix for this code is given by

$$M(\mathcal{S}) = [M_1|M_2] \tag{7.1}$$

which has $n - k$ rows. To bring $M(\mathcal{S})$ into a standard form we will use Gaussian elimination on blocks of $M(\mathcal{S})$. To do this we need to do elementary matrix operations so we need to check if those can be done while ending up with the check matrix of an equivalent code. Swapping rows of $M(\mathcal{S})$ is allowed since it corresponds to reordering the stabilizers. Also swapping of columns can be done because by doing this we just put the qubits into a different order. Finally we can also replace a row by adding another row which is equivalent to replace a generator S_i with S_iS_j where $i \neq j$.

The procedure to bring $M(\mathcal{S})$ into standard form works as follows: First by Gaussian elimination on M_1 we bring $M(\mathcal{S})$ into the form:

$$\left[\begin{array}{cc|cc} I & A & B & C \\ 0 & 0 & D & E \end{array}\right] \tag{7.2}$$

Where the first block has r rows and the second block $n - r$ rows. Subsequently we perform a Gaussian elimination on E and obtain:

$$\left[\begin{array}{ccc|ccc} I & A_1 & A_2 & B & C_1 & C_2 \\ 0 & 0 & 0 & D_1 & I & E_2 \\ 0 & 0 & 0 & D_2 & 0 & 0 \end{array}\right] \tag{7.3}$$

Where the the blocks on the right hand side have $k + s$ columns. The first r generators do not commute with the last s unless $D_2 = 0$. Hence we can disregard the last s rows. Additionally we may eliminate $C_1 = 0$ by adding rows from below. Finally our check matrix takes the standard form:

$$\left[\begin{array}{ccc|ccc} I & A_1 & A_2 & B & 0 & C_2 \\ 0 & 0 & 0 & D_1 & I & E_2 \end{array}\right] \tag{7.4}$$

Codes with a check matrix as in (7.4) are in standard form.

References

[1] Sergey Bravyi, Bernhard Leemhuis, Barbara M. Terhal *"Majorana Fermion Codes", arXiv:1004.3791v2 [quant-ph], v2 2010*

[2] Martin Suchara, Sergey Bravyi, Barbara M. Terhal *"Constructions and Noise Threshold of Topological Subsystem Codes", arXiv:1012.0425v2 [quant-ph], v2 2010*

[3] Michael A. Nielsen, Isaac L. Chuang *"Quantum Computation and Quantum Information", Cambridge University Press 2010*

[4] David Poulin *"Stabilizer Formalism for Operator Quantum Error Correction", quant-ph/0508131, v4 2006*

[5] Paolo Zanardi, Daniel Lidar, Seth Lloyd *"Quantum tensor product structures are observable-induced", quant-ph/0308043, 2003*

[6] David W. Kribs, Raymond Laflamme, David Poulin, Maia Lesosky *"Operator quantum error correction", arXiv:quant-ph/0504189v3, 2003*

[7] Michael A. Nielsen, David Poulin *"Algebraic and information-theoretic conditions for operator quantum error-correction", arXiv:quant-ph/0506069v1, v1 2005*

[8] Daniel Gottesman *"An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation", arXiv:0904.2557v1, v1 2009*

[9] Panos Aliferis, Andrew W. Cross *"Subsystem fault tolerance with the Bacon-Shor code", arXiv:quant-ph/0610063v3, v3 2007*

[10] Bernhard Leemhuis *"Geometrical Quantum Codes", Master thesis, University of Amsterdam 2010*

[11] Héctor Bombín *"Structure of 2D Topological Stabilizer Codes", arXiv:1107.2707v1 [quant-ph], v1 2011*

[12] Dave Bacon *"Operator Quantum Error Correcting Subsystems for Self-Correcting Quantum Memories", arXiv:quant-ph/0506023v4, v4 2005*

[13] Dave Bacon, Andrea Casaccino *"Quantum Error Correcting Subsystem Codes From Two Classical Linear Codes", arXiv:quant-ph/0610088v2, v2 2006*

EIDESSTATTLICHE ERKLÄRUNG

Ich versichere, dass ich meine Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.