Parity Check Schedules for Hyperbolic Surface Codes

Jonathan Conrad
September 18, 2017

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

The present work was submitted to the
Institute for Quantum Information
Faculty of Mathematics, Computer Science and Natural Sciences - Faculty 1

First reviewer
Prof. Dr. B. M. Terhal

Second reviewer
Prof. Dr. F. Hassler
Acknowledgements

Foremost, I would like to express my gratitude to Prof. Dr. Barbara M. Terhal for giving me the opportunity to complete this thesis under her guidance, insightful discussions and her helpful advice.

I would like to give my sincere thanks to Dr. Nikolas P. Breuckmann for his patience with all my questions and his good advice.

Further, I would also like to thank Dr. Kasper Duivenvoorden and Christophe Vuillot for many interesting and helpful discussions. In general, I would like to thank the IQI for its hospitality and everyone in it for making my stay a very joyful experience.

As always, I am grateful for the constant support of my family and friends.
Table of Contents

1 Introduction 1

2 Preliminaries 2

3 Stabilizer Codes 5
  3.1 Errors 6
  3.2 The Bit Flip Code 6
  3.3 The Toric Code 11

4 Homological Code Construction 15
  4.1 $\mathbb{Z}_2$-Homology 18

5 Hyperbolic Surface Codes 21
  5.1 The Gauss-Bonnet Theorem 21
  5.2 The Small Stellated Dodecahedron as a Hyperbolic Surface Code 26

6 Parity Check Scheduling 32
  6.1 Separate Scheduling 33
  6.2 Interleaved Scheduling 39
  6.3 Coloring the Small Stellated Dodecahedron 45

7 Fault Tolerant Quantum Computation 47
  7.1 Fault Tolerance 47
  7.2 Fault Tolerance of the Small Stellated Dodecahedron Code 52
  7.3 The Decoder 57
  7.4 Error Models 58

8 Conclusion 60

9 Appendix 61
  A Coloring Algorithms 61
  B Lookup Tables for the Small Stellated Dodecahedron Code 63
  C The Small Stellated Dodecahedron is a Cayley-Graph 64
### Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\oplus$</td>
<td>$\mathbb{Z}_2$ addition: $a \oplus b = a + b \mod 2$; $a, b \in \mathbb{Z}_2$</td>
</tr>
<tr>
<td>$\neg$</td>
<td>The set exclusion</td>
</tr>
<tr>
<td>$V^\otimes_n$</td>
<td>The $n$-fold tensor product of $V$</td>
</tr>
<tr>
<td>$\bigotimes_i^N$</td>
<td>The $N$-fold tensor product over objects labeled by $i = 1, 2, ..., N$</td>
</tr>
<tr>
<td>$\langle g_1, g_2, ..., g_n \rangle$</td>
<td>The group generated by generators $g_1, g_2, ..., g_n$.</td>
</tr>
<tr>
<td>$\text{wt}(O)$</td>
<td>The weight of an operator $O$</td>
</tr>
<tr>
<td>$\mathcal{H}$</td>
<td>A complex Hilbert space</td>
</tr>
<tr>
<td>$\mathcal{G}_n$</td>
<td>The Pauli group acting on $n$ qubits</td>
</tr>
<tr>
<td>$\mathcal{C}_n$</td>
<td>The Clifford group acting on $n$ qubits</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>The stabilizer group</td>
</tr>
<tr>
<td>$\mathcal{L}_S$</td>
<td>The group of logical operators corresponding to the stabilizer group $\mathcal{S}$</td>
</tr>
<tr>
<td>$[[n, k, d]]$</td>
<td>A stabilizer code with $n$ data qubits, $k$ logical qubits and distance $d$</td>
</tr>
</tbody>
</table>
1 Introduction

Whilst the exploration of the possibilities introduced by performing computation on quantum systems instead of classical ones has already given us a peek into its enormous potential, due to their high sensitivity to errors, only a fraction of it has become accessible in experimental realizations. Although improvements in the experimental setups and the choice of the qubits allow a reduction of the error rate, it is practically impossible to implement a system completely disconnected from any noise introduced by the environment or due to undesired physical interactions inside the system. This serves as a motivation to design quantum error correcting codes as storage for quantum information.

This thesis is structured as follows: Firstly, the basic idea of error correction, the stabilizer code as well as the corresponding notion of the toric code will be reviewed briefly, leading us to discuss the basic approaches of scheduling the parity checks and general rules which these schemes need to respect.

Secondly, we will focus our attention on a small code constructed on a hyperbolic surface, which besides a relatively high, constant encoding rate, offers a beautiful correspondence to a well-studied polyhedral structure. The code arising from the structure of the small stellated dodecahedron will be studied in its properties of efficient scheduling and fault-tolerance.
2 Preliminaries

Before we can outline the basic ideas of quantum error correction, it will be necessary to introduce some basic notions of general quantum information theory.

A qubit is a two-level quantum system described by a two-dimensional Hilbert space $H$ whose basis is given by the Pauli-$\sigma_z$ basis $B = \{|0\rangle, |1\rangle\}$, also called the computational basis. The quantum systems to be considered will always be viewed as closed physical systems, such that quantum information is always conserved. In this sense, the set of all possible operations on $n$ qubits will be given by the unitary group $U(2^n)$.

The most basic operators —so called gates— acting on a single qubit are the single-qubit Pauli operators. These are given by

\[
\begin{align*}
X &= \sigma_x = X^\dagger, \\
Y &= \sigma_y = Y^\dagger, \\
Z &= \sigma_z = Z^\dagger, \\
Y &= iXZ, \\
Z^2 &= X^2 = Y^2 = I,
\end{align*}
\]

These operators generate the Pauli group as a subgroup of $U(2)$,

\[G_1 := \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\} \subset U(2).\]

Analogously, the Pauli group $G_n$ acting on $n$ qubits, which are described on $H^{\otimes n}$, is the union of the single-qubit Pauli groups defined on the separate subspaces:

\[G_n := \bigotimes_{i=1}^n g_i \mid g_i \in G_i : H_i \rightarrow H_i \bigotimes_{i=1}^n \subset U(2^n), \quad (2.1)\]

with lower indices denoting which particular subspace $H_i$ the operator acts on (no action on certain subspaces $H_i$ indicates an unity operator $I$ filled in). Following this convention, we will denote operators on the joint Hilbert space as $O = O_1O_2O_3\ldots$, leaving out trivial operators and the indices referring to the specific subspace.

Further important operations consist of:

- The Hadamard gate $H = H^{-1} = H^\dagger$ transforms between the eigenstates of the $\pm 1$ eigenvalues of $Z$ and $X$ fulfilling the following relations:

\[
\begin{align*}
HXH &= Z, \\
HZH &= X, \\
H|0\rangle &= \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) =: |+\rangle, \\
H|1\rangle &= \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) =: |-\rangle.
\end{align*}
\]

\[\text{[Footnote]} \text{This is practically the phase-space conservation due to the Liouville-theorem. A proof is given in [11].}\]
The \textit{CNOT} gate $C_{A,B}$ is a hermitian and unitary 2-qubit gate, operating on $\mathcal{H}_A \otimes \mathcal{H}_B$ which acts on the computational basis as follows:

$$C_{A,B} : |x_1⟩_A|x_2⟩_B \rightarrow |x_1⟩_A|x_1 \oplus x_2⟩_B.$$  

The qubit in $\mathcal{H}_A$ will be called \textit{control} and the one in $\mathcal{H}_B$ \textit{target}. By applying the CNOT to all input combinations consisting of $|0⟩, |1⟩$ ($|+⟩, |-⟩$) one finds, that this operation can be rephrased in the following commutation relations (also consider Fig. 2.2):

\begin{align*}
C_{A,B}(X_A \otimes I_B) &= (X_A \otimes X_B)C_{A,B}, \quad (2.2) \\
C_{A,B}(I_A \otimes Z_B) &= (Z_A \otimes Z_B)C_{A,B}. \quad (2.3)
\end{align*}

The Hadamard and the CNOT gates are not part of the Pauli group $G_n$, but belong to a supergroup of it — the Clifford-group — which consists of operators in $U(2^n)$ conjugating elements from the Pauli group to one another.

$$C_n := \{c \in U(2^n) \mid cg^{-1} \in G_n; \ g \in G_n\} \supset G_n \quad (2.4)$$

The Pauli group is thus called a normal subgroup or \textit{normalizer} of $C_n$.

The evolution of a state throughout the action of different gates can also be represented via the \textit{circuit-model} in form of Fig. 2.1. Notice that this picture is equivalent to the reversed, written out equation as time runs from left to right.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure2_1.png}
\caption{An example of a quantum circuit enabling third-party controlled teleportation. The time runs from left to right. The gate included in the dashed box is a CNOT gate $C_{1,2}$, taking Qubit 1 as control and qubit 2 as target. The devices pictured in the bottom right represent Pauli-$X$ and Pauli-$Z$ gates, the device labeled $H$ is the discussed Hadamard gate, the remaining two boxes represent measurements in the Pauli-$X$ (on qubit 1) and Pauli-$Z$ basis (on qubit 2). Single lines illustrate \textit{quantum channels} while double lines symbolize \textit{classical channels}. It is up to party 2 to submit his measurement outcome to 3 in order for him to obtain the correct state teleported from 1. For further details consider Chapter 4 in \cite{10}.}
\end{figure}

In the newly introduced \textit{circuit-model} we find the commutator-relations for the CNOT, Eq. (2.2), as:

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure2_2.png}
\caption{CNOT commutation relations.}
\end{figure}

3
As a last item in our initial toolbox we define the \textit{weight} of an operator:

\textbf{Definition 1.} (Weight) The weight $\text{wt}(\cdot)$ of a Pauli group element is the number of qubits the operator acts on nontrivially.

e.g. $\text{wt}(O = X_1) = 1$ or $\text{wt}(O = X_1 Y_2 Z_3) = 3$. 
3 Stabilizer Codes

In this chapter we will review some basic ideas of quantum error correction in order to build towards an understanding of the toric code. Classical information is protected by redundancy. Instead of storing the information in just one physical bit (e.g. the magnetic polarization of an atom), the information is kept in several copies of such bit (e.g. a whole magnetic domain), such that any distortion of a small subset of these bits can be averaged out when one tries to retrieve this information.

However, when one tries to protect information stored on a quantum level from errors in similar fashion, this does not seem easy at first sight, due to the No-Cloning Theorem.

**Theorem 1. (No-Cloning)**

There is no unitary operation $U$ which creates copies of the state of one system on a second, ancillary system. Explicitly

$$\not\exists U : |\psi\rangle \otimes |a\rangle \mapsto |\psi\rangle \otimes |\psi\rangle \in \mathcal{H}_A \otimes \mathcal{H}_B, \forall |\psi\rangle \in \mathcal{H}_A.$$

**Proof.** Suppose such a map $U$ exists, then

$$U(|\psi\rangle \otimes |a\rangle) = |\psi\rangle \otimes |\psi\rangle,$$

$$U(|\phi\rangle \otimes |a\rangle) = |\phi\rangle \otimes |\phi\rangle.$$  

Taking the inner product of both these states yields

$$\langle \psi | \otimes \langle a | U^\dagger U |\phi\rangle \otimes |a\rangle = \langle \psi | |\psi\rangle \langle \psi | |\phi\rangle$$

$$\Leftrightarrow \langle \psi | \phi \rangle = \langle \psi | \phi \rangle^2.$$

This can only hold for $\langle \psi | \phi \rangle = 0$ or $\langle \psi | \phi \rangle = 1$ and thus contradicts our initial assumption of such $U$ existing for all $|\psi\rangle$. $\square$

While this appears to be an obstacle for designing error-prone storages for quantum information, the quantum world offers many interesting alternatives for such designs, which are in general called quantum error correcting codes (QECC). These codes $C$, aim to provide an encoding of $k$ logical qubits into an (mostly) higher amount of $n$ physical qubits.\(^1\) Clearly, it is desirable to minimize the encoding rate $n/k$ while maximizing the ability of the code to safely store the given information.

\(^1\)In classical codes, e.g. the mentioned redundancy-encoding on common harddrives, these quantities are estimated around $k = 1 \mapsto n \approx 10^{12}$. One logical state is encoded in the polarization of a ferromagnetic domain.
3.1 Errors

A code state $|\psi\rangle$ is a composite state of $n$ physical qubits. Due to the two-dimensionality of the single qubits, any error $E_i$ operating on each physical qubit can be written in the $U(2)$-basis given by the Pauli-operators:

$$E_i = \alpha I_i + \beta X_i + \gamma Y_i + \delta Z_i.$$  

Such an error maps the code state $|\psi\rangle$ onto the superposition state

$$E_i |\psi\rangle = (\alpha I_i + \beta X_i + \gamma Y_i + \delta Z_i) |\psi\rangle.$$

Thus, a syndrome measurement — the determination by which $E'_i \in \{I_i, X_i, Y_i, Z_i\}$ the code state was altered — collapses this state onto one of the four states $|\psi\rangle, X_i |\psi\rangle, Z_i X_i |\psi\rangle, Z_i |\psi\rangle$, from which the reverse operation can be applied to retrieve the unerroneous state $|\psi\rangle$. As $Y_i = i X_i Z_i$, it is sufficient to reduce the set of errors to look for onto $X_i$ and $Z_i$.

3.2 The Bit Flip Code

The bit flip and phase flip code are repetition codes, separately capable of determining the positions of $X$-errors (bit flips) and $Z$-errors (phase flips). The Hadamard transformation acts as an isomorphism between these two codes. Thus, these codes are also called dual to each other

$$C_{\text{Phaseflip}}^\perp = C_{\text{Bitflip}}^\perp, \quad C_{\text{Bitflip}}^\perp = C_{\text{Phaseflip}}^\perp.$$

Initially, the logical state to be protected has to be encoded. This is achieved by adding two ancillary qubits and perform CNOTs to map the original state given on $|\psi\rangle$ onto the ancillas.

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad \rightarrow \quad |\psi_L\rangle = \alpha |000\rangle + \beta |111\rangle$$

$$|\psi\rangle = \alpha |+\rangle + \beta |-\rangle \quad \rightarrow \quad |\psi_L\rangle = \alpha |++\rangle + \beta |--\rangle$$

Figure 3.1: Single encoding step protecting against bit (phase-) flip-errors with encoded states $|\psi_L\rangle$.

---

2Note, that this includes both the information about which Pauli-operator was applied and which physical qubit is infected.
This procedure can be described as an extension of the Hilbert space $\mathcal{H}_1 \rightarrow \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \mathcal{H}_3$ while transforming the computational basis in the following way

$$\{|0\rangle, |1\rangle\} \rightarrow \{|0_L\rangle := |000\rangle, |1_L\rangle := |111\rangle\}.$$  

The states $|000\rangle$ and $|111\rangle$ are called codewords of this code, as they define the logical (protected) states $|0_L\rangle$ and $|1_L\rangle$, their span is called the codespace $\mathcal{C}$. The procedure for phase flips is shown on the lower picture in Fig. 3.1, one notices, that due to Eq. (2.2) the CNOTs are performed in the reverse direction as in the bit flip encoding procedure, allowing

$$\{|+, -\rangle\} \rightarrow \{|+L\rangle := |+++\rangle, |-L\rangle := |----\rangle\}.$$  

Note, that this is not a violation of the No-Cloning Theorem since the encoded state $|i\rangle = \alpha|000\rangle + \beta|111\rangle$ is not a separable state. The protection offered by this code is thus not merely due to the inherited redundancy, but to its structure of entanglement.

We have introduced codewords of length 3 which encode $k=1$ logical qubit onto $n=3$ physical ones. In order to flip one logical basis state to another, it would be necessary to flip all three physical qubits, respectively performing an operation of weight $wt(X_L = X_1X_2X_3) = 3$.

**Definition 2.** (Distance) The distance $d$ of a quantum code with codespace $\mathcal{C}$ is the minimum weight of a nontrivial operator $O \subseteq G_n$, which acts normal on the codespace:

$$d = \min_{O: C \rightarrow C} wt(O), |\psi_L\rangle \in \mathcal{C}, \langle \psi_L | O | \psi_L \rangle = 0.$$  

A stabilizer code $\mathcal{C}$ with distance $d$ encoding $k$ logical qubits onto $n$ physical ones is called a $[[n, k, d]]$-code.

The bit flip and the phase flip code both are considered as $[[3, 1, 1]]$-codes, as a single $Z$ operator on any qubit of the bit flip- or any single $X$ operator on the phase flip code already pose as such a minimum weighted nontrivial operator.

However, let us limit our considerations onto the intentionally protected degrees of freedoms of the bit flip code. In this frame, the crucial quantity will be the $X$-distance $d_X = 3$, corresponding to the minimum weight of a composite $X$-operator satisfying Definition 2. Any $X$-error $E : |\psi\rangle \mapsto |\psi\rangle = E|\psi\rangle$, which flips no more than one of these qubits, can be corrected by a simple majority vote on the given states. Let such an error occur with probability $p$ and let us assume, that these errors occur statistically independent from each other. As $p > p^2(1-p)$ (we assume single-qubit errors for example are statistically independent), if we find out that our system is in state $|001\rangle$, we would conclude that the error was $E = X_3$ rather than $E = X_1X_2$ and apply the correction operation to the third qubit. But in case that the actual error was the joint error on the first two qubits, this miscorrection together with the original error on the first two qubits has effectively performed the operation $X_1X_2X_3 = X_L$, generating a logical error.

This leads to the conclusion, that this code is only capable of correctly correcting bit flip errors with weight up to $wt(E) = 1$. As the general probability of two or more of the bits
being simultaneously flipped by an error in the bit flip code is \( p_e = 3p^2(1 - p) + p^3 \), we find that this encoding-scheme makes the storage of our information more reliable, when

\[
p_e = 3p^2(1 - p) + p^3 < p \iff p < \frac{1}{2} =: p_{\text{pseudothreshold}}.
\]

Furthermore, we can state generally:

**Theorem 2.** Any error \( E \) on a code with distance \( d \) with error-weight \( t := \text{wt}(E) \) is correctable, if the following inequality is satisfied:

\[
2t + 1 \leq d.
\]

This is given, since any miscorrection will not induce an extra error \( E_{\text{corr}} \) with \( \text{wt}(E_{\text{corr}}) > t \) if the correction scheme is based on comparison of error probabilities.

Thus, any correctable bit flip error maps our original codespace to one of the spaces described by the following bases:

- \( E = I : B_0 = \{ |000\rangle, |111\rangle \} \)
- \( E = X_1 : B_1 = \{ |100\rangle, |011\rangle \} \)
- \( E = X_2 : B_2 = \{ |010\rangle, |101\rangle \} \)
- \( E = X_3 : B_3 = \{ |001\rangle, |110\rangle \} \)

Finding out which of these bases describes our system—the localization of the error—suffices to enable us to correct it. Keeping in mind that a direct measurement in the \( Z \)-basis of all three qubits would destroy any superpositions, the task of the error localization is accomplished by measuring the parity via an ancilla qubit instead. A measurement outcome of \( Z_iZ_{i+1} = -1 \) would imply, that one of the two qubits targeted by the measurement has experienced a bit flip error. This occurrence will be called (error-)syndrome of the corresponding parity-check operator or stabilizer, respectively. Following this scheme, we observe a one-to-one correspondence between the (weight-1) error location and the syndromes (denoted as \( \times \)) as listed below.

<table>
<thead>
<tr>
<th>( E )</th>
<th>( B_i )</th>
<th>( S_1 = Z_1Z_2 )</th>
<th>( S_2 = Z_2Z_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I )</td>
<td>{</td>
<td>000\rangle,</td>
<td>111\rangle }</td>
</tr>
<tr>
<td>( X_1 )</td>
<td>{</td>
<td>100\rangle,</td>
<td>011\rangle }</td>
</tr>
<tr>
<td>( X_2 )</td>
<td>{</td>
<td>010\rangle,</td>
<td>101\rangle }</td>
</tr>
<tr>
<td>( X_3 )</td>
<td>{</td>
<td>001\rangle,</td>
<td>110\rangle }</td>
</tr>
</tbody>
</table>

To generalize this idea of measuring parity-check operators we define:

**Definition 3.** *(Stabilizer Group)* The stabilizer group is the abelian subgroup of the Pauli group on \( n \) qubits, which fixes the codespace.

\[
S := \{ S \in \mathcal{G}_n | S|\psi\rangle = |\psi\rangle \forall |\psi\rangle \in \mathcal{C} ; S^2 = I \}.
\]
Taking $S \in S$ as measurement operators —due to the unitarity and hermiticity of $S$ given by $S \in U(2^n)$ together with $S^2 = I$— we will always obtain a $+1$ measurement outcome for all stabilizers if the codespace is unaltered and as Pauli group elements always either commute or anticommute, $S E |\psi\rangle = -E S |\psi\rangle = -E |\psi\rangle$, a $-1$ outcome for those stabilizers incident to an error.

Furthermore, since we demand all logical states to be stabilized by $S$, it is necessary for all the stabilizers to commute with the logical operators. In the given example this is satisfied due to the fact that $\{Z_i, X_j\} = 0$ iff $i = j$ and else $[Z_i, X_j] = 0$:

$$Z_1 Z_2 |111\rangle = Z_1 Z_2 X_1 X_2 X_3 |000\rangle = (-1)^2 X_1 X_2 X_3 Z_1 Z_2 |000\rangle = X_1 X_2 X_3 |000\rangle = |111\rangle.$$

This notion also has to be held generally.

**Definition 4.** (Logical Operators) The set of logical operators $L_S$ of a code $C$ with stabilizer $S$ is the centralizer of $S$ excluding $S$: 

$$L_S := \{ L \in G_n \mid [L, S] = 0 \forall S \in S \} - S.$$ 

With this definition the distance of a stabilizer code is simply the minimum weight of any logical operator $L \in L_S$. It is sufficient to consider a minimal generating set to examine the actions of the stabilizer group 

$$S = \langle S_1, S_2, S_3, \ldots, S_r \rangle,$$

with all the generators satisfying the conditions given in Definition 3, as all $2^r$ resulting elements would share these properties. In the bit-flip code, the stabilizer group is given by $\langle Z_1 Z_2, Z_2 Z_3 \rangle$. The third nontrivial element of this group $S_1 S_2 = Z_1 Z_2 Z_3 = Z_1 Z_3$ is obviously redundant, as the error-localization is already perfectly achieved by only considering $S_1$ and $S_2$.

We can depict this stabilizer code by introducing three dots (vertices) representing the data qubits and two edges connecting them to symbolize the generators of the stabilizer group.

![Figure 3.2: The bit flip code. Qubits are represented by the red dots, the boundary of the blue edges make up the support of the Z-stabilizer operators, the green edge represents the logical $X_L$ operator, acting on all incident data qubits.](image)

The short blue lines $AB, BC$ represent the generators of the stabilizer group and act as $Z$ on their boundaries (the vertices). Their union, the long blue line $AB + BC = AC$,
represents in this interpretation due to $Z^3_b = I$ the third stabilizer $S_1 S_2 = Z_1 Z_3$.

In this illustration, both stabilizer-lines also include a small red dot, which represents an ancilla qubit. Note, that in future illustrations the ancilla qubit will not be depicted, but its presence taken as implicit. The ancilla qubits are the qubits whose measurements induces the effective measurement of the stabilizer and function as storage of the parity-check information. In this code, the measurement of the stabilizers consists of initializing the ancilla qubits in their $|0\rangle$ state, followed by two CNOTs, each taking one of the qubits lying on the boundary of the stabilizer as control and the ancilla qubit as target. Commuting the $Z$-measurement through the CNOTs effectively introduces the stabilizer operators acting on the data qubits.

The state measured is the bracketed state $|\psi\rangle = C_{b,m} C_{a,m} |0\rangle_m |\psi\rangle_{ab}$, but since the CNOTs take the data qubits as control, their state regarding the $Z$-basis remains unchanged and we obtain the desired $Z_a Z_b$-parity eigenvalue $Z_{ab} = \pm 1$ as measurement outcome. In this context, we will call the generated operator of $Z_m Z_a Z_b$ the \textit{effective measurement operator}.

Parity measurements in the phase $\{|+\rangle, |-\rangle\}$-basis are performed analogously. If we encode phase- instead of bit-information (e.g. via the encoding step presented on the right side in Fig. 3.1) the corresponding arrangement of stabilizers can be simply achieved by taking Fig. 3.2 and reinterpreting the blue lines as $S_1^+ = X_1 X_2$ and $S_2^+ = X_2 X_3$. The measurement is, due to Eq. (2.2), obtained by applying the CNOTs in the opposite direction.

Furthermore, the logical operator can also directly be constructed from Fig. 3.2. This is done by fixing a $X$-operator on one arbitrary vertex and adding $X$-operators on the opposite boundaries of every incident edge, such that in the end the achieved $X_L$ operator

---

3 In the picture of the CNOT acting as modular addition, both $|\psi\rangle_a$ and $|\psi\rangle_b$ being in the $|0\rangle$ respectively $|1\rangle$ state would correspond to no action on the ancilla qubit respectively flipping it twice, thus again obtaining effective trivial action.
shares two vertices with every generator of the stabilizer group (every blue line). Assume we extend this collection of vertices into a code with some higher number of data qubits $n$ and define periodic boundary conditions in terms of stabilizers connecting the qubits. Then in order to satisfy Definition 2, the only possibility for a logical $X_L$ operator on this one dimensional code structure would be a complete circuit, crossing each vertex (qubit) exactly once, such that again every stabilizer shares exactly two vertices with the circuit.

![Figure 3.3: A repetition code with $n = 12$ qubits.](image)

### 3.3 The Toric Code

The so far discussed one-dimensional bit flip and phase flip repetition codes separately allow encoding of states for protection against bit- and phase flips. However, in order for quantum computation to differ significantly from classical computation, a code protecting the state of a logical qubit must be able to safely inherit both phase- and bit degrees of freedom. A particularly interesting framework sharing this ability came with the introduction of the surface- or toric code by A. Yu. Kitaev [7] which we now will review briefly. Consider a $D \times D$ square lattice with periodic boundaries.$^4$

![Figure 3.4: The toric code.](image)

$^4$one can imagine the openly drawn boundaries wrapped around and attached to their parallel partner to form a torus.
Instead of the vertices, as in the previous discussed construction, the edges connecting two lattice-vertices will be considered as physical qubits\(^5\), of which there are \(n = \frac{4D^2}{2} = 2D^2\).\(^6\) The faces (plaquettes) and vertices of this square-tiling will be associated with the weight \(4 - S_Z\) and \(S_X\) generators of the stabilizer group acting on the qubits lying on their (co-)boundaries (e.g. the blue plaquette defines \(S_{Z,i} = Z_AZ_BZ_CZ_D\) and the green vertex \(S_{X,j} = X_EX_FX_GX_H\)).

It can be seen easily that all properties of Definition 3 are satisfied, as stabilizers of the same type (X or Z) and stabilizers not sharing any qubits always commute and any non-zero overlap between X- and Z-stabilizers is always of weight 2 (commuting the stabilizers through each other raises a factor \((-1)^2 = 1\)). This form of constructing the stabilizers gives rise to—in perfect analogy to the composition of stabilizers in the bit flip code—an important observation: Compositions of plaquette-stabilizers with nonzero overlap on their edge-sets always form bigger plaquettes, whose boundary again defines the Z-action of this composite stabilizer as depicted on the left of Fig. 3.5.

![Stabilizer composition & Dual space C⊥.](image)

The same observation can also be made for the X-stabilizers when one considers the dual code. In this geometric construction, the dual code can be obtained by replacing every vertex with a plaquette, every plaquette with a vertex and reorienting all the short edges, such that they again form a connection of two vertices as shown as dashed lines on the right side of Fig.3.5. With this at hand, one can see, that every loop of \(Z(X)\) operators on the (dual-)code \(C (C^⊥)\) which encloses a convex surface can be interpreted as an element of the stabilizer group. With this and the construction steps of Fig.3.3 for commutativity in

\(^5\)In this figure marked by the red dots, which in future illustrations will be left out.

\(^6\)This can be seen by multiplying the number of faces \(F = D^2\) by the number of edges per face \(r\) and thus effectively counting every qubit exactly twice. Getting rid of this factor thus yields \(n = \frac{Fr}{2}\).
mind, the only remaining possibilities for logical operators are those connected operator-lin es (defined such that they act as $Z(X)$ on every intersected qubit), which reach from boundary to boundary.

Since the blue-filled square is a stabilizer and as such acts trivially on the codespace,

$$Z_L = Z_1 Z_2 Z_6 Z_7 Z_8 Z_4 Z_5 \equiv (Z_3 Z_6 Z_7 Z_8)Z_L' = Z_1 Z_2 Z_3 Z_4 Z_5 = Z_L.$$

All paths connecting the upper and lower boundaries are equivalent and the logical $Z$-operator has a minimum weight of $wt(Z_L) = D = 5$. The same holds for the $X_L$ operators connecting the left and right boundary, which is also of minimum weight $wt(X_L) = D = 5$. In order for any path to connect two parallel boundaries, it will always intersect any path, which connects the two perpendicular boundaries an odd number of times. Thus, $X_L$ and $Z_L$ always overlap on an odd number of data qubits, necessarily yielding $\{X_L, Z_L\} = 0$.

It has already been found, that this code makes use of $n = 2D^2$ data qubits, yielding a total of $2D^2$ free parameters ($+1$ or $-1$ eigenvalues of the individual data qubits) on each

---

7One can imagine playing a game of snake, any even number of complete turnarounds just brings one back to the origin.

8$Z_L X_L$ thus always commutes to $(-1)^{2k+1} X_L Z_L$ for some $k \in \mathbb{Z}$, as different Pauli operators on the same qubit always anticommute.
the Pauli-\(Z\) and the Pauli-\(X\) basis. In order to determine the number of logical qubits it encodes, one has to examine the number of constraints the stabilizers induce. Due to the composition rule, multiplication of all \(F = \#\{Faces\}\) plaquette-stabilizers \(S_z\) form a big plaquette, which acts nontrivially on the boundary-set \(\partial S\) of the whole surface. However, due to the choice of periodic boundary-conditions, this boundary set is empty \(\partial S = \emptyset\). Thus, this composition rule implies a linear dependency of the \(F\) \(Z\)-stabilizers \(S_z\) and \(V = \#\{Vertices\}\) \(X\)-stabilizers \(S_x\)^9, namely

\[
I = \prod_{i=1}^{F} S_{z,i}, \quad I = \prod_{j=1}^{V} S_{x,j}. \tag{3.1}
\]

Due to these two restrictions, there are only \(r = F + V - 2 = 2D^2 - 2\) linearly independent stabilizers. Such that in total, this code inherits leftover degrees of freedom, which accounts for the number of logical qubits this code encodes.

Thus, the toric code is a \([[2D^2, 2, D]]\)-code with \(X\)- and \(Z\)- stabilizers of weight 4. The number of logical qubits shares a direct correspondence to the genus - the number of holes\(^{10}\) - \(g\) of the surface on which the given tiling (in this case squares) can be embedded, namely

\[
k = 2g. \tag{3.2}
\]

To properly understand why this is the case, we still lack a number of generalizations and tools which are to be postponed into the next chapter.

---

9 considering the dual code \(C^\perp\).

10 this will be properly defined in the next Chapter.
4 Homological Code Construction

Before exploring more sophisticated code designs, it is important to keep in mind, that generally the geometric properties of the such defined codes don’t necessarily directly correspond to the geometric properties of the device in which the data qubits are implemented. These codes rather design a complex entanglement-structure between the data qubits, giving rise to the logical ones.

The previously discussed bit flip- and surface code have been constructed in a particularly similar way. In the bit flip construction, we associated the data qubits with the 0-dimensional vertices and the stabilizers with 1-dimensional edges. The surface code gave rise to data qubits, which were associated with 1-dimensional edges while the Z-stabilizers were identified by 2-dimensional faces (0-dimensional vertices on the dual code) and the X-stabilizers with 0-dimensional vertices (2-dimensional faces on the dual code). The generalization of these ideas can be rigorously captured by the notion of $\mathbb{Z}_2$-homology as intensively discussed by Nikolas Breuckmann and Barbara M. Terhal in [2] and are to be reviewed in this chapter.

Consider a regular-tiled surface, denoted as $T_{r,s}$. $r$ denotes the number of edges incident to each face of the tiling, $s$ the number of edges meeting at a vertex. The notion of regularity denotes, that every face and every vertex inherits these properties and is, in this notation, fixed by the usage of the Schläfli-symbol $\{r,s\}$. For instance, the surface code was constructed upon a simple square-grid, which is a $\{4,4\}$-tiling.

Figure 4.1: A fragment of a $\{6,3\}$ tiled surface and its dual tiling $\{3,6\}$.

The regular tiling of the surface introduces faces, edges and vertices, which will later on be referred to as 2-cells, 1-cells and 0-cells. For an i-cell, i will also be called the cell dimension and $T$ a cell-complex. The dual tiling $T^\perp$ can be obtained in analogous fashion as discussed for the toric code in Fig. 3.5 and is $\{s,r\}$-regular.

One can start to identify objects given by this tessellation in similar fashion as in the construction of the toric code. The linearly independent set of $E = \dim(C_1)$ 1-cells (edges)

\[ \mathbb{Z}_2 = \mathbb{Z} \mod 2 = \{0,1\}. \]
are designated to become associated with the data qubits, the $F = \dim(C_2)$ 2-cells (faces) with the Z-Stabilizers and the $V = \dim(C_0)$ 0-cells (vertices) with the X-stabilizers. But before diving too deep into the proper homological examination of such code, we can first try to answer the question of how the genus relates to the number of encoded logical qubits as stated in Eq. (3.2) while reviewing some fundamental concepts of topology.

**Definition 5. (Genus)[16]** The genus of a surface is the maximum number of cuts along non-intersecting simple loops on a surface, which can be performed without disconnecting the surface. It is equivalent to the number of handles a (potentially) multihandled torus has.

In the example of the torus folded from the surface with two periodic boundaries — one cut along a simple loop leaves us with a cylinder without top- and bottom-cover, which is still connected and performing a second such cut will disconnect this surface — and the sphere $g = 0$ where any cut along a closed curve leaves us with two disjoint fragments.

![Figure 4.2: Tori with genus $g = 1, 2, 3$ (left to right).](image)

**Definition 6. (Homeomorphism)[15]** Let $X_1$ and $X_2$ be topological spaces. A homeomorphism is a continuous and invertible map $f : X_1 \mapsto X_2$. Two topological spaces related by a homeomorphism are called homeomorphic.

Graphically, a homeomorphism between two objects can be viewed as continuous deformation of one into the other — only stretching and compressing but no tearing or hole-poking is allowed. The genus of a topological space is thus initially defined to be invariant under homeomorphism.

For example, the coffee-mug and the torus are homeomorphic, just as every convex solid is homeomorphic to a sphere ($g = 0$).

Naturally, the question of how homeomorphic equivalence classes can be characterized arises. This characterization can be performed by considering topological invariants —
quantities constant under homeomorphisms by which objects can be labeled. For instance, the genus is already defined to be a topological invariant \[15\].

**Theorem 3.** *(Euler-Characteristic)* [15] Let \( n \) be the maximal cell dimension of a cell-complex \( T \) and let \( k_i \) be the number of linearly independent \( i \)-cells, then the Euler-characteristic of the cell-complex \( T \)

\[
\chi(T) := \sum_{i=0}^{n} (-1)^i k_i
\]

is a topological invariant.

With this definition a cell-complex \( T = T(F, E, V) \) with \( n = 2 \) as previously considered has an Euler-characteristic equal to

\[
\chi(T) = F + V - E. \tag{4.1}
\]

**Theorem 4.** The Euler characteristic for an orientable\(^2\) surface \( S \) with genus \( g \) is:

\[
\chi(S) = 2 - 2g,
\]

and for a non-orientable one, one has

\[
\chi(S) = 2 - g.
\]

The proofs of Theorems 3 and 4 can be revisited in [18].

**Theorem 5.** A tiled, orientable surface \( T = T(F, E, V) \) with genus \( g \), \( F \) faces representing Z-stabilizer, \( V \) vertices representing X-stabilizer and \( E \) edges representing data qubits encodes \( 2g \) logical qubits.

**Proof.** The number of logical qubits \( k \) equals the number of parameters minus the number of linearly independent restrictions on these parameters. Namely:

\[
k = n - r = E - (F + V - 2).
\]

The term of ’-2’ arises due to Eq. (3.1). With Eq. (4.1) and Theorem 4 this reads

\[
k = E - (\chi(T) + E - 2) = 2 + 2g - 2 = 2g.
\]

Analogously, a tessellation of a non-orientable surface with genus \( g \) would encode \( k = g \) logical qubits.

\(^2\)Orientable means, that it is possible to make a consistent choice of a normal vector on this surface.
4.1 $\mathbb{Z}_2$-Homology

The set of all $i$-cells for fixed $i$ on $T$ introduces a vector space $C_i$ which can be identified with $\mathbb{Z}_2^{m_i}$ with $m_i$ being the number of $i$-cells contained in $T$ ("identify" means, that these spaces are found to be isomorphic, namely $C_i \sim \mathbb{Z}_2^{m_i}$). The elements from $C_i$ are called $i$-chains.

The boundary operator

$$\partial_i : C_i \mapsto C_{i-1}$$

is defined to map an $i$-cell onto all $(i - 1)$-cells on its boundary. E.g. a face is mapped to the edgset surrounding this face, an edge is mapped to its two incident vertices:

$$\partial_2 : \begin{array}{c}
\begin{array}{c}
\text{+} \hspace{1cm} \text{+}
\end{array}
\end{array}
\mapsto
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
$$

Figure 4.3: Boundary operator action.

Analogously, the coboundary operator

$$\delta_i : C_i \mapsto C_{i+1}$$

maps $i$-cells onto all incident $(i + 1)$-cells, which is equivalent to the boundary operator $\partial_{i+1}$ acting on the dual Code $C^\perp$:

$$\delta_0 : \begin{array}{c}
\begin{array}{c}
\text{+} \hspace{1cm} \text{+}
\end{array}
\end{array}
\mapsto
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
\end{array}

\delta_1 : \begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
\mapsto
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{+}
\end{array}
\end{array}
$$

Figure 4.4: Coboundary operator action.

Viewed in form of matrices acting on $C_i$, this relationship (left side of Fig. 4.4) is expressed by

$$\delta_i = \delta^T_{i+1} \Leftrightarrow \partial_i = \delta^T_{i-1}$$

(4.2)

as discussed in [2]. Per definition of the boundary operator, an $i$-chain that can be written as a boundary of some higher-dimensional chain has no boundary itself and with Eq. (4.2) this can also be turned into the corresponding statement about the coboundary operator (analogue on the dual code):

$$\partial_i \circ \partial_{i+1} = 0 \Leftrightarrow \text{im}(\partial_{i+1}) \subseteq \text{ker}(\partial_i)$$

$$\partial_i \circ \partial_{i+1} = \delta^T_{i-1} \circ \delta^T_i = (\delta_i \circ \delta_{i-1})^T$$

$$\Rightarrow \delta_i \circ \delta_{i-1} = 0 \Leftrightarrow \text{im}(\delta_{i-1}) \subseteq \text{ker}(\delta_i).$$
Relevant for this discussion are the notions of the cycle space $Z_1 = \ker(\partial_1)$ — consisting of all closed loops in $T$ —, the boundary space $B_1 = \text{im}(\partial_2)$ — consisting of all loops, which can be written as the boundary of a collection of plaquettes — with $B_1 \subseteq Z_1$, and their analogue on the dual $T^\perp$: the coycle space $Z^1 = \ker(\delta_1)$ and the coboundary space $B^1 = \text{im}(\delta_0)$ with $B^1 \subseteq Z^1$.

Since the $C_i$ span vector spaces, the idea of having the notion of a scalar product in each $C_i$ becomes relatively intriguing. This is simply defined in terms of the $Z^m_i$ componentwise multiplication:

$$\langle a, b \rangle_{C_i} = \sum_j a_j \cdot b_j, \quad a, b \in Z^m_i \sim C_i. \quad (4.3)$$

With this notion and Eq. 4.2 we can identify

$$\langle \delta a, b \rangle_{C_{i+1}} = \langle a, \partial_{i+1} b \rangle_{C_i} \quad (4.4)$$

$$\Rightarrow \langle \delta_0 a, b \rangle_{C_1} = \langle a, \partial_1 b \rangle_{C_0} \quad (4.5)$$

and

$$\langle \delta_1 a, b \rangle_{C_2} = \langle a, \partial_2 b \rangle_{C_1}. \quad (4.6)$$

Thus, with $\bar{a} = \delta_0 a \in B^1 = \text{im}(\delta_0)$, if $b \in \text{im}(\partial_2) = B_1$, Eq. (4.5) yields

$$\langle \bar{a}, b \rangle_{C_1} = 0,$$

implying, that the coboundary space is the orthogonal complement of the cycle space (as this hold for all $a$ and $b$).

Similarly, if one takes $a \in Z^1 = \ker(\delta_1)$ for Eq. (4.6), while setting $\bar{b} = \partial_2 b \in B_1$, one obtains $\langle a, \bar{b} \rangle_{C_1} = 0$, thus the cocycle space is the orthogonal complement of the boundary space.

To summarize, we have found the relationships

$$B^1 = Z_1^\perp \subseteq Z^1 \quad \text{and} \quad Z_1 = B_1^\perp \supseteq B^1. \quad (4.7)$$

With these fundamental considerations, we can continue to construct a quantum error correcting code on this $\{r, s\}$ regular tiled surface $T$.\(^3\)

As already mentioned, the $\text{dim}(C_1) = n$ edges are identified with the data qubits. The coboundary space $B^1$ (the edge set incident to the vertices) can be associated with qubits acted upon by the vertex-X-stabilizer $S_X$ and the boundary space $B_1$ (the edges surrounding each face) with the qubits acted upon by the face-Z-stabilizer $S_Z$. Thus, when asking the question of how many qubits such a code $C(T)$ can encode, we can again subtract the

\(^3\)Note that $T$ refers to the cell-complex embedded on this surface, not the surface itself.
numbers of the data qubits by the number of restrictions posed onto them:

\[ k = n - r = \dim(C_1) - \dim(B^1) - \dim(B_1) = \dim(C_1) - \dim(Z_1^+) - \dim(B_1) = \dim(C_1) - (\dim(C_1) - \dim(Z_1)) - \dim(B_1) = \dim(Z_1/B_1) = \dim(H_1). \]

In the first line Eq. (4.7) has been used, in the last line the first homology group has been defined as all closed loops, which cannot be expressed as the boundary of some face on \( T \): \( H_1 := Z_1/B_1 \). When remembering how the logical operators for the surface code has been constructed, one sees, that these were exactly those kind of operators. For a torus, the dimension of the first homology group \( \dim(H_1) = 2 \) reproduces our previous observations from the surface code (on the surface with periodic boundaries there are only two linearly independent closed curves not enclosing any collection of faces, one from the upper to the lower boundary and one from the left to the right).

To summarize, this form of consideration not only doesn’t require the extra terms induced by linear dependency of the stabilizers when calculating the number of logical qubits (e.g. the ’\(-2\)'-term in the proof of Theorem 5 as the constraints are now given by the dimension of the (co-)boundary space), but also offers explicit identification of the logical operators and holds for general cell-complex-constructions. The correspondence between the code \( C \) and the regular tiled surface \( T_{r,s} \) is listed in Tab. 4.1.

![Figure 4.5: Elements from \( H_1 \) on the torus](image)

<table>
<thead>
<tr>
<th>( T_{r,s} )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 )</td>
<td>data qubits</td>
</tr>
<tr>
<td>( \dim(C_1) )</td>
<td>( n )</td>
</tr>
<tr>
<td>( B_1 )</td>
<td>( S_Z )</td>
</tr>
<tr>
<td>( B^1 )</td>
<td>( S_X )</td>
</tr>
<tr>
<td>( r )</td>
<td>( \dim(B_1) + \dim(B^1) )</td>
</tr>
<tr>
<td>( \mathcal{L}_Z )</td>
<td>( H_1 )</td>
</tr>
<tr>
<td>( \mathcal{L}_X )</td>
<td>( H^1 )</td>
</tr>
<tr>
<td>( k )</td>
<td>( \dim(H_1) )</td>
</tr>
</tbody>
</table>

Table 4.1: Code-Homology correspondence.

---

\[ \text{This is called the quotient space of } Z_1 \text{ and } B_1 \text{ and basically states “all elements of } Z_1 \text{ up to equivalence by elements from } B_1”. \]
5 Hyperbolic Surface Codes

5.1 The Gauss-Bonnet Theorem

The homological code construction opens up a general way to use regular tilings defined on arbitrary connected surfaces with periodic boundary conditions to define a quantum code. However, the questions arise, which \( \{r, s\}\)-tiling and what kind of surface offers the most beneficial properties for a quantum error correcting code to have — for instance a large genus is desirable as \( k = 2g \). To first examine the properties of the surface, we need some further insights.

**Definition 7.** (Gaussian Curvature) The Gaussian Curvature \( K \) in a point \( P \) on a surface is defined to be the product of the inverse minimum and maximum valued radii \( R_1 \) and \( R_2 \) of arcs spanned by a fixed point on a normal vector on \( P \)

\[
K = \frac{1}{R_1 R_2}.
\]

We can always rescale the surface (respectively the object of which our surface describes the boundary, as we only consider surfaces with periodic boundary conditions) such that \( K \) is either \(-1\), \(0\) or \(1\) respective of the general topology of the surface.

The plane with \( K = 1 \) is called *spherical* \( S^2 \), the (locally flat) one with \( K = 0 \) *euclidean* \( E^2 \) and the saddle-like one *hyperbolic* \( H^2 \).
The main difference between those planes can be found in their angular defect — the inner angular sum of polygons embedded on such surface varies with its curvature. For instance, geodesic triangles (the boundary edges are chosen as the shortest paths between the nodes) drawn on the euclidean plane admit an inner angular sum of $\pi$, while it is slightly bigger for a triangle on a sphere and smaller if its embedded on the hyperbolic plane. The same accounts implicitly for the surface of this triangle.

![Figure 5.3: Triangles on curved surfaces [22].](image)

The Gauss-Bonnet-Theorem establishes this connection between the geometry and the topology of general surfaces. As the complete proof and all its impacts reaches far beyond the frame of this thesis, we shall only examine a few relevant aspects of it. The more curious reader is referred to [21] and [23].

**Theorem 6. (Gauss-Bonnet)[23]** Let $M$ be a closed, compact and orientable surface in $\mathbb{R}^3$, then

$$\int_M K dS = 2\pi \chi(M)$$

with the $\chi(M)$ being the Euler-Characteristic of this surface.

As established in Theorem 4 in Chapter 4, this directly implies a connection between the curvature of the surface and its genus $g$,

$$\int_M K dS = K \text{area}(M) = 2\pi(2 - 2g).$$

Since we are interested in regular $\{r,s\}$-tilings on such surface with periodic boundary conditions, we limit our considerations on surfaces with constant curvature. Thus, as $K \sim -g$, it becomes apparent that it is the hyperbolic surfaces with $K = -1$, which promise themselves as interesting when it comes to code construction.

As shown in [21], the Gauss-Bonnet Theorem further yields:

**Theorem 7. (Gauss-Bonnet-Theorem for a hyperbolic triangle)** Let $\Delta$ be a triangle embedded on a hyperbolic surface with internal angles $\beta_i, i = 1, \ldots, 3$. Then

$$\text{area}(\Delta) = \pi - \sum_{i}^{3} \beta_i.$$
Thus, as we can simply cut every \( r \)-gon into \( r - 2 \) triangles\(^1\), we directly obtain the area of an general \( r \)-gon \( P_r \) on this surface:

\[
\text{area}(P_r) = \sum_{j=1}^{r-2} \text{area}(\Delta_j)
\]

\[
= (r - 2)\pi - \sum_{i=1}^{3(r-2)} \beta_i
\]

\[
= (r - 2)\pi - \sum_{k} \alpha_k
\]

(5.4)

with \( \alpha_k, k = 1...r \) as the inner angles of that \( r \)-gon. The transition from line 2 to 3 follows from the triangulation as can be seen in following Fig. 5.4.

![Figure 5.4: Proof of Eq. (5.4).](image)

With these ideas in place, we can proceed to determine the encoding rate \( k/n \) of a code defined via a regular tessellation \( T_{r,s} \) with \( F \) \( r \)-gons \( P_r \) as faces on such a surface by inserting \( K = -1 \) in Eq. 5.2:

\[
(-1) \cdot \text{area}(T_{r,s}) = 2\pi(2 - 2g)
\]

\[
\iff k = 2 + \frac{\text{area}(T_{r,s})}{2\pi}
\]

\[
\iff = 2 + \frac{F \cdot \text{area}(P_r)}{2\pi} \quad \quad \quad n = \frac{Fr}{2}
\]

\[
\iff = 2 + n - \frac{2n}{r} - \frac{2n}{s}
\]

\[
\frac{k}{n} = 1 + \frac{2}{n} - 2 \left( \frac{1}{r} + \frac{1}{s} \right).
\]

(5.7)

\(^1\)as the number of edges of a \( r \)-gon equals the number of its nodes, we can prove by induction: \( r = 3 \) is trivial, \( r \rightarrow r + 1 \) follows from considering the newly added node with its two adjacent nodes as the new triangle.
In Eq. (5.6), (5.4) has been inserted using $\alpha_k = \frac{2\pi}{s} = \text{const.}$ as implied by the regular tiling as illustrated for $\{r, s\} = \{6, 3\}$ in Fig. 5.5 below.

![Figure 5.5: Inner angles of a \{6,3\} tiling.](image)

Thus, as every polyhedron can be decomposed into triangles as depicted on the right side of Fig. 5.5, due to the hyperbolic angular defect, we obtain

\[
\frac{\pi}{r} + \frac{\pi}{s} < \frac{\pi}{2},
\]

\[
\Leftrightarrow \frac{1}{r} + \frac{1}{s} < \frac{1}{2}. \quad (5.8)
\]

Inserting this into Eq. (5.7), it is noticeable that the encoding rate of codes defined via tessellations of hyperbolic surfaces is asymptotically ($n \to \infty$) positively constant! This, for instance stands in contrast to the toric code where the number of logical qubits remains constantly $k = 2$ while the number of physical qubits needed grows quadratically with the distance (the level of protection offered by this code).

Furthermore, as there is also an infinite\(^2\) amount of possible $\{r, s\}$ tilings admitting Eq. (5.8), this definitely spans an interesting class of codes to be examined more closely.

\(^{2}r \text{ and } s \text{ can be arbitrarily large.}\)
The homological codes defined via those \( \{r, s\} \)-tiling are examples of Low-density-parity-check (LDPC) codes, meaning that the parity-check operators of this code involve only a low, constant amount of qubits and each qubit participates only in a constant amount of parity checks.

The curious reader might ask why the \( \{6, 3\} \) tiling is constantly used as an example throughout this thesis. This becomes clear when the \( r, s \) solutions of \( 1/r + 1/s = 1/2 \) are computed. This equation only has integer solutions for \( \{r, s\} = \{4, 4\}, \{6, 3\} \) and \( \{3, 6\} \). Thus, there are actually not many tilings to embed nicely on the flat surfaces of these pages.
5.2 The Small Stellated Dodecahedron as a Hyperbolic Surface Code

The smallest examples of hyperbolic codes turn out to correspond to some non-convex polyhedra. Due to their polyhedral structure, these naturally admit a set of edges, vertices and faces consistent with the discussed $\mathbb{Z}_2$ homology. A prominent class of non-convex polyhedra are the four stellations of the regular dodecahedron. This thesis mainly examines the code gained from the small stellated dodecahedron which is composed of 12 pentagrammic faces. Its dual polyhedron is the great dodecahedron. The parameters are as listed below in Fig. 5.7.

\[
\begin{array}{c|c}
F & 12 \\
V & 12 \\
E & 30 \\
\chi = F + V - E & -6 \\
g = (2 - \chi)/2 & 4 \\
\{r, s\} & \{5, 5\}
\end{array}
\]

Figure 5.7: The small stellated dodecahedron.

\[
\begin{array}{c|c}
F & 12 \\
V & 12 \\
E & 30 \\
\chi = F + V - E & -6 \\
g = (2 - \chi)/2 & 4 \\
\{r, s\} & \{5, 5\}
\end{array}
\]

Figure 5.8: The great dodecahedron.

As can be seen from the invariance of the Euler characteristics in Fig. 5.7, this polyhedron is homeomorphic to a 4-handled torus. Due to Theorem 5, the corresponding code encodes $k = 2g = 8$ logical qubits into $n = E = 30$ data qubits. The encoding rate is thus
$\frac{k}{n} = \frac{4}{15} \approx 26.7\%$. Associating the pentagrammic faces with the Z-stabilizer generators and the vertices with X-stabilizer generators, we find, there are 12-stabilizer generators of each type. All stabilizers are of weight $wt(S) = 5$.

One can construct the small stellated dodecahedron by taking a regular dodecahedron and extending all its edges simultaneously until they meet. Each face of the original dodecahedron then becomes one of the small stellated one.

The labeling of the vertices as introduced in Fig. 5.9 will be used whenever explicit reference to the stabilizers or qubits is appropriate. Both types of stabilizer-operators will be labeled by $i = 0, \ldots, 11$, referring to the vertex (X-stabilizer) or the face underlying this vertex (Z-stabilizer). The edges (qubits) will be referred to as ordered tuples $(i, j)$, $i < j$ of the vertices it connects.

The distance of this code is $d = 3$. This can be seen from explicitly constructing minimal-weight logical operators which we will do in the following.
**Logical Z-operators $Z_L$**  The logical Z-operators are most easily found when the vertex connectivity of the small stellated dodecahedron is viewed from a more disentangled point of view as depicted in Fig.5.10.

Figure 5.10: Disentangling the Graph $G = (V, E)$ of the small stellated dodecahedron. Each vertex (blue or red) corresponds to a X-stabilizer acting on all incident edges.

One sees that the underlying graph which only incorporates the edges and vertices of the small stellated dodecahedron but not its faces, is in fact the graph of the regular icosahedron. The original faces (or more precisely — the boundary set of the original faces) can still be found as the union of those edges, which connects all five neighbors of one specific node. In this case it is the node lying on the opposite side which labels the corresponding Z-stabilizer (this can be understood from the disentangling process).

To construct the logical $Z_L$ operators, it is thus necessary to identify those Z-type operators, which commute with all these X-stabilizers or equivalently -share an even overlap in their support (the edges). From Fig. 5.10 the minimal-weight candidates with this property can be easily found as those weight-3 $Z$-operators, which have their support on the small triangles spanned by three neighboring vertices.
There are 20 of those triangles in the graph of the small stellated dodecahedron, but due to their nontrivial overlap, this is a linearly dependent, overcomplete set. One could e.g. multiply 5 logical Z-operators and obtain a Z-stabilizer as product as depicted in the right side of Fig. 5.11.

\[
\prod_{i=1}^{5} Z_{L,i} = S_z.
\]

Figure 5.11: Logical operator $Z_L$. 
Logical X-operators $X_L$  Since the faces of the small stellated dodecahedron stand in one to one correspondence to those of the regular dodecahedron, it is equivalent to construct the X-logicals operators upon this. The edges corresponding to the support of these logical operators have to be chosen such that they establish an even overlap with the boundary sets of all faces. The easiest way to find these is by considering the dual graph of the dodecahedron — the icosahedron. The triangles of the such obtained icosahedral graph (as on the left in Fig. 5.11) can thus be associated with the minimum weight X-logical operators, with each of the triangles corresponding to one of the vertices of the dodecahedron.

This gives 20 candidates for weight-3 logical X-operators (one for each node of the regular dodecahedron). Note that these, just as the logical $Z_L$, are still linearly dependent from each other.

One could —e.g. for explicit readout of the logical states— fix a minimal set of logical operators by choosing 8 linearly independent $Z_L$ from the 20 candidates, and choose corresponding $X_L$s in analogous fashion such that each $Z_L$ and $X_L$ overlap on exactly one qubit to fulfill the canonical commutation-relations

$$\{Z_{L,i}, X_{L,i}\} = 0,$$
$$[Z_{L,i}, X_{L,j}] = 0 \quad i \neq j.$$  

As this code is not directly examined on the surface on which it is embedded, the linear dependency between the logical operators is not as obvious as in the case of the toric code. However, it can be checked that any weight-3 logical operator (as well as any logical operator with higher weight) can be constructed as a product of the 8 fixed weight-3 logical operators and corresponding stabilizers. This can be either done numerically or via following consideration (for the $Z_L$. The set of $X_L$ can be examined analogously):
Choosing the blue and red triangles on the top and bottom of Figure 5.13 as fundamental logical set, it becomes obvious that the 5th, uncolored triangle on the top and bottom can be directly obtained by multiplying the four blue or red with the corresponding maximally overlapping stabilizer. Any triangle on the central “belt” can be written as the product of a stabilizer, two triangles from the top and two further triangles on this central “belt”. To compose these two triangles on the central belt as a product of stabilizer and logicals operators, one needs to notice that the depicted trapezoids (right in Fig. 5.13) can be composed from two triangles from the top or bottom and one overlapping stabilizer.

Table 5.1: Example of minimal set of logical operators. $P_{\text{edge}}$ acts with $P$ on the qubit on the indexed edge.

<table>
<thead>
<tr>
<th>$Z_L$</th>
<th>$X_L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z_{(0,10)}Z_{(0,7)}Z_{(7,10)}$</td>
<td>$X_{(0,10)}X_{(2,4)}X_{(1,3)}$</td>
</tr>
<tr>
<td>$Z_{(0,11)}Z_{(4,11)}Z_{(1,4)}$</td>
<td>$X_{(1,11)}X_{(7,9)}X_{(8,10)}$</td>
</tr>
<tr>
<td>$Z_{(4,11)}Z_{(2,11)}Z_{(2,4)}$</td>
<td>$X_{(4,11)}X_{(6,8)}X_{(7,10)}$</td>
</tr>
<tr>
<td>$Z_{(2,11)}Z_{(5,11)}Z_{(2,5)}$</td>
<td>$X_{(2,11)}X_{(8,10)}X_{(6,9)}$</td>
</tr>
<tr>
<td>$Z_{(3,11)}Z_{(5,11)}Z_{(3,5)}$</td>
<td>$X_{(3,11)}X_{(6,9)}X_{(7,10)}$</td>
</tr>
</tbody>
</table>

A last and important observation that can be made from the illustrations above is, that none of the possible minimal-weight logical operators share an overlap with any stabilizer of the same type of weight greater than one. This can also be explicitly checked numerically and will later become a crucial feature of this code when examining its properties of fault tolerance.
6 Parity Check Scheduling

When one actually wants to measure the parity of the stabilizers, it is generally not possible to do this at once (measure all stabilizers simultaneously). This is simply due to the practical impossibility of using one qubit as control or target for two CNOTs at the same time.

Naturally, it is always desired to minimize the total time $T$ the circuit implementing the stabilizer needs to perform all necessary CNOTs, as every gap in the circuit allows for disturbance due to external noise (assuming perfect gates).

An instance, in which all the stabilizers of the code are measured will be called parity check cycle or simply a round of the error-correction procedure.

Figure 6.1: Weight 5 Z-parity check cycle with noise $N$. 

|0\rangle_{m,a} \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} N \quad N \quad N \quad N \\ N \quad N \quad N \quad N \\ N \quad N \quad N \\ N \quad N \quad N \quad N \\ N \quad N \quad N \quad N \quad N \end{array} t : 1 \quad 2 \quad 3 \quad 4 \quad 5

An instance, in which all the stabilizers of the code are measured will be called parity check cycle or simply a round of the error-correction procedure.
6.1 Separate Scheduling

Our first step consists of separately scheduling the parity check cycle of a code defined on a
tiled surface for X- and Z-stabilizers. This will be achieved by defining a graph and finding
a vertex coloring.

Definition 8.

1. A Graph $G = (V, E)$ is a set of nodes\(^3\) $V$ and edges $E$, such that every edge connects
two nodes. The total number of nodes is denoted as $|V|$, the total number of edges
$|E|$.

2. Two nodes $i, j$ are called adjacent if they are connected by an edge, meaning $(i, j) \in
E$.

3. A directed Graph is a Graph $G = G(V, E, \sigma)$ with a set of directions $\sigma \in \{-1, 0, 1\}^{\lvert E \rvert}$
on its edges.

4. A vertex-coloring of a graph $G$ is a map $C : i \in V \mapsto C(i) \in \{1, ..., n\}$ such that
$C(i) \neq C(j)$ if $(i, j) \in E$. No two adjacent nodes share the same color.

5. A vertex-coloring of a directed graph $G(V, E, \sigma)$ is a coloring of $G(V, E)$ which addi-
tionally satisfies $C(i) \leq C(j)$ if $(i, j) \in E$ and $\sigma((i, j)) = \pm 1$.

6. The chromatic number or chromatic index $\chi(G)$ is the minimal number of colors
required to define a vertex-coloring of the graph $G$.

7. The degree $\delta(i)$ of a node is the number of edges incident to this node.

8. The degree $\Delta(G)$ of a graph is the maximal degree of this graph $\Delta(G) = \max_{i \in V \subseteq G} \delta(i)$.

With Def. 8.1 and 8.4 it becomes clear, that the problem of scheduling the parity-check
cycle—first illustrated for the Z-stabilizers—can be viewed as the problem of finding a
minimal vertex coloring of the following graph in Fig. 6.2:

\(^3\)In future, we will often speak of “nodes” instead of “vertices” in the context of graph-colorings as we
are already using the term “vertices” for the 0-cells of the homological construction.
On the left side of Fig. 6.2, the CNOTs of one Z-stabilizer are outlined. The red dot indicates the ancilla qubit of the corresponding face, the data qubits that this Z-stabilizer acts on, are represented by the 6 edges on the boundary of this face. It is neither possible to simultaneously perform all the highlighted CNOTs, nor can any pair of CNOTs acting on the same edge be activated at the same time. Transforming this into a coloring problem, we setup a new scheduling graph $P_Z$. In $P_Z$ we define nodes indicating the activation of the corresponding CNOT between the edge and the face. Nodes incident to the same qubit-edge are connected by an edge (depicted in blue) and nodes belonging to the same stabilizer are also connected by an edge (depicted in red) in $P_Z$. Thus, a coloring of this graph $P_Z$ will directly yield the parity-check schedule of this code without violating the discussed constraints by interpreting the color of the node as the time step $t$ in which it is activated.

In our example this can be done in analogous fashion for the X-stabilizers.
Figure 6.3: Scheduling graph for X-stabilizers $P_X$.

From this point of view, the minimization of the overall time $T$ of one parity-check-cycle corresponds to the minimization of colors used by the vertex-coloring. It can be seen, that for the Z-stabilizer schedule, $T \geq 6$ and for the X-stabilizer schedule $T \geq 3$ are necessary. This is due to the red subgraphs consisting of 6 or 3 nodes which are all adjacent to each other. These subgraphs are called complete graphs or cliques.

**Definition 9.**

1. A graph consisting of $n$ nodes is called a complete graph $K_n$, if all nodes are adjacent to each other.

2. A subgraph $C \subseteq G$ of a graph $G$, which is complete, is called clique.

As all nodes in $K_n$ are adjacent to each other, is it trivially implied, that the chromatic number of a graph $G$ cannot be smaller than the size of the largest clique.

$$\chi(G) \geq n : K_n \subseteq G.$$ 

However, a valid question is if the chromatic number can be bounded by some constant independent of the size of the underlying tiled surface. It would be a very dreadful trade-off if every data qubit added to the code—in order to safely encode logical qubits—also led to a higher error rate on these. Luckily, this is not the case, as given by Brooks’ theorem.

**Theorem 8.** (Brooks’) A graph $G$ with degree $\Delta$ admits a vertex-coloring with $\chi(G) \leq \Delta + 1$.

**Proof.** Proof by induction over the number of nodes $n$ of $G$. 

35
• $n = 1$: This graph is colored by one color, thus $\chi = 1$. As there is no edge, $\Delta = 0$. Thus, $1 \leq 1$ is true.

• $n \to n + 1$: Let $G_{n+1}$ have degree $\Delta$. Remove one node with maximal degree $\Delta$ as well as all incident edges. The graph $G' = G - \{n+1\}$ is inductively $\Delta + 1$-colorable. Adding back the removed node and its incident edges, there is one color available to color it as the node is only adjacent to $\Delta$ nodes of $G'$.

Thus, for the graphs illustrated in Fig. 6.2 or 6.3, the chromatic number is upper bounded by $\chi(P_z) \leq 7$ or $\chi(P_x) \leq 4$.

From the generic way the coloring graphs are constructed from these homological codes defined on $\{r, s\}$-tiled surfaces, one can thus observe that every node is incident to $n - 1$ edges of the red complete graphs $K_n$ ($n = r$ or $s$) and one blue edge, thus we obtain

$$\Delta(P_z) = r \quad \Rightarrow \quad T^{\text{min}}_Z \in \{r, r+1\}$$

and

$$\Delta(P_x) = s \quad \Rightarrow \quad T^{\text{min}}_X \in \{s, s+1\}$$

and hence:

$$T^{\text{sep}}_{X/Z}(r + s, r + s + 2).$$

We can also see that the small stellated dodecahedron admits $T^{\text{min}}_{X/Z} \in \{5, 6\}$ and thus an overall minimal time to perform all the parity-checks in $T^{\text{min}}_{X/Z} \in \{10, 12\}$. The corresponding graphs $P_z$ and $P_x$ can be illustrated as in following Fig. 6.4

![Figure 6.4: Scheduling graphs $P_z$ (left) and $P_x$ (right). The pentagrammic cliques $K_5$ (red) induce a lower bound on any coloring of this graph.](image)

Of course, having bounds for a minimal coloring doesn’t yet give us this coloring. The explicit coloring of the graph can be obtained by various algorithmic approaches. In this work, the coloring is mainly done by Linear-Integer-Programming (ILP). Brute-force searches for
later issues have been conducted with a greedy algorithm. These approaches are outlined in Appendix A.

By ILP, a 5-coloring of $P_Z$ and $P_X$ could be obtained, such that a sequential application of all X- and the Z-stabilizers can be performed in a total of $T(X + Z) = 10$ time steps. The result of the coloring is shown in Fig. 6.5 below (without the edges) and the corresponding circuit for the Z-checks in Fig. 6.6.

Figure 6.5: 5-coloring of $P_Z$ (left) and $P_X$ (right).
Figure 6.6: $T = 5$ Parity-Z schedule. The red separated segments indicate each timestep, such that all CNOTs in one segment can be performed simultaneously.
6.2 Interleaved Scheduling

With a time of $T^{\min}(X + Z) \leq r + s + 2$ for sequential X- and Z-stabilizer action, the obvious question is, if it is possible to gain a further speedup for the circuit if the X- and Z-stabilizers are not performed separately. This problem also can be formulated in terms of a graph coloring problem as introduced in the previous section. This scheduling graph will be referred to as $P$:

Naturally, all the constraints of the separated scheduling remain. In addition—as one qubit cannot be used as control and target at the same time—the green edges in Fig. 6.7 need to be inserted. In the upper part of the highlighted “interleaving” section, a X-stabilizer and Z-stabilizer act on the same pair of data qubits. However, not every coloring (scheduling) is applicable in these circumstances. The local order of the CNOTs in the interleavings is relevant for the feasibility of the circuit. To see this, we first recapture the stabilizer-implementations by CNOTs for the separated case:

Figure 6.7: Interleaved scheduling of X and Z stabilizer measurement.
Remember, that as discussed in Chapter 3, the boxed operators perform the effective measurement of the system with both $I_xZ_z$ and $X_xI_z$ applied to the ancilla qubits in $|+\rangle_z, |0\rangle_z$, simply yield an eigenvalue of +1 and the remaining parts of the operator $Z_aZ_b$ or $X_aX_b$ make up the stabilizer measurement.

Abbreviating the set of CNOTs in this subcircuit as $Q = C_{b,z}C_{a,z}C_{x,b}C_{x,a}$, we hold on to the commutation transformation of the applied measurement as:

\[
\begin{align*}
X_x & \xrightarrow{Q} X_xX_aX_bI_z, \\
Z_z & \xrightarrow{Q} I_xZ_aZ_bZ_z.
\end{align*}
\]

It is easy to see, that these relations still hold if in Fig. 6.8 the CNOTs in boxes A or B are permuted, also when the whole Boxes A and B are interchanged. When the first and third or the second and forth CNOT are interchanged, this leads to a different effective measurement:

\[
\begin{align*}
|+\rangle_x & \quad \xrightarrow{Q} \quad \begin{array}{c}
\xrightarrow{X_x}\text{C}_{b,z}\text{C}_{x,b}\text{C}_{x,a}\text{C}_{a,z} \quad |+\rangle_x |0\rangle_z |\psi\rangle_{ab} \\
\xrightarrow{Z_z}\text{C}_{b,z}\text{C}_{x,a}\text{C}_{x,b}\text{C}_{a,z} \quad |+\rangle_x |0\rangle_z |\psi\rangle_{ab}
\end{array} \\
= & \quad \begin{array}{c}
\text{C}_{b,z}\text{C}_{x,a}\text{C}_{x,b}\text{C}_{a,z} \quad X_xX_aX_b |+\rangle_x |0\rangle_z |\psi\rangle_{ab} \\
\text{C}_{b,z}\text{C}_{x,a}\text{C}_{x,b}\text{C}_{a,z} \quad Z_xZ_aZ_bZ_z |+\rangle_x |0\rangle_z |\psi\rangle_{ab}
\end{array}
\end{align*}
\]

Figure 6.8: Interleaved Parity-X measurement with sequential X- and Z-stabilizer application with $Q = AB$.

Figure 6.9: Interleaved parity X and Z measurement with $(1,3)$-permuted CNOTs. This circuit is given by $Q'$.
Figure 6.10: Interleaved parity X and Z measurement with (2,4)-permuted CNOTs, given by $Q'$. 

One can see, that compared to the sequential case, any permutation of the CNOTs in which a control and a target operation get interchanged on a single qubit, the circuit performs an effective measurement of:

$$X_x (C_{x,b} C_{a,z} C_{b,z} C_{x,a} |+⟩_x |0⟩_z |ψ⟩_{ab}) = C_{x,b} C_{a,z} C_{b,z} C_{x,a} X_x X_a X_b X_z |+⟩_x |0⟩_z |ψ⟩_{ab}$$

$$Z_z (C_{x,b} C_{a,z} C_{b,z} C_{x,a} |+⟩_x |0⟩_z |ψ⟩_{ab}) = C_{x,b} C_{a,z} C_{b,z} C_{x,a} Z_x Z_a Z_b Z_z |+⟩_x |0⟩_z |ψ⟩_{ab}$$

Thus, the X-parity measurement additionally includes the respective Z-ancilla qubit. As the Z-ancilla qubit was initialized in $|0⟩$, the X-measurement of the Z-ancilla qubit will not simply yield a $+1$ eigenvalue, but as

$$P(|+⟩|0⟩) = |⟨+|0⟩|^2 = |⟨-|0⟩|^2 = P(|-⟩|0⟩) = \frac{1}{2},$$

a random eigenvalue $±1$ will be returned, just as the initial Z-ancilla $|0⟩$ is randomly projected into $|±⟩$. The Z-parity measurement has the identical problem as

$$P(|0⟩|+⟩) = |⟨0|+⟩|^2 = |⟨1|+⟩|^2 = P(|1⟩|+⟩) = \frac{1}{2}.$$

Thus, this order in the interleaving will not only return a random measurement outcome of the stabilizers, but also project the corresponding ancilla qubits into a random state in the wrong basis. To further investigate this problem, we can capture the faultless local CNOT orderings, denoted as $Q'$, via the single or the double-headed arrows in following picture:

| $| +⟩_x$ | $X_x (C_{x,b} C_{a,z} C_{b,z} C_{x,a} | +⟩_x | 0⟩_z | ψ⟩_{ab})$ |
|---|---|
| $| 0⟩_z$ | $Z_z (C_{x,b} C_{a,z} C_{b,z} C_{x,a} | +⟩_x | 0⟩_z | ψ⟩_{ab})$ |
Figure 6.11: Interleaving of X- and Z-stabilizer. The two green edges in the circuit (left) can be formulated in the scheduling graph as the two green edges of the corner C (right). The directions are given by the coloring of the corresponding vertices. The arrows of the green edges of the two unlabeled corners are not drawn in, but also need to be considered when coloring the interleaved scheduling graph \( P \).

The highlighted corner \( C \) is a collection of vertices whose colors imply the transformation (\( Q \) or \( Q' \)) the corresponding interleaving of the CNOTs induces. We will call a corner either parallel or antiparallel, depending on the local order of the CNOTs induced by the coloring. Let us illustrate this on following corner \( C = (i, j, k, l) \) with the nodes \((i, j, k, l)\) inheriting colors \((\alpha, \beta, \gamma, \delta)\):

\[
\begin{align*}
C \text{ parallel: } & (\alpha < \gamma) \land (\beta < \delta) \lor (\alpha > \gamma) \land (\beta > \delta) \\
C \text{ antiparallel: } & (\alpha < \gamma) \land (\beta > \delta) \lor (\alpha > \gamma) \land (\beta < \delta)
\end{align*}
\]

Figure 6.12: Corner \( C \).

With the directions of the arrows given by the inequalities between the colors, a parallel corner—
—represents the interleaving as described in Figure 6.8 or the simple interchanging of the boxes $A$ and $B$. Both parallel orderings give rise to the correct operator transformation of $Q: X_x \rightarrow X_x X_a X_b I_z$ and $Z_z \rightarrow I_x Z_a Z_b Z_z$, given by the separated applications of the X- and Z-stabilizers (in the interleaving).

If the coloring is such, that the corner is antiparallel—

—then this corner will imply a local order of $Q'$ in the subcircuit of the interleaving, which again translates into the undesired X-measurement of the corresponding Z-ancilla and the Z-measurement of the X-ancilla.

In order to design a well-functioning circuit without the undesired effective measurements—we will refer to this as a proper schedule in the future—, each projective measurement applied to any corresponding ancilla qubit (e.g. $X$ applied to an X-ancilla) has to commute through the circuit such, that no effective measurement of any ancilla qubit is performed in the wrong basis. This is given if and only if the following corner criterion is satisfied.

**Corner criterion:** A proper schedule is one, in which every face (Z-stabilizer) and every vertex (X-stabilizer) is incident to no antiparallel corner.

The condition of “only if” is given due to the fact, that each X- and Z-stabilizer overlap on either one or no corner. Assume, we fix a X- and a Z-stabilizer that overlap on an antiparallel corner. Then, in order to cancel off each effective measurement operator which acts in the wrong basis arising from each projective measurement of the X- or Z-ancilla, it would be necessary for the same two stabilizers to overlap on a second antiparallel corner. As this is not possible due to the way the code is constructed (this would also lead to the effective stabilizer-measurement operators to terminate), antiparallel corners cannot be allowed for a scheduling to be proper.

This constraint is obviously already satisfied by the sequential execution of the X- and Z-stabilizers. Thus, the existence of the coloring satisfying this constraint is out of question, as well as an upper bound of $\chi$ is already given by $\chi \leq r + s + 2$ as discussed in previous Section 6.1. Therefore, the remaining question whether or not a interleaved schedule with less timesteps can be found is purely of algorithmic nature.

The corner criterion can easily be checked once a coloring has been done. Thus, per brute-force coloring, as discussed in Appendix A, we can always generate random colorings and check if the corner criterion is satisfied.

To put this into a form feasible with ILP, we can express the inequalities between colors of specific nodes $i, k \in N$ of with colors $C(i) =: \alpha, C(k) =: \gamma$ with the boolean variable $x_{ic}$
into a boolean expression

\[ x_{ic} = 1 \quad \text{if color } c \text{ is used on } i \]

\[ (\alpha < \gamma) = \sum_{a=1}^{N} \sum_{g=a+1}^{N} x_{kg}x_{ia}. \quad (6.1) \]

The expression on the right in Eq. (6.1) is 1 if the left statement is true and 0 otherwise. With this, the oddness of a corner \( C = (i, j, k, l) \) as labeled in Fig. 6.12, can be expressed by

\[ \text{Odd}(C) = 1 + \sum_{a=1}^{N} \sum_{g=a+1}^{N} x_{kg}x_{ia} - \sum_{d=1}^{N} \sum_{b=d+1}^{N} x_{jb}x_{ld} \mod 2. \quad (6.2) \]

With this, the corner criterion becomes following expression:

\[ \sum_{C: C \cap S \neq \emptyset} \text{Odd}(C) = 0 \quad \forall S \in F \cup V. \quad (6.3) \]

\( F \) denotes the set of all faces, \( V \) the set of all vertices, such that \( S \) is the set of all stabilizers.

One can notice, that this is in fact a quadratic constraint. However, these can also be handled by ILP software tools as Gurobi.

---

Note that the corner criterion holds for general homological codes constructed in arbitrary dimensionalities as Z- and X-stabilizer always overlap a single corner (Lemma 2.9 in [8]) —e.g. on a 4D hyperbolic code one can choose the 2-cells as qubits, the 1-cells as X- and the 3-cells as Z-stabilizers.
6.3 Coloring the Small Stellated Dodecahedron

With these central ideas in place, we can return to scheduling the parity checks for the small stellated dodecahedron code. The scheduling graph can be setup as follows:

![Interleaved small stellated dodecahedron](image)

Figure 6.13: Interleaved small stellated dodecahedron.

The formulated constraints on interleaved scheduling have been implemented in both algorithmic scheduling approaches, brute-force-coloring (Appendix A) and ILP. Unfortunately, a proper interleaved schedule incorporating $T < 10$ could not be found. The optimal proper schedule therefore remains the sequential performance of the separated schedules for the X- and Z-stabilizers. Note, that the X- and Z-stabilizer labeled 11 in Fig. 5.9 have been omitted from this schedule, since the linear dependence between all stabilizers (Eq. (3.1)) brands the information gained from this measurement as redundant.

![Proper coloring of the scheduling Graph P](image)

Figure 6.14: A proper coloring of the scheduling Graph $P$ obtained by the sequential order of the separated X- and Z-stabilizer schedules.
Figure 6.15: $T = 10$ Complete schedule.
7 Fault Tolerant Quantum Computation

In this chapter we will “zoom out” of the explicit construction and features of quantum-error-correcting-codes and briefly review the basic syntax of fault tolerant quantum computation introduced by Panos Aliferis in Chapter 2 of [3]. This will lead us to some ground rules which the error correction procedure has to obey, allowing us to discuss the feasibility of fault tolerant quantum storage on the small stellated dodecahedron code.\footnote{In the syntax introduced by Aliferis \textit{et. al}, the code is also called 1-block as it corresponds to a first level of encoding.} Although the main focus of this chapter lies in the construction of the error correction procedure for a fault tolerant quantum storage, the basic idea of how logical gate operations can be wrapped into a fault tolerant implementation will also be introduced in the following section. With these ideas and the later defined error correction procedure it can be seen, how also fault tolerant quantum computation could principally be performed on the small stellated dodecahedron code.

7.1 Fault Tolerance

The notion of fault tolerance states, that if we use a quantum error correcting code with distance $d$ that corrects arbitrary Pauli errors on up to $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ qubits of this code, we expect a noisy simulation to be robust against faults that act nontrivially on no more than $t$ locations. In such simulation (we speak of the encoded circuit simulating the logical one) we will perform continuous error-correction on the qubit storage between logical gates to minimize the effect of any error.

In the rigorous framework, logical gates are represented in terms of gadgets. A logical gate on the encoded qubits (i.e. application of $Z_L$ on one of the encoded qubits or CNOTs between encoded qubits) is introduced by a gadget, abbreviated by $Ga$. Error-correction (measurement of all the stabilizers grouped with the subsequent correction) is introduced by error-correction gadgets denoted as $EC$. The original operations on the unencoded qubits will be denoted as $0$-Gadgets or $0$-$Ga$.

![Figure 7.1: Error correction- and logical operation gadget.](image)

To obtain the fault tolerant implementation of $0$-$Gas$, these will be replaced by Rectangles, consisting of the encoded gate acting on the code followed by an instance of error-correction.
Figure 7.2: Replacement rule generating the fault tolerant implementation of a two qubit gate, the Rec.

To formulate the general idea of fault tolerant circuit design, the notion of correctness of Recs is needed. In simple terms, correctness introduces the idea of what it means for the Rec to simulate the corresponding ideal (faultless) 0-Ga without producing “too many” errors. In the context of quantum error correcting codes, the phrase of “too many” refers to “a not correctable amount” of errors. To put this in to clearer terms, we first need the notion of an ideal decoder (noted as just decoder). The ideal decoder performs ideal error correction on its input and outputs the decoded logical qubit(s). With this, the correctness of a Rec is given as follows:

**Definition 10. (Correctness)**[3] A Rec is correct if the Rec followed by an ideal decoder is equivalent to the ideal decoder followed by the ideal 0-Ga.

Schematically this means:

\[
\text{correct Rec} \xrightarrow{\text{decoder}} \text{ideal 0-Ga}
\]

While these notions can also be extended to correct measurement Recs and correct state-preparation Recs, we will first limit our considerations to ordinary gadgets. For plain quantum storage purposes, the gadget would simply be the identity operator. However, since error-correction is to be performed continuously in the circuit, we are able to find more structure in the implementation of gadgets.

Figure 7.3: Two exRecs, indicated by dashed lines, overlap on an EC.

The Recs grouped with the preceding ECs are defined as extended Rectangles or exRecs. The reason why this grouping is interesting is that these building blocks of the circuit are directly related to what it means for a computation to be implemented fault tolerantly. This is expressed as:
“A computation can be performed fault tolerantly, if exRecs can be implemented such that, assuming there are not too many errors on the input as well as not too many errors inside the exRec, there also won’t be too many errors on the output.”

Obviously, this definition is just as too vague as unhandy when one tries to make concrete statements about fault tolerant implementations of any computation. In order to formulate this more rigorously, the phrase of “not too many errors happening in an exRec” needs some clarification.

**Definition 11. (Goodness)[3]**

1. Consider a fault-tolerant quantum circuit simulation based on a distance $d$ quantum error correcting code and let $t = \lfloor \frac{d-1}{2} \rfloor$. A exRec is good if it contains at most $t$ faults.

2. A Ga or EC which contains no more than $r \in \mathbb{N}$ faults is called $r$-good.

With this, above statement about fault tolerance becomes

**Definition 12. (Fault Tolerance)** A computation can be performed fault tolerantly, if every Rec contained by a good exRec is correct.

In fact, Aliferis [3] has proven, that this can be achieved, given a few necessary properties on the ECs and Gs. The properties will be the guidelines to determine if a specific code can be implemented fault tolerantly and how the ECs need to be constructed.

However, in order to understand why and how these properties are relevant, the notion of an $s$-filter is needed. $s$ is an integer between zero and the total number of data qubits. The $s$-filter performs a faultless orthogonal projection onto the subspace spanned by syndromes corresponding to Pauli errors acting on at most $s$ qubits in the code. Thus, if the syndrome of a hypothetical stabilizer measurement indicates a Pauli-correction operator with weight greater than $s$, the code is rejected by this $s$-filter and otherwise left unchanged.

**Theorem 9. (Goodness Implies Correctness)[3]** Consider a fault-tolerant quantum circuit simulation based on a distance-$d$ quantum error-correcting code that is executed using gadgets satisfying following properties:

- **EC Properties** Let $t = \lfloor \frac{d-1}{2} \rfloor$
  
  (a) $\forall r \leq t$, the output of an $r$-good EC passes through an $r$-filter, or:

\[
\begin{align*}
\text{r-good} & \quad \text{EC} \quad = \quad \text{r-good} \quad \text{EC} \quad -\quad \text{r-filter}
\end{align*}
\]
(b) \( \forall s, r \) such that \( s + r \leq t \), if a code which passes through an \( s \)-filter is input to an \( r \)-good EC followed by an ideal decoder, the EC acts as identity.

\[ \overset{s\text{-filter}}{-} \overset{\text{r-good EC}}{-} \overset{\text{decoder}}{-} = \overset{s\text{-filter}}{-} \overset{\text{decoder}}{-} \]

- **Ga Properties** Consider \( k \) encoded qubits which respectively pass through \( \{s_1, s_2, ..., s_r\} \)-filters and let them be input to an \( r \)-good gadget \( \text{Ga} \) for a distance \( d \) quantum code simulating a \( k \)-qubit gate, where \( s := \sum_{i=1}^{k} s_i + r < t \). Then

(c) all output encoded qubits pass through \( s \)-filters, or

\[ \overset{\{s_i\} - \text{filters}}{-} \overset{\text{r-good Ga}}{-} = \overset{\{s_i\} - \text{filters}}{-} \overset{\text{r-good Ga}}{-} \overset{s\text{-filters}}{-} \]

(d) the \( r \)-good \( \text{Ga} \) following a \( \{s_i\} - \text{filter} \) is correct.

\[ \overset{\{s_i\} - \text{filters}}{-} \overset{\text{r-good Ga}}{-} \overset{\text{decoder}}{-} = \overset{\{s_i\} - \text{filters}}{-} \overset{\text{decoder}}{-} \overset{\text{ideal} 0\text{-Ga}}{-} \]

*Then, the Rec contained in a good \text{exRec} is correct.* Schematically, this theorem means

\[ \overset{\text{EC}}{-} \overset{\text{Ga}}{-} \overset{\text{EC}}{-} \overset{\text{decoder}}{-} = \overset{\text{EC}}{-} \overset{\text{decoder}}{-} \overset{\text{ideal} 0\text{-Ga}}{-} \]

**Proof.** By definition, a \text{exRec} is good if it contains no more than \( t = \lfloor \frac{d-1}{2} \rfloor \) faults. Suppose there are \( s \) faults in the leading EC, \( r \) faults in the Ga and \( s' \) faults in the trailing EC with \( s + r + s' \leq t \). Then, we can write the left hand side of upper equation as

\[ \overset{\text{EC}}{-} \overset{\text{Ga}}{-} \overset{\text{EC}}{-} \overset{\text{decoder}}{-} = \overset{\text{s-good EC}}{-} \overset{\text{r-good Ga}}{-} \overset{\text{s'-good EC}}{-} \overset{\text{decoder}}{-} \mid \text{use (a)} \]

\[ = \overset{\text{s-good EC}}{-} \overset{s\text{-filter}}{-} \overset{\text{r-good Ga}}{-} \overset{(s+r)\text{-filter}}{-} \overset{\text{s'-good EC}}{-} \overset{\text{decoder}}{-} \mid \text{use (b),(c)} \]

\[ = \overset{\text{s-good EC}}{-} \overset{s\text{-filter}}{-} \overset{\text{r-good Ga}}{-} \overset{\text{decoder}}{-} \overset{\text{ideal} 0\text{-Ga}}{-} \mid \text{use (d),(a)} \]

The EC properties (a) and (b) are those, which are relevant for our considerations. The construction and fault tolerant implementation of Ga will not be discussed in this frame. Let \( t = 1 \). The rules for constructing the EC are as follows.
1. (a): Let $r = 0$. Then a faultless EC should take any state of the code as input and output a state in the codespace.

2. (a): Let $r \leq 1$. Then a EC with at most one fault should also output a state corresponding to a syndrome indicating an error with weight less than 1.

3. (b): Let $s = 0$ and $r \leq 1$. A EC with at most one fault that has an input that is in the codespace produces the same result after ideal decoding as if there were no faults inside the EC.

4. (b): Let $s \leq 1$ and $r = 0$. Then a faultless EC with an input state with at most one error outputs a state, that is corrected into the same logical state as without the faultless EC.

While 1. and 2. state the basic ideas of error correction — the correction procedure with sufficiently less errors in its input shall output a state, that does not invoke any error-syndromes on a following, perfect syndrome measurement and errors within the EC don’t lead to uncorrectable errors —, it is 3. and 4. that propose the key property guiding the construction of EC gadgets. Both properties formulate the constraint, that the error correction gadget shall not produce any error, that — by ideal error correction — propagates to form a logical error.

With these guidelines, we shall proceed to examine the code of the small stellated dodecahedron in its feasibility with fault tolerant quantum computation.
7.2 Fault Tolerance of the Small Stellated Dodecahedron Code

The small stellated dodecahedron code is a distance $d = 3$ code with weight-$5$ stabilizers. Thus, with Theorem 2 the EC shall be able to tolerate up to weight $t = \left\lfloor \frac{d-1}{2} \right\rfloor = 1$ errors. To determine how a fault tolerant implementation of this code can be achieved without introducing too many extra qubits, it is necessary to check if and how the properties of Sec. 7.1 can be satisfied assuming only weight-1 errors on the data- or ancilla qubits. Trivially, single errors on the data qubits will be reported by the two incident stabilizers.

![Image](image.png)

Figure 7.4: Two $X$-stabilizer (left) and two $Z$-stabilizer (right) report an error syndrome. The edges highlighted in green are to be corrected by application of $Z$ and $X$.

This already handles all weight-1 errors that can directly happen on the data qubits. Note, that a $Y$-error on some data-qubit simply corresponds to a combined single $Z$- and $X$- error which will be detected and can be corrected independently. As this is a weight-$3$ code (with the vertices of the dodecahedron representing the weight $3$ $X_L$ and the triangles in the disentangled —left in Figure 7.4— graph representing the weight $3$ $Z_L$), this correction scheme will not miscorrect any weight $1$ error on the data qubits resulting in a logical error. The critical part however, are errors on the ancilla qubits.
Figure 7.5: A weight 5 X-stabilizer with an X-error on the ancilla qubit between the 3\textsuperscript{rd} and 4\textsuperscript{th} timestep.

Figure 7.5 shows the error propagation of a weight 1 X-error on the ancilla qubit of an X-stabilizer (using the CNOT commutation relations discussed in Chapter 2, Eq. (2.2)) onto the data qubits. Analogously, a Z-error propagates from the the Z-ancilla onto the the data qubits taken as control of the CNOTs.

Let’s assume, that the illustrated circuit corresponds to the a stabilizer-application of the small stellated dodecahedron code with qubits \(a\) and \(c\) lying on the same distance-3 \(X\) \(L\) operator. Then, by matching the error with minimum weight, the third one would be flipped and we obtain a logical error.

Figure 7.6: A weight-2 error on the light green highlighted qubits. The exclamation marks signal the report of an error syndrome. By pure weight-1 error matching on the data qubits the dark green qubit will be miscorrected, such that the resulting state incorporates a logical error.

However, as has already been found in Section 5.2 and can be revisited from Figure 7.5, this will actually not happen for the small stellated dodecahedron code. Any Z-stabilizer has
maximally weight-1 overlap with any distance-3 $Z$-logical operator, just as any $X$-stabilizer has maximally weight-1 overlap with any distance-3 $X$-logical operator. Thus, a single $X$- or $Z$- error happening on some ancilla qubit will never propagate onto more than one qubit lying on a distance-3 $X$- or $Z$- logical operator. Furthermore, an error on the ancilla qubit that happens at the beginning of a stabilizer round will propagate onto all data qubits of this stabilizer. The generated error chain is simply equivalent to an extra application of the stabilizer and thus leads to an effectively trivial error. An error, that happens after the first CNOT of the stabilizer circuit or just before the final one, will generate a syndrome that —by the just illustrated syndrome matching for errors on data qubits— will be corrected correctly or ‘miscorrected’ into a completion of the stabilizer. Thus, these errors can also be handled trivially.

Figure 7.7: An error, that happens at the beginning of the stabilizer-circuit produces a trivial error. An error after the first step generates a syndrome, that is miscorrected into a completion of the stabilizer and as such terminated. Note, that the depicted time steps are not necessarily the ones of the complete stabilizer application but correspond to the local order of this stabilizer indicey by the complete scheduling. Also note, that the central ancilla qubit is not the one that actually labels the one measuring the shown $Z$-stabilizer, but lies on its opposite side considering Fig. 5.10.

While this already seems to be a promising feature of the code, this does not yet guarantee thatmiscorrection of error-chains induced by weight-1 errors will never lead to logical
failures as illustrated above. To see this, consider the overlap of two Z-stabilizers on two Z-logical operators as depicted in Fig. 7.8.

![Figure 7.8: Two Z-stabilizers (green and blue) overlap on two Z-logical operators (the triangles).](image)

If now the scheduling of the parity checks is such, that both pairs of qubits 1, 4 and 2, 3 are the last qubits in the circuit for the green and blue stabilizer, then both error chains would result from equally probable weight-1 error on the ancilla qubits happening during the application of the stabilizer (compare to Fig. 7.5) and their syndromes (indicated by the red exclamation marks) would be indistinguishable. Miscorrecting the wrong error-chain between those two syndromes would then lead to a logical error on both logical operators.

![Figure 7.9: Critical parts of scheduling the small stellated dodecahedron. The blue/green edges respectively belong to the same stabilizer. If both sets of edges can suffer errors due to an error that has propagated from the respective ancilla qubit, their syndromes would be indistinguishable. As both error chains would result from a weight-1 error, a minimum weight matching decoder will induce a logical error. The corresponding nodes of the coloring problem have been drawn in.](image)

A first approach to suppress the probability of errors on ancilla qubits is by substituting them with encoded states, e.g. similar to the bit flip code — or general cat states $\frac{1}{\sqrt{n}} (|0\rangle^\otimes n +$
\[ |1\rangle^\otimes n \] as discussed in [3]. However, this would ruin one of the big advantages of this code - the low total number of physical qubits necessary (this value is also called overhead). Without further encoding the ancilla qubits, the overhead of this code is at \(30 + 2 \times 12 = 54\) physical qubits to encode the \(k = 8\) logical ones.

Another way to prevent such effective error chains, that would be miscorrected into logical failures, lies in the choice of the scheduling. If a scheduling —equivalently, the coloring of the graph presented in Section 6.2— can be obtained, such that no two stabilizers, related in the fashion as depicted in Fig. 7.9, both target the chains highlighted in blue or red in their last two/three steps, then the problem of miscorrection into logical failures can be omitted for single qubit errors.

Equivalently, this problem can be solved by finding a schedule, that offers unique syndromes for every propagated single qubit error after the locally second and third time step on all ancilla qubits.

Such a schedule could be obtained by applying a degree-of-saturation algorithm (Appendix A), with \(T = 10\), and has already been presented in Figures 6.14 and 6.15 in Section 6.2. All possible syndromes on the X- and Z-stabilizers generated by single errors on the data or ancilla qubits and the corresponding data qubits to be corrected can be captured by two lookup tables (one for X-syndromes and Z-correction and one for Z-syndromes and X-correction) which will be given in Appendix B.

With these lookup tables, we will be able to define a decoding procedure that satisfies the EC properties 3. and 4. of previous Section 7.1.
7.3 The Decoder

The EC will now be constructed as follows. We apply three rounds of the complete stabilizer-circuit $S$. Before each round, the ancilla qubits need to be reinitialized in their ground state ($|0\rangle$ or $|+\rangle$). After each round, the syndrome information of the X-and Z-ancilla qubits will be stored separately in a classical register.

Figure 7.10: An error correction gadget and the corresponding classical registers. Each ancilla qubit is represented as black circle on the time layer. Error syndromes are marked in red. Syndromes corresponding to ones saved in the list are marked by connecting them. The X-register undergoes a syndrome error on the first layer.

With this scheme, it is possible to omit single errors that happen on the ancilla qubits, which do nothing but generate a wrong measurement outcome —e.g. a Z-error on a X-ancilla qubit generates an incorrect syndrome, but does not propagate to any data qubits. These types of errors will be called syndrome errors. Conversely, data errors are those errors, that either happen directly on the data qubits or first appear on the ancilla qubit of some stabilizer and then propagate onto data qubits —e.g. a X-error on a X-ancilla qubit in the stabilizer application could be such a data error.

In the time layering, data errors can be identified as their corresponding syndromes “propagate” through the different layers and —for initial weight-1 errors— match to syndrome sets contained in the lookup tables. In Fig. 7.10 such a propagation can be seen for layers 2 and 3 in the X-register. Most probably, the corresponding error originated from some data error during the first stabilizer round or a data error occurring between the first and second round. Either way, this error will be corrected by applying the correction Z-operator on the qubit given by the lookup table. The syndrome chain in the Z-register can be correspondingly matched to a data error, that occurred just before the application of the first stabilizer. We will call a syndrome, that is noted in the lookup table an allowed syndrome. For the following construction, we will extend the lookuptable by the trivial syndrome $s = 0$, corresponding to no correction.
Concretely, the decoder shall be constructed in the following way:

The syndrome $s$ is corrected using the lookup table when

1. the majority of the 3 syndromes $s$ is an allowed syndrome.
2. the only nontrivial syndrome $s = s_3$ is an allowed syndrome in layer 3.
3. all syndromes are different and $s = s_3$ is allowed.

Per construction of the lookup tables, (1.), (2.) and (3.) automatically satisfy the EC properties (b) and (a) for $r \leq 1$ stated in Theorem 9. It could be argued, that errors happening in the last stabilizer round will remain undetected for the single EC. However, the explicit local detection is unnecessary to satisfy conditions (a) and (b) for faulty ECs, as the both the ideal decoder as well as the $s$-filter are naturally equipped with the set of allowed syndromes given by the lookup tables in App. B.

By these means, the small stellated dodecahedron code can be implemented fault tolerantly against weight-1 errors.

### 7.4 Error Models

We have seen that weight-1 errors are by construction correctable by this code. In physical implementations, these errors are given by specific error models. In this section we will see, that the symmetric depolarizing noise as well as CNOT errors are correctable by given implementation of the code.

- **Symmetric depolarizing noise**: This noise model is simply an expression of the entropic factor. The transformation of the density operator $\rho$ is as such given by

  $$
  N(\rho) = (1 - p)\rho + \frac{p}{2} I
  = (1 - \frac{3p}{4})\rho + \frac{p}{4}(X\rho X + Y\rho Y + Z\rho Z)
  = (1 - \tilde{p})\rho + \frac{\tilde{p}}{3}(X\rho X + Y\rho Y + Z\rho Z)
  $$

  in simple terms, this basically means that any error occurs with probability $\tilde{p} = \frac{3p}{4}$. If an error happens, then $X$-, $Y$-, $Z$- errors all occur with equal probability. Per construction, these types of errors are already handled by our decoder.

- **CNOT errors**: CNOT errors occur depending on the actual physical implementation of the CNOTs as two-qubit gates. In a simple error model, these can be modeled in terms of combined Pauli-action on control($C$)- and target($T$) qubit as

  $$
  E_{CNOT} = P_C P_T; \quad P_C, P_T \in \{I, X, Y, Z\}
  $$
happening just after the application of the CNOT while all possible combinations occur with equal probability \( \frac{p}{16} \) and thus, all nontrivial combinations yield a probability of \( \frac{p}{15} \). Interestingly, it is easy to see that by the constructed decoder — or more concretely, by being able to correct all weight-1 data errors and sorting out single syndrome errors, these type of errors are already handled. In order to do so, we need to decompose all possible CNOT errors into syndrome errors and data errors. Trivially, errors in form of \( \text{I}_C \text{P}_T \) are simple data errors, their correctability has already been shown. Errors of type \( \text{P}_C \text{I}_T \) can be data- or syndrome-errors or even both, but due to the separation of the X-syndrome register and the Z-syndrome register, these can also be clearly identified and corrected. More interesting are CNOT errors, which act nontrivially on both, the control- and the target qubit. To illustrate that these errors are handled, consider a \( \text{X}_C \text{Z}_T \) or one \( \text{Y}_C \text{Y}_T \) error happening in the first stabilizer round of the EC on some CNOT of a Z-stabilizer. All other nontrivial combinations can be decomposed analogously.

![Figure 7.11: A \( \text{X}_T \text{Z}_C \) error on a CNOT of the first stabilizer round decomposes into a syndrome error \( \text{X}_T \) and a data error \( \text{Z}_C \). The upper qubit in the circuit is the ancilla qubit initialized in \( |0\rangle \).](image1)

![Figure 7.12: A \( \text{Y}_T \text{Y}_C \) error on a CNOT of the first stabilizer round decomposes into two data errors \( \text{Z}_T \) and \( \text{X}_C \). As both data errors are captured by a separate X- and Z-register, they can be corrected independently.](image2)

Thus, it can be seen that, given the capability of sorting out syndrome errors and identifying data errors that resulted from \( t = 1 \) errors in the circuit and handling the X- and Z-register independently, a single CNOT error is equivalent to a weight-1 data error on the separated registers.
8 Conclusion

Hyperbolic surface codes provide good properties in terms of their encoding rate and general overhead for active error correction. We have seen how they can be described and how logical qubits arise from their topology. A general approach and necessary ground rules to solve the scheduling problem of the parity-checks for general hyperbolic surface codes has been established, together with a non-scaling upper bound for the maximal time of the total stabilizer circuit. These concepts have been used to examine the small stellated dodecahedron code. By explicitly finding a feasible parity-check schedule and designing the decoding procedure, the small stellated dodecahedron code was found to be suitable for fault tolerant quantum computation.
9 Appendix

A Coloring Algorithms

The Greedy Algorithm [24] The greedy algorithm takes a permutation of the nodes $P(V)$ and the edge-set $E$ as argument and starts off with an empty color-container $K = \emptyset$: The total number of colors is finally given by $|K|$. Note, that the obtained coloring and $|K|

Algorithm 1 Greedy coloring

for $i = 1 \rightarrow |V|$ in $P(V)$ do
    if $C(i) = \emptyset$ then
        for $j = 1 \rightarrow |K|$ in $K$ do
            if $i$ not adjacent to any node $k$ with $C(k) = j$ then
                $C(i) = j$
            end if
        end for
    end if
    if still $C(i) = \emptyset$ then
        $C(i) = |K| + 1$, $K_{\text{new}} = K \cup \{|K| + 1\}$
    end if
end for

is highly dependent on the permutation of the nodes $P(V)$. Just as a perfect coloring with $|K| = \chi$ can be obtained by a suitable permutation, this permutation also directly implies an order of the coloring. Any directed coloring of an acyclic digraph can be obtained via the greedy algorithm if a permutation $P(V)$ can be found, which admits the order of the digraph.

With this in mind, the brute-force searches for a suitable coloring of the graph in Section 6.2 could be conducted via

Algorithm 2 Brute-force coloring

while $C : V \rightarrow K$ does not meet the constraints do
    $P(V) = \text{SHUFFLE}(V)$
    Greedy color$\leftarrow P(V)$
end while

Lastly, the degree of saturation algorithm (Dsatur) is a variation of the greedy algorithm, that instead of coloring the vertex-set by the order of a fixed input-permutation, dynamically determines the next node to be colored by taking the one with a highest saturation degree
sat(n). That is, the node with a maximal number of adjacent nodes already assigned a color.

Algorithm 3 Dsatur coloring

\[
\begin{align*}
X &= V \\
\text{while } X \neq \emptyset \text{ do} & \\
& \quad \text{Choose } i \in X \text{ by } \max(\text{sat}(i)) : i \in X \\
& \quad \text{for } j = 1 \to |K| \text{ in } K \text{ do} & \\
& \quad & \quad \text{if } i \text{ not adjacent to any node } k \text{ with } C(k) = j \text{ then} \\
& \quad & \quad \quad C(i) = j \\
& \quad & \quad \end{align*}
\]

\[
\text{if still } C(i) = \emptyset \text{ then} & \\
& \quad C(i) = |K| + 1, K_{\text{new}} = K \cup \{|K| + 1\} \\
& \quad X \leftarrow X - \{i\} \\
\text{end while}
\]

**Integer Linear Programming**  Integer linear programming is a class of NP-hard problems of following form:

\[
\min_{x \in \mathbb{Z}^n} \{c^T x \mid Ax \leq b, x \geq 0\}.
\]

In this work such problems are solved with the mathematical optimization software *Gurobi 7.0.2*. The problem of coloring a graph \(G = (V, E)\) can be formulated in this form:

Define new variables \(y_k \in \{0, 1\} \quad k \in \{1,...,|V|\}\). \(y_k = 1\) if color \(k\) is used, else \(y_k = 0\),

\[
\begin{align*}
& \quad x_{ik} \in \{0, 1\} \quad i \in \{1,...,|V|\}, k \in \{1,...,|V|\}: \quad x_{ik} = 1 \text{ if color } k \text{ is used on node } i.
\end{align*}
\]

Thus the problem of minimizing the number of colors

\[
\min \sum_{k=1}^{|V|} y_k
\]

has to follow following constraints:

1. \(\sum_{k=1}^{|V|} x_{ik} = 1 \ \forall i,\)
2. \(x_{ik} + x_{jk} \leq 0 \text{ if } (i, j) \in E,\)
3. \(x_{ik} - y_k \leq 0.\)

Constraint 1. makes sure that every node gets exactly one color, constraint 2. expresses that no two adjacent nodes have the same color. Constraint 3. ensures that as soon as color \(k\) gets assigned to node \(i\), \(y_k\) is also set to one.
## B Lookup Tables for the Small Stellated Dodecahedron Code

The following lookup tables provide a matching between syndromes generated by weight-1 errors on data- or ancilla qubits and the data qubits on which correction has performed. The labeling of the syndromes/stabilizers is as introduced in Section 5.2. The corresponding schedule/coloring is the one presented in Figures 6.15 and 6.14.

### Table 9.1: Lookup tables for X-syndromes and Z-corrections (left) and Z-syndromes and X-corrections (right).

<table>
<thead>
<tr>
<th>X Syndromes</th>
<th>Qubits</th>
<th>Z Syndromes</th>
<th>Qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td>6, 9</td>
<td>(6, 9)</td>
<td>4</td>
<td>(0, 10), (6, 9)</td>
</tr>
<tr>
<td>1, 4</td>
<td>(1, 4)</td>
<td>4</td>
<td>(7, 10), (7, 9), (5, 7)</td>
</tr>
<tr>
<td>4, 8</td>
<td>(4, 8)</td>
<td>2, 8</td>
<td>(6, 9), (1, 9), (7, 9)</td>
</tr>
<tr>
<td>2, 9</td>
<td>(2, 9)</td>
<td>2, 5</td>
<td>(1, 10), (1, 7), (2, 11)</td>
</tr>
<tr>
<td>0, 7</td>
<td>(0, 7)</td>
<td>3, 3, 4, 9</td>
<td>(4, 10), (6, 10)</td>
</tr>
<tr>
<td>0, 10</td>
<td>(0, 10)</td>
<td>5</td>
<td>(1, 7)</td>
</tr>
<tr>
<td>2, 3</td>
<td>(2, 3)</td>
<td>6</td>
<td>(3, 11)</td>
</tr>
<tr>
<td>1, 11</td>
<td>(1, 11)</td>
<td>6, 10</td>
<td>(5, 11)</td>
</tr>
<tr>
<td>4, 7, 10</td>
<td>(7, 10), (4, 11)</td>
<td>0, 4</td>
<td>(3, 7)</td>
</tr>
<tr>
<td>8, 10</td>
<td>(8, 10)</td>
<td>7</td>
<td>(1, 7)</td>
</tr>
<tr>
<td>9, 2</td>
<td>(4, 8), (6, 8)</td>
<td>1, 5, 9</td>
<td>(4, 8), (2, 8), (6, 8)</td>
</tr>
<tr>
<td>7</td>
<td>(4, 10), (7, 10), (4, 11)</td>
<td>4, 10</td>
<td>(6, 8), (6, 6), (6, 9)</td>
</tr>
<tr>
<td>3</td>
<td>(3, 5)</td>
<td>4</td>
<td>(5, 11), (3, 9)</td>
</tr>
<tr>
<td>1, 2, 3, 5</td>
<td>(1, 5), (4, 4), (2, 4)</td>
<td>4, 5</td>
<td>(0, 7)</td>
</tr>
<tr>
<td>4, 10</td>
<td>(4, 10)</td>
<td>8</td>
<td>(7, 9)</td>
</tr>
<tr>
<td>2, 6</td>
<td>(2, 6)</td>
<td>2</td>
<td>(0, 10)</td>
</tr>
<tr>
<td>3, 11</td>
<td>(5, 11)</td>
<td>1</td>
<td>(2, 8)</td>
</tr>
<tr>
<td>0, 9, 10</td>
<td>(0, 10), (4, 9)</td>
<td>0, 1, 9, 10</td>
<td>(2, 5), (2, 11)</td>
</tr>
<tr>
<td>1, 4</td>
<td>(1, 4)</td>
<td>0, 3, 5, 6</td>
<td>(2, 4), (1, 8), (4, 11)</td>
</tr>
<tr>
<td>3, 7</td>
<td>(3, 11), (2, 6), (6, 9)</td>
<td>0, 4, 7</td>
<td>(3, 9), (0, 9), (4, 11)</td>
</tr>
<tr>
<td>9, 8</td>
<td>(6, 8)</td>
<td>1</td>
<td>(5, 11), (5, 11)</td>
</tr>
<tr>
<td>0, 4, 7, 6</td>
<td>(6, 6), (3, 7), (6, 7)</td>
<td>7, 10</td>
<td>(5, 11)</td>
</tr>
<tr>
<td>2, 4</td>
<td>(2, 4)</td>
<td>2, 6, 9</td>
<td>(4, 10), (1, 10), (5, 10)</td>
</tr>
<tr>
<td>5, 9</td>
<td>(5, 9)</td>
<td>3, 6</td>
<td>(4, 10), (4, 11)</td>
</tr>
<tr>
<td>6</td>
<td>(4, 9)</td>
<td>1</td>
<td>(0, 10)</td>
</tr>
<tr>
<td>1, 4, 6, 10</td>
<td>(4, 10), (0, 6)</td>
<td>0, 9</td>
<td>(1, 11)</td>
</tr>
<tr>
<td>3, 7</td>
<td>(1, 7)</td>
<td>4, 5, 8</td>
<td>(7, 9), (3, 7)</td>
</tr>
<tr>
<td>9, 9</td>
<td>(0, 9)</td>
<td>0</td>
<td>(2, 4)</td>
</tr>
<tr>
<td>2</td>
<td>(2, 4)</td>
<td>2, 9</td>
<td>(6, 9)</td>
</tr>
<tr>
<td>9, 11</td>
<td>(5, 9)</td>
<td>7</td>
<td>(1, 10)</td>
</tr>
<tr>
<td>1, 11</td>
<td>(8, 11)</td>
<td>2, 9</td>
<td>(2, 6), (6, 9)</td>
</tr>
<tr>
<td>4, 6, 8</td>
<td>(4, 11), (6, 8)</td>
<td>1</td>
<td>(1, 10)</td>
</tr>
<tr>
<td>0, 4, 8</td>
<td>(0, 4), (6, 7)</td>
<td>1, 2</td>
<td>(5, 11), (2, 4)</td>
</tr>
<tr>
<td>3</td>
<td>(3, 5), (2, 4)</td>
<td>2, 9</td>
<td>(2, 6), (6, 9)</td>
</tr>
<tr>
<td>7, 10</td>
<td>(7, 10)</td>
<td>2, 9</td>
<td>(6, 9)</td>
</tr>
<tr>
<td>1, 10</td>
<td>(4, 10)</td>
<td>6</td>
<td>(1, 10)</td>
</tr>
<tr>
<td>0, 5, 8, 9</td>
<td>(2, 8), (0, 9), (2, 8)</td>
<td>5</td>
<td>(1, 4)</td>
</tr>
<tr>
<td>2, 10</td>
<td>(2, 8), (8, 10)</td>
<td>0, 1, 4, 9</td>
<td>(2, 9), (2, 5), (2, 11)</td>
</tr>
<tr>
<td>4, 5, 6, 8</td>
<td>(5, 11), (4, 11), (6, 8)</td>
<td>4, 5</td>
<td>(0, 9), (0, 10), (0, 9)</td>
</tr>
<tr>
<td>10</td>
<td>(1, 11), (2, 8), (8, 10)</td>
<td>4, 6</td>
<td>(5, 11), (5, 11)</td>
</tr>
<tr>
<td>0, 1, 3, 9</td>
<td>(1, 10), (0, 10), (3, 9)</td>
<td>8</td>
<td>(3, 4)</td>
</tr>
<tr>
<td>9, 7</td>
<td>(1, 9)</td>
<td>4, 5</td>
<td>(0, 6)</td>
</tr>
<tr>
<td>0, 4, 11</td>
<td>(4, 11)</td>
<td>7</td>
<td>(5, 7)</td>
</tr>
<tr>
<td>4, 6</td>
<td>(0, 10), (4, 10), (0, 6)</td>
<td>2</td>
<td>(4, 8)</td>
</tr>
<tr>
<td>1, 7, 9</td>
<td>(1, 11), (7, 9)</td>
<td>7</td>
<td>(2, 4), (6, 8)</td>
</tr>
<tr>
<td>0, 2, 3, 9</td>
<td>(6, 9), (2, 5)</td>
<td>6</td>
<td>(7, 10)</td>
</tr>
<tr>
<td>4, 1, 10</td>
<td>(5, 6), (6, 9)</td>
<td>6</td>
<td>(7, 10)</td>
</tr>
</tbody>
</table>

63
C The Small Stellated Dodecahedron is a Cayley-Graph

Definition 13. A regular edge coloring of a directed graph \( G = G(V, E, \sigma) \) is a map \( E \mapsto C \subset \mathbb{N} \), such that for every vertex \( v \in V \) and every color \( c \in C \) there is exactly one edge \( e \in E \) with origin- or terminal vertex \( v \) such that \( C(e) = c \).

Definition 14. [26] Let \( G = (V, E) \) be a graph and let \( \Omega \) a fixpoint-less set of generators of \( G \) satisfying

\[
    x \in \Omega \Rightarrow x^{-1} \in \Omega \text{ and } \emptyset \notin \Omega.
\]

The graph \( \Gamma = \Gamma(G, \Omega) \) is a Cayley-graph if

\[
    \forall e \in E : e = (v, gv) \quad v \in V, g \in \Omega.
\]

Theorem 10. [27] A connected graph \( G \) is isomorphic to a Cayley graph of some group \( G \) with generating set \( \Omega \) iff \( G(V, E, \sigma) \) has regular edge coloring \( C : E \mapsto \Omega \) which satisfies:

- If there is a closed directed path \((v_0, v_1, ..., v_n = v_0)\) with colors \( s_i = c(v_{i-1}, v_i) \forall 1 \leq i \leq n \), then every directed path with the same colors is also closed.

This Theorem is rigorously proven in [27].

As already discussed in Sec. 5.2, every face of the small stellated dodecahedron can uniquely be identified with one of its vertices (e.g. the one 'above' the face or the joint neighbor of the boundary set of the face). Thus, it is sufficient to show that the graph describing the set of vertices- which we will now call \( D(V, E) \)- of the small stellated dodecahedron is a Cayley graph to conclude the same for the set of its faces. It has been shown in Fig. 5.10, that \( D(V, E) \) is in fact the icosahedral graph. The icosahedral graph however, is a Cayley graph due to Theorem 10 as shown below.

\[
    \langle \Omega \rangle = \langle r, g, b \mid r^2 = g^3 = b^5 = 1 \rangle
\]

Figure 9.1: Regular edge-coloring of the icosahedral graph.
Bibliography


EIDESSTAATLICHE VERSICHERUNG

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Bachelorarbeit selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.