Abstract

We look at conserved quantities in the dense $O(n)$ loop model with periodic boundary conditions. Using computational techniques, high-order conserved quantities are calculated exactly by taking logarithmic derivatives of the transfer matrix. With these results a recursive description is conjectured. These conjectures are in turn used to generate higher-order derivatives of the transfer matrix, and it is verified that the result commutes with the Hamiltonian (and is indeed conserved).

Motivated by connections to other models, the ground state of the $O(n = 1)$ loop model is calculated for several system sizes using the power method. We aim to use these results to conjecture the probability that a cluster with some properties (such as size) appears. This is successful in some but not all cases.
## Contents

0.1 Parameters ................................................. 3

1 Introduction ................................................. 4
  1.1 Locality .................................................. 10
  1.2 Equivalence to other models ......................... 11
    1.2.1 q-state Potts model .................................. 11
    1.2.2 Six-vertex model ..................................... 16
    1.2.3 Heisenberg XXZ chain .................................. 16
    1.2.4 Critical bond percolation .......................... 18

2 Conserved Quantities ........................................ 20
  2.1 Conjectured formula ................................. 21
    2.1.1 Connected words ..................................... 22
    2.1.2 Even k ............................................. 24
    2.1.3 Odd k ............................................. 25
    2.1.4 Disconnected words ............................... 25
  2.2 Constructing new $Q_k$ ................................ 26

3 Ground state statistics ..................................... 29
  3.1 Ground state vector ................................. 29
    3.1.1 Cluster probabilities .............................. 30
  3.2 Open boundary conditions ......................... 33

4 Conclusion .................................................. 38

5 Discussion .................................................. 40

6 Acknowledgements ........................................... 41

7 Appendix ...................................................... 42
  7.0.1 Alternating sign matrices .......................... 42
  7.0.2 Example calculation of probability formula 42
  7.1 Algorithms .............................................. 45
    7.1.1 Reducing a word .................................... 45
    7.1.2 Calculating all words in $T(n_1, \ldots, n_i)$ ...... 46
    7.1.3 Diagrammatic representation of $TL_a$ ........... 49
    7.1.4 Equivalent forms of words ....................... 49
    7.1.5 Generating all unique words in $TL_a$ .......... 50
0.1 Parameters

\( n \) The loop weight in the \( O(n) \) loop model.

\( b \) Parameter for anisotropy in the homogeneous \( O(n) \) loop model.

\( T(b) \) Transfer-matrix for the \( O(n) \) loop model.

\( \mathbf{TL}_a \) The Temperley-Lieb algebra with \( a \) different generators \( e_i \).

\( \hat{e}_i \) Generators of the Temperley-Lieb Algebra.

\( e_i \) Implicit sum of generators in the Temperley-Lieb algebra over all positions.

\( \tilde{Q}_k \) Conserved quantities in the \( O(n) \) loop model generated by taking logarithmic derivatives of the Transfer matrix at the point \( b = 0 \).

\( Q_k \) Symmetrised or anti-symmetrised versions of \( \tilde{Q}_k \).

\( k \) Used in the section on conserved quantities as the order of a quantity \( Q_k \). In the section on the ground state of the \( O(n = 1) \) loop model, it stands for cluster size.

\( L \) System size (size of the finite dimension, or circumference).

\( w \) A word in the TL-algebra formed by taking a product of operators.

\( l \) Length of a word defined as the number of generators \( e_i \) that are needed to write a reduced word.

\( d \) Distance from a cluster to the winding cluster measured in lines that one needs to cross to get to the winding cluster.

\( P(w) \) The number of generators \( e_{i+1} \) that are to the left of \( e_i \) in a reduced version of a word \( w \).
1 Introduction

In this thesis we focus on the $O(n)$ loop model on the square lattice. This section starts by introducing the model and some key concepts within the model. The study of the $O(n)$ loop model is mainly motivated by equivalences to other (physical) models. In fact, the $O(n)$ loop model owes its name to its equivalence to spin models with an $O(n)$ symmetry, see e.g. [12, 6] and the references therein. We will not describe the precise equivalence, but some examples will be shown in section [1.2].

Consider the square lattice. On each face of the lattice one chooses with some probability a configuration from [ ] or [ ], such that non-intersecting loops are formed. The red lines are only allowed to have an open ending at the boundary of the system. In this thesis, only the dense $O(n)$ loop model will be considered where only the configurations [ ] and [ ] are allowed with probability $b$ and $(1 - b)$ respectively where $0 \leq b \leq 1$. We will choose $b$ to be constant in our system, resulting in the homogeneous $O(n)$ loop mode.

The partition sum in this model is given by:

$$Z(b) = \sum_c b^\# [ ] (1 - b)^\# [ ] n^\# [ ]$$

(1.1)

where $\# [ ]$ stands for the number of times [ ] appears in a configuration, $\# [ ]$ for the number of [ ], and $\# [ ]$ stands for the total number of closed loops in a configuration. The sum is over all possible configurations of the lattice. An example configuration is shown in Figure 1a. This thesis will focus on periodic boundary conditions in the horizontal direction and semi-infinite in the vertical direction such that our system becomes a semi-infinite cylinder. Let $L$ be the width (circumference) of the cylinder. Only on the very top of the cylinder can a curve be open, all other curves are closed. For even system sizes, the open curves on the boundary are pairwise connected, forming a link pattern (see Figure 1b). For odd system sizes, only one curve starting on the boundary is not connected to any other point on the boundary, thus this curve continues all the way down the cylinder. We can depict the link pattern formed on top of the cylinder by placing it on a disk where only the connectivity between the points on top of the cylinder is depicted (see Figures 1b and 1d).

The link patterns act as a basis for a vector space. Thus a general state in such a space would be a linear combination of link patterns. Let $LP_L$ be the space spanned by all link patterns patterns on a cylinder of width $L$. The dimension of $LP_L$ is given by the number of different link patterns for a system size of $L$. For even system sizes, this number is given by the Catalan number $|LP_{L=2n}| = C_n = \frac{2n!}{n!(n+1)!}$. For odd system sizes, there is one open line moving from the boundary to infinity (see Figure 1b). For each even link pattern, we can add such a line on $L$ different positions, thus $|LP_{L=2n+1}| = L \times C_n$. Linear operators act on this vector space by mapping a linear combination of link patterns to again a linear combination of link patterns. Conceptually, one can think of the action of a linear operator on a link pattern as the action of adding a linear combination of new layers on top of the cylinder.

Different configurations have different probabilities dependent on $b$. As such the different link patterns also have an associated probability.

Let us write the partition sum as a product of single-layer partition sums:

$$Z(b) = \text{Tr}\{T(b)^N\}$$

(1.2)

where $N$ now denotes the number of layers in the cylinder. In this formulation, $T$ can be thought of as a one layer partition sum: It sums all possible configurations that can appear in one layer and gives each configuration the correct weight dependent on the parameters $b$ and $n$. Alternatively, one can view $T(b)$ as a linear operator mapping a link pattern to every link pattern that can be formed by addition of a single layer with corresponding weight dependent on $b$. Thus in the vector space $LP_L$, the operator $T(b)$ is a matrix. There are $2^L$ unique configurations for one layer (as there are two possible configurations for each position). The matrix $T(b)$ is called the Transfer matrix [3]. For our model:

**Definition 1. Transfer Matrix:**

$$T(b) = \prod_{i=1}^{L} \left( (1 - b) [ ] + b [ ] \right)$$

(1.3)

---

1 Also called the Completely Packed (CP) $O(n)$ loop model.
2 It will be shown in the Appendix that two transfer matrices $T(b), T(b')$ commute regardless of the values of $b,b'$, in which case our results do not depend on the specific value of $b$.
3 Since we are dealing with an infinite system, there are an infinite number of configurations and the partition sum is also infinite.

---
1 INTRODUCTION

(a) Even system size. Generated with $b = 0.3$.

(b) The link pattern on the boundary (top) of the configuration for $L = 10$ to the left.

(c) Odd system size. Generated with $b = 0.5$.

(d) The link pattern on the boundary (top) of the configuration for $L = 11$ to the left.

Figure 1: Random configurations of the dense $O(n)$ loop model on the square lattice for even and odd sizes. The link patterns on the boundary (top) of the cylinder are depicted on the right.

where $i$ denotes the position of the plaquettes. Note that is normalized via Newton’s binomial:

$$
\sum_{m=0}^{L} \binom{L}{m} (1 - b)^m b^{L-m} = 1 \quad (1.4)
$$

As a linear operator the transfer matrix maps a link pattern to every other possible link pattern that can appear by addition of a single layer with correspondent weight. We can apply the same trick again and write the transfer matrix as a product of smaller versions acting on single plaquettes:

$$
T(b) = \text{Tr}\{ R_{0,0}(b) R(b) \ldots R_{L,0}(b) \} \quad (1.5)
$$

where the index 0 denotes the layer (in this case we just named the top layer 0) and the indices $R_{i,j}$ denotes the position of the $R$-matrix in the lattice. In this definition the $R$-matrix is:

$$
R = (1 - b) \begin{array}{c|c} 1 & 2 \\ \hline 2 & 1 \end{array} + b \begin{array}{c|c} 1 & 2 \\ \hline 3 & 4 \end{array} \quad (1.6)
$$

Now we can parameterize $b$ in two parameters $z$ and $w$:

$$
b = \frac{qz - q^{-1}w}{qw - q^{-1}z} \quad (1.7)
$$

where choosing $z = we^{i\theta}$ and $q = e^{2\pi i 3}$ ensures proper normalization. We can make a visualization of this by picturing the parameter $w$ as a horizontal line and $z$ as a vertical line. At any crossing of the two parameters, there is an $R$-matrix dependent on the two parameters via equation (1.6). This is shown in Figure 2a. The transfer matrix in this visualization is shown in Figure 2b.

The $R$-matrices have several important properties. Firstly, the inverse of an $R$-matrix has the property:

$$
R(z,w)R(w,z) = C(z,w)I \quad (1.8)
$$

Where $C(z,w)$ is some constant. Pictorially this is shown in Figure 3a (omitting $C$). Furthermore, the $R$-matrices satisfy the Yang-Baxter equation (YBE) [5]:

$$
R_{12}(\lambda, \mu)R_{13}(\lambda, \nu)R_{23}(\mu, \nu) = R_{23}(\mu, \nu)R_{13}(\lambda, \nu)R_{12}(\lambda, \mu) \quad (1.9)
$$
1 INTRODUCTION

Figure 2: Visual representations of the $R$-matrix and the Transfer matrix. At any crossing of two parameters, there is an $R$-matrix dependent on the parameters via equation 1.6.

Figure 3: Unitarity condition (inverse) and the Yang-Baxter equation for the $R$-matrices.

Figure 4: Pictorial derivation of $[T(b), T(b')] = 0$ for $\lambda, \mu$, and $\nu$ arbitrary parameters. The YBE is depicted pictorially in Figure 3b. With these two equations, we can show that two transfer matrices $T(b)$ and $T(b')$ commute for arbitrary $b$. Take two transfer matrices, depicted by the top left figure in 4 with different parameters $w$. Hence, they each have a different resulting $b$. We can insert two $R$-matrices at any point by explicitly writing an identity and using equation 1.8. Now we can apply the YBE repeatedly to change the position of the new $R$-matrices. We can do this until the two additional $R$-matrices are next to each other again. Finally, we can again use relation 1.8 to get rid of the two $R$-matrices. Note that we omitted the constant in the first and the last step, as the two constants exactly cancel out one another. Thus we see that the two transfer matrices $T(b)$ and $T(b')$ indeed commute.

Figure 4: Pictorial derivation of $[T(b), T(b')] = 0$

$[T(b), T(b')] = 0 \quad \forall b, b' \in [0,1]$ (1.10)

Commuting transfer matrices play an important role in quantum integrability, discussed in e.g. [5]. In classical physics, a system is integrable if one can write down enough independent equations of motion such that one can solve differential equations describing the time evolution of the system by integrating them thus fully solving the system. Quantum integrability relies on the existence of a set of algebraically independent operators commuting with the Hamiltonian (conserved quantities) that grows proportional to the system size. For a model expressed in a transfer matrix, such a set can be generated from the transfer matrix on the condition that two transfer matrices $T(b)$ and $T(b')$ commute. One can then use
the transfer matrix to generate commuting quantities\footnote{We later define (anti-)symmetrical versions of $\tilde{Q}$ called $Q$ that are used more throughout this thesis. Therefore there is a tilde on this operator.}

$$\tilde{Q}_k := C_k \left( \frac{d}{db} \right)^k \ln T(b) \bigg|_{b=0}$$ (1.11)

With $C_k$ some arbitrary constant (set $C_k = 1$ for now), and each of these quantities commutes with one another $[\tilde{Q}_k, \tilde{Q}_m] = 0$. If the Hamiltonian is contained in this sequence, all these quantities are conserved. $R$-matrices that satisfy the YBE lead to commuting transfer matrices, and thus to a system with sufficient conserved quantities via equation 1.11. For a more complete overview of the applications, we refer to either the book by Baxter \[3\], the lecture notes by Jacobsen \[6\].

Since two transfer matrices commute for any value of $b$, we can just rescale $b$ in our model arbitrarily. Moving forward, we choose the transfer matrix:

$$T = \prod_{i=1}^{L} (b_i + b)$$ (1.12)

From this transfer matrix, we can write down the sequence of commuting quantities $\tilde{Q}_k$ (equation 1.11). Rewriting the logarithm in terms of derivatives of $T(b)$:

$$\tilde{Q}_k = C_k \left( \frac{d}{db} \right)^k \ln T(b) \bigg|_{b=0} = C_k \left( \frac{d}{db} \right)^{(k-1)} T^{-1}(b) \frac{d}{db} T(b) \bigg|_{b=0}$$ (1.13)

which is valid if $k > 0$. $T^{-1}(b)$ is defined as the inverse of the operator $T(b)$ at the point $b$. Since the action of operators corresponds to adding a layer on top of the cylinder, we want $T^{-1}(b)T(b) = 1$. Pictorially, this means we want all lines of the operator $T^{-1}(b)T(b)$ to go straight. However, not every operator always has an inverse and it could be difficult to find. Let us take a closer look at the transfer matrix. Sorting all terms in $T(b)$ by their order in $b$, we can write:

$$T(b) = b^0 O_0 + b^1 O_1 + b^2 O_2 + ...$$ (1.14)

where $O_i$ is the collection of all the terms proportional to $b^i$. Pictorially, we get:

$$T(b) = \begin{array}{c}
1 \\
+ b \begin{array}{c}
\end{array} \\
+ b^2 \begin{array}{c}
\end{array} \\
+ ... \\
+ b^L \begin{array}{c}
\end{array}
\end{array}$$ (1.15a)

Note that we can write:

$$O_i = \frac{1}{i!} \left( \frac{d}{db} \right)^i T(b) \bigg|_{b=0}$$ (1.16)

Note that at the point $b = 0$, $T(b)$ reduces to just $O_0$, which is a shift operator: it maps a link pattern to a rotated version of itself. Hence, the inverse operator is well defined at least at the point $b = 0$ that we are interested in. We get:

$$T^{-1}(b) \bigg|_0 := \begin{array}{c}
\end{array}$$ (1.17)

This is a valid definition as long as we only evaluate $T^{-1}(b)$ and the function $\tilde{Q}_k$ at the point $b = 0$. $O_1$ is given by:

$$O_1 = \sum_{i=1}^{L} \begin{array}{c}
\end{array}$$ (1.18)
Each term in this sum corresponds to a shift everywhere except at the one. This gives for the first nontrivial commuting quantity \( \tilde{Q}_1 \):

\[
\tilde{Q}_1 = T^{-1}(b) \left( \frac{d}{db} T(b) \right)_{b=0} = \sum_{i=1}^{L} \cdots \sum_{i=1}^{L} \left| \begin{array}{c}
1 \\
2 \\
\vdots \\
L-1 \\
L
\end{array} \right| = \sum_{i=1}^{L} \cdots \sum_{i=1}^{L} \left| \begin{array}{c}
1 \\
2 \\
\vdots \\
L-1 \\
L
\end{array} \right|
\]

(1.19)

Where the right hand side simply displays the resulting connections following the red lines. We will take this as minus the Hamiltonian of our system:

\[
H = -\tilde{Q}_1 = -\sum_{i=1}^{L} \cdots \sum_{i=1}^{L} \left| \begin{array}{c}
1 \\
2 \\
\vdots \\
L-1 \\
L
\end{array} \right|
\]

(1.20)

Taking this as our Hamiltonian is a natural choice as it corresponds to the Hamiltonian of several physical models, e.g. the Heisenberg XXZ spin chain. These correspondences are discussed in section 1.2.

Let:

Definition 2. A monoid is given by

\[
\left| \begin{array}{c}
1 \\
2 \\
\vdots \\
L-1 \\
L
\end{array} \right|
\]

Monoids are a representation of the affine Temperley-Lieb algebra (\( TL_a \)):

Definition 3. The affine Temperley-Lieb algebra \( TL_a \) is defined by a generators \( \tilde{e}_1, \tilde{e}_a \) and the shift operator \( u \) along with multiplication and summation with the following rules:

\[
\tilde{e}_i \tilde{e}_i = n \tilde{e}_i \tag{1.21a}
\]

\[
\tilde{e}_i \cdot \tilde{e}_{i \pm 1} \cdot \tilde{e}_i = \tilde{e}_i \tag{1.21b}
\]

\[
[\tilde{e}_i, \tilde{e}_j] = 0 \quad \forall|i-j| > 1 \tag{1.21c}
\]

\[
u \cdot \tilde{e}_i = \tilde{e}_{i+1} \cdot u \tag{1.21d}
\]

\[
\tilde{e}_L+1 = \tilde{e}_1 \tag{1.21e}
\]

where \( n \) is the loop weight in the \( O(n) \) model. The accent on top is added because the quantity \( e_i \) is reserved for the sum over all positions of \( \tilde{e}_i \), as this sum will be used a lot more in this thesis. The transfer matrix in this representation becomes:

\[
T = \prod_{i=1}^{L} (1 + b \tilde{e}_i)
\]

(1.22)

We will make some definitions for this algebra below. Let (following [8, 9]):

Definition 4. A word is a sequence of generators \( \tilde{e}_i \). Two words are equivalent if equations (1.21) can be used to make them equal.

where the length of a word is equal to the number of generators that it consists of. It is possible to write a word in many different forms using the TL-rules (equations 1.21). For instance, the word \( \tilde{e}_1 = (\tilde{e}_1 \tilde{e}_2)^a \tilde{e}_1 \) \( \forall \alpha \in \mathbb{Z} \). Define a reduced word:

Definition 5. A reduced word is a word that cannot be written as a product of fewer generators using the TL-rules (1.21).

A reduced word may still have many equivalent reduced forms via equation 1.21c e.g. \( \tilde{e}_1 \tilde{e}_3 = \tilde{e}_3 \tilde{e}_1 \).

Finally:

Definition 6. Let \( w \) be a word, \( i_l \) be the index of the generator with the lowest index and \( i_h \) the highest index. A word is connected iff all \( \tilde{e}_i \) for \( i \in [i_l, i_h] \) appear in a reduced form of \( w \) exactly once. A word that is not connected is disconnected.

Since every way to reduce a nonreduced word involves the action of two copies of one single generator \( \tilde{e}_i \), every connected word is also in a reduced form.
Figure 5: Monoids as a graphical representation of $\text{TL}_a$. Straight lines are omitted

In Figures 5a-5c it is shown that monoids indeed satisfy the rules of the TL-algebra. A product of generators $\tilde{e}_i \tilde{e}_j$ in the pictorial representation means putting a monoid on position $i$ above one on position $j$, while keeping all other lines straight.

We see that in the first rule indeed a loop is closed resulting in a factor $n$. Thus, the first commuting quantity is given by a sum of monoids on all positions. As with the Hamiltonian, other conserved quantities are usually a combination of some words summed over all possible positions on the cylinder. Therefore, it is useful to take the sum over all positions implicitly.

$$e_{i_1}e_{i_2}... := \sum_{j=1}^L \tilde{e}_{i_1+j} \tilde{e}_{i_2+j}...$$

(1.23)

The relative distance between the generators $\tilde{e}_i$ is constant as the whole word moves position. Note that this is a single sum over the position, not a separate sum for each generator. To explicitly make the distinction between the sum of one word $\tilde{e}_1 \tilde{e}_2$ over all positions of the cylinder and the product of two words each summed over all positions separately ($e_1$ and $e_2$), we will denote the product of two words explicitly by using a dot ($w_1 \cdot w_2$). Any time this is omitted the reader can assume a single word summed over all positions is denoted. For example, the multiplication $\cdot$ between two words $e_1$ and $e_2$ is given by:

$$e_1 \cdot e_2 = \left( \sum_{i=1}^L e_{1+i} \right) \left( \sum_{j=1}^L e_{2+j} \right)$$

(1.24a)

$$= \tilde{e}_1 \tilde{e}_2 + \tilde{e}_2 \tilde{e}_3 + ... + \tilde{e}_1 \tilde{e}_3 + \tilde{e}_2 \tilde{e}_4 + ...$$

(1.24b)

$$= e_1 e_2 + e_1 e_3 + ... + e_1 e_L + e_1 e_1$$

(1.24c)

$$= \sum_{\alpha=1}^L e_1 e_{2+\alpha}$$

(1.24d)

The Temperley-Lieb rules are just as valid for $e_i$ as they are for $\tilde{e}_i$: nothing in the order of the generators is changed by taking the implicit sum over all positions. Thus we can still use the rules to reduce a word $e_ie_j...$ in the same way. With monoid, we now mean an object with an index $e_i$ that satisfies the TL-rules, so we will both use it for $\tilde{e}_i$ and $e_i$.

As noted above, $\tilde{Q}_k$ commutes with all other $\tilde{Q}_j$. Since the Hamiltonian is given by $H = -\tilde{Q}_1$, all $\tilde{Q}_k$ also commute with the Hamiltonian. Hence the commuting operators $\tilde{Q}_k$ is are conserved quantities with:
1.1 Locality

Theorem 1.1. In $\hat{Q}_k$, there are only terms resulting from the action of an uninterrupted sequence of $e_i$. Define these terms to be locally constructed as opposed to nonlocally constructed words.

Let $i_l$ be the lowest index of all generators in a word and $i_h$ the highest. Then uninterrupted means that every $\hat{e}_i$ with $i \in [i_l, i_h]$ must also be in the word at least once. It is important to note that the resulting words just have to be constructed in this way. To make this more clear, we could calculate the quantities $\hat{Q}_k$ without ever using the TL-rules to simplify and add up the resulting words. In that calculation, only words consisting of an uninterrupted sequence of generators can contribute. However, after applying the TL-rules (e.g. to reduce the word $e_1e_3e_2e_1$ to $e_1e_3$) the resulting words can be interrupted. The difference between locally constructed and nonlocally constructed words is depicted in Figure 6.

![Figure 6: The difference between the local and nonlocal construction of a word. The operator on the left contains all $\hat{e}_i$ in the range 1–3 at least once, whereas the operator on the right misses $\hat{e}_2$. The resulting operator $\hat{e}_1\hat{e}_3$ is the same.](image)

**Proof:** Suppose $\hat{Q}_k$ contains a word $v$ constructed as $v = q_1q_2$, where $q_1$ only contains $e_i$ where $i \in [m + 1, n - 1]$ and $q_2$ only $e_j$ where $j \in [n + 1, m - 1]$ (periodic). Then this word also occurs in the result of taking the same derivative of a modified transfer matrix where the $m$-th and $n$-th terms in the product in equation 1.22 are replaced by just 1. Let:
1.2 Equivalence to other models

As mentioned above, one of the main reasons for studying $O(n)$ loop models is that they are equivalent to several physical models. The precise connection between spin models with $O(n)$ symmetry will not be discussed in detail here. However, the equivalence to the $q$-Potts model, the six-vertex model, the Heisenberg $XXZ$-chain, and bond percolation on the square lattice will be discussed in this chapter.

1.2.1 q-state Potts model

The $q$-state Potts model is defined by considering particles with a spin $\sigma_i$ that can take $q$ different values on a lattice. Only nearest neighbor interactions are taken into account. We consider the standard Potts model on a square lattice, where two neighboring particles contribute a negative energy $-\epsilon$ if and only if the two particles have the same spin value [16]. The couplings are allowed to be different in the vertical direction ($\epsilon_v$) and horizontal direction ($\epsilon_h$). The Hamiltonian is given by:

$$H_{\text{Potts}} = -\epsilon_h \sum_{\langle i,j \rangle} \delta_{i,j} - \epsilon_v \sum_{(k,l)} \delta_{k,l}$$

(1.27)

where $\langle i, j \rangle$ denotes nearest neighbors in the horizontal direction and $(k,l)$ in the vertical direction and $\delta_{i,j} := \delta(\sigma_i, \sigma_j)$. Thus for the partition sum, we get:

$$Z_{\text{Potts}} = \sum_c \exp(-\beta H_{\text{Potts}}) = \sum_c \exp \left( \beta \epsilon_h \sum_{\langle i,j \rangle} \delta_{i,j} + \beta \epsilon_v \sum_{(k,l)} \delta_{k,l} \right)$$

(1.28)

where $c$ runs over all $q^N$ different configurations. Following the argument of [2], we will now show that this is equivalent to the $O(n)$ loop model on the square lattice. Rewriting the partition sum as a product of two-particle interactions:

$$Z_{\text{Potts}} = \sum_c \prod_{\langle i,j \rangle} e^{\beta \epsilon_h \delta(\sigma_i, \sigma_j)} \prod_{(i,j)} e^{\beta \epsilon_v \delta(\sigma_i, \sigma_j)}$$

(1.29)

Let:

$$v_h \delta_{i,j} + 1 := (e^{\beta \epsilon_h} - 1) \delta_{i,j} + 1 = e^{\beta \epsilon_h \delta_{i,j}}$$

$$v_v \delta_{k,l} + 1 := (e^{\beta \epsilon_v} - 1) \delta_{k,l} + 1 = e^{\beta \epsilon_v \delta_{k,l}}$$

(1.30a)

(1.30b)

Then:
1.2 Equivalence to other models

The partition sum now consists of a product of two terms for each nearest neighbor interaction in the system. Let $E$ be the total number of two-particle interactions. Then the total partition sum has $2^E$ terms. A one-to-one correspondence can be made with each of these terms and a graph. Following [2], let $G$ be a graph with $N$ vertices on the same positions as the spin particles (in our case the square lattice) and $E$ edges between all neighboring vertices. A term in $Z$ is represented by placing a bond on an edge $(i,j)$ if the term $v_\delta_{i,j}$ was chosen and leaving the edge empty if 1 was chosen. Using this correspondence, each graph with $l_v$ vertical bonds and $l_h$ horizontal bonds stands for a term proportional to $v_v^{l_v} v_h^{l_h}$. A connected component in a graph is defined as a subset of vertices such that all of them are connected via paths of bonds, but none is connected to a vertex outside the set (i.e. a connected cluster of vertices). Then a connected component represents a set of particles of the same spin value. But that cluster can still have $q$ different values. Thus a graph with $C$ connected components represents $q^C$ different spin configurations each having the same weight. We can summarize this in a partition function where we sum over all possible graphs:

$$Z_{\text{Potts}} = \sum_{\varepsilon} \prod_{(i,j)} (v_r \delta_{i,j} + 1) \prod_{(k,l)} (v_v \delta_{k,l} + 1)$$
1.2 Equivalence to other models

As an example, let us consider four neighboring particles on the square lattice for \( q = 2 \):

\[
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array}
\]

Figure 7: Four particle 2-state Potts model

We first write down the partition function of the Potts model:

\[
Z_{\text{Potts}} = \sum_c \prod_{(i,j)} (v_h \delta_{i,j} + 1) \prod_{(k,l)} (v_h \delta_{k,l} + 1)
\]

\[
= \sum_c (v_h^2 v^2_h \delta_{1,2} \delta_{1,4} \delta_{2,3} \delta_{3,4} + v_h v^2_h \delta_{1,2} \delta_{1,4} \delta_{2,3} + \delta_{1,4} \delta_{2,3} \delta_{3,4} + v_h^2 \delta_{1,4} \delta_{2,3})
\]

\[
+ v_h^2 v_h^2 (\delta_{1,2} \delta_{1,4} \delta_{2,3} \delta_{3,4} + v_h v_h \delta_{1,2} \delta_{1,4} + \delta_{1,2} \delta_{2,3} + \delta_{2,3} \delta_{3,4} + \delta_{3,4} \delta_{2,3})
\]

\[
+ v_h (\delta_{1,4} + \delta_{2,3}) + v_h^2 \delta_{1,2} \delta_{3,4} + v_h (\delta_{1,2} + \delta_{3,4} + 1)
\]

For four particles, we can explicitly perform the sum over all configurations. For \( q = 2 \), we sum once over all unique configurations determined by which spins are of equal value and multiply the result by \( q = 2 \) to account for the two possible values. We get:

\[
Z_{\text{Potts}} = 2 \times \left( v_h^2 v^2_h + 2 v_h v^2_h + 2 v^2_h v_h + 4 v_h v_h + v^2_h + 2 v_h + v + 1 \right)
\]

\[
= 1 = 2 = 3 = 4
\]

\[
+ v_h v_h + v_h + v + 1
\]

\[
= 1 = 2 = 3 \neq 4
\]

\[
+ v_h v_v + v_h + v_v + 1
\]

\[
= 1 = 2 \neq 3
\]

\[
+ v_h v_v + v_h + v_v + 1
\]

\[
= 1 = 3 \neq 4
\]

\[
+ v_h v_v + v_h + v_v + 1
\]

\[
= 2 = 3 = 4
\]

\[
+ v^2_h + 2 v_h + 1
\]

\[
= 1 = 2 \neq 3
\]

\[
+ 1
\]

\[
= 1 = 3 \neq 2
\]

\[
+ v^2_h + 2 v_v + 1
\]

\[
= 1 = 4 \neq 2
\]

\[
= 2 v_h^2 v^2_h + 4 v_h v^2_h + 4 v^2_h v_v + 16 v_h v_v + 4 v^2_h + 4 v^2_v + 16 v_v + 16
\]

Now compare this to the partition function summing over all graphs. We get:

\[
Z_{\text{Graphs}} = \sum G q^{C} v^i_h v^j_h
\]

\[
= \bullet \\
\bullet \\
\bullet \\
\bullet \\
\end{array}
\]
1.2 Equivalence to other models

Giving:

\[
Z_{\text{Graphs}} = 2^1 v_h^2 v_v^2 + 2^1 v_h^2 v_v + 2^1 v_h v_v^2 + 2^1 v_h^2 v_v + 2^1 v_h v_v^2 \\
+ 2^2 v_h v_v + 2^2 v_h v_v + 2^2 v_h v_v + 2^2 v_h v_v + 2^2 v_h^2 \\
+ 2^2 v_h^2 + 2^3 v_v + 2^3 v_h + 2^4 v_v + 2^4
\]

\[
= 2v_h^2 v_v^2 + 4v_h^2 v_v + 4v_h v_v^2 + 16v_h v_v + 4v_h^2 + 4v_v^2 + 16v_h + 16v_v + 16
\]

The two partition sums are indeed equal. The authors in [2] can now move from graphs to loops in the following way. Let \( \mathcal{L} \) be the square lattice used before. Then, let \( \mathcal{L}' \) be a new lattice with a vertex on each edge of \( \mathcal{L} \) and some vertices around the boundaries of \( \mathcal{L} \) such that the edges connecting the new vertices form polygons around the vertices in \( \mathcal{L} \) (see Figure 8). Vertices placed on the edges in \( \mathcal{L} \) are internal vertices (blue) and the vertices around the boundaries of \( \mathcal{L} \) are external vertices (red). Different versions of \( \mathcal{L}' \) with different external vertices are allowed as long as a polygon is formed around each vertex in \( \mathcal{L} \).

Figure 8: The square lattice \( \mathcal{L} \) in black and the polygon lattice \( \mathcal{L}' \) in blue. The red vertices are internal and the blue vertices external.

Now any graph on \( \mathcal{L} \) can be associated with a polygon graph in the following way. On all edges of \( \mathcal{L} \), there is an internal vertex of \( \mathcal{L}' \). If the edge is occupied by a bond, the vertices on \( \mathcal{L}' \) are connected such that they do not intersect the edge (Figure 9a). If the edge on \( \mathcal{L} \) is not occupied by a bond, the vertices are connected such that they separate the two vertices on \( \mathcal{L} \) connected by that edge (Figure 9b).

If we rotate the lattices \( \mathcal{L} \) and \( \mathcal{L}' \), we retrieve the two configurations in the fully packed loop model (Figures 9c, 9d). Thus the partition sum over all graphs is equivalent to the partition sum of the dense \( \mathcal{O}(n) \) loop model.

In this way, a loop is formed around every connected component in a graph. Additionally, there is a loop within each closed cycle. Thus the loop configuration equivalent to a graph with \( C \) connected components and \( S \) cycles has \( p = C + S \) loops. The number of closed cycles in a graph on the square lattice is given via Euler’s rule [2]:

\[
S = C - N + l_v + l_h
\]

(1.37)

where \( N \) is the number of vertices. We can rewrite the partition function using \( C = \frac{1}{2}(p + N - l_v - l_h) \):
1.2 Equivalence to other models

Figure 9: From a graph on $\mathcal{L}$ to polygons on $\mathcal{L}'$.

Figure 10: Loops around a graph.

Figure 11: The different vertices and a random configuration of the six-vertex model.
\[ Z = \sum_G q^{l_v^h + l_h^v} = q^{N/2} \sum_G q^{v_v^h (q^{-1/2} v_h)^{l_v} (q^{-1/2} v_h)^{l_h}} = q^{N/2} \sum_G q^{l_v^h x_h^v x_h^h} \] (1.38)

where \( x_h = q^{-1/2} v_h \). This is equivalent to the partition sum for the O(n) loop model where \( n = q^{1/2} \).

### 1.2.2 Six-vertex model

The authors of [2] move on to show that the O(n) loop model is also equivalent to the six-vertex model. Consider the lattice \( L' \) from the previous section. Place arrows on each edge of \( L' \) such that there are two arrows pointing towards each vertex and away from each vertex. For every internal vertex, there are six possibilities shown in Figure 11a. An example configuration is shown in Figure 11b.

The six-vertex model is not only defined on the square lattice, but works on any planar graph where each vertex is connected to four nearest neighbors. A Boltzmann weight is assigned to each vertex dependent on the angles that the edges make at a vertex. The explicit dependence of the weights on the angles is given in [2], but we will just consider the square lattice where all angles are \( \pi/2 \). Let:

\[
q^{1/2} = 2 \cosh \theta \tag{1.39a}
\]

\[
z = e^{\theta/2\pi} \tag{1.39b}
\]

An internal vertex can be of two types \( r \) in our model (vertical \( v \) or horizontal \( h \)) depending on the edge in the lattice \( L \) it is located on. Weights in this case are given by (following the vertex numbering in Figure 11a):

\[
w_1 = w_2 = 1 \tag{1.40a}
\]

\[
w_3 = w_4 = x_r \tag{1.40b}
\]

\[
w_5 = z^\pi + x_r z^\pi \tag{1.40c}
\]

\[
w_6 = z^\pi + x_r z^{-\pi} \tag{1.40d}
\]

Furthermore, to each external vertex a weight of \( z^\phi \) is assigned where \( \phi \) is the angle that the two edges meeting in the external vertex make. The weight of a configuration is given by the product of the weights of all vertices. For the configuration in Figure 11b, we get the weight:

\[
w = (z^{\pi/2})^8 \times w_2 \times w_3 \times w_3 = (z^{\pi/2})^8 \times x_h^2 \tag{1.41}
\]

The partition sum is the sum over all configurations of the weights. It is proven in [2] that the partition sum for the six-vertex model on \( L' \) is equivalent to the partition sum of the \( q \)-state Potts model up to a constant via:

\[
Z_{q\text{-Potts}} = q^{N/2} Z_{6\text{-vertex}} \tag{1.42}
\]

Which will not be shown in detail in this thesis. But since the partition sum for the \( q \)-Potts model is equivalent to that of the O(n) loop model, the six-vertex model is also equivalent to the O(n) loop model.

Six-vertex models are also known as ice-type models, as the bonds in this model can be used to represent hydrogen bonds in the crystal structures of several materials, most notably ice [11].

### 1.2.3 Heisenberg XXZ chain

In addition to the two-dimensional models described above, remarkable connections between the Heisenberg XXZ-chain and combinatorics have been made [13, 10]. Furthermore, it turns out the dense O(n = 1) loop model is equivalent to some versions of the Heisenberg XXZ-chain dependent on the boundary conditions [10], and we can find the connections to combinatorics in our model. As an example, consider an XXZ-chain with periodic boundary conditions and odd system size:
\[ H_{\text{odd}} = -\sum_{i=0}^{L-1} \frac{1}{2} \left( \sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y - \frac{1}{2} \sigma_i^z \sigma_{i+1}^z - \frac{3}{2} \right) \]  

(1.43)

This is also a representation of the TL-algebra, where:

\[ \hat{e}_i = \frac{1}{2} \left( \sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y - \frac{1}{2} \sigma_i^z \sigma_{i+1}^z + \frac{1}{2} \right) \]  

(1.44a)

\[ H_{\text{odd}} = \sum_{i=0}^{L} (1 - \hat{e}_i) \]  

(1.44b)

Thus it is no surprise that one can map several versions the Heisenberg XXZ-chain to the dense O(n = 1) loop model. Similar mappings exist for different boundary conditions (e.g. the O(n = 1) loop model with periodic boundary conditions and even system size can be mapped to a slightly different version of the XXZ Hamiltonian). Additionally, the connections between the ground state of the XXZ-chain and combinatorics described below can also be observed in the ground state of the O(n = 1) loop model.

For an odd number of sites \( L = 2m - 1 \), the Hamiltonian has a lowest eigenvalue \(-3L/4\) corresponding to the ground state vector \([15] [13]\). Note that the operator corresponding to a spin flip on each position:

\[ R = \prod_{i=1}^{L} \sigma_i^z \]  

(1.45)

commutes with this Hamiltonian. Thus, say there is an eigenvector \( H |\psi_i\rangle = E_i |\psi_i\rangle \) with nonzero total spin in the \( z \) direction \( \sum_{j=1}^{L} \sigma_j^z |\psi_i\rangle \neq 0 \). Then the same eigenvector with every spin flipped is also an eigenvector of \( H \) with the same eigenvalue \( E_i \) but opposite \( S_z \). Hence any eigenvalue corresponding to a state with \( S_z \neq 0 \) is twofold degenerate. Furthermore, the eigenvectors are also eigenvectors of the shift operator. Thus only symmetric combinations of basis states can appear in eigenstates of \( H \):

\[ |\uparrow\downarrow\uparrow\rangle = |\uparrow\uparrow\uparrow\rangle + |\downarrow\uparrow\uparrow\rangle \]  

(1.46)

Following [13], we denote the coefficients of these components in the ground state by translating the up and down spins to bits, e.g. \( \psi_{001011} \) denotes the coefficient for \( |\uparrow\downarrow\uparrow\uparrow\rangle \). In this notation, the coefficient \( \psi_{00...011...1} \) is the smallest. Moreover, if the ground state is normalized such that \( \psi_{00...011...1} = 1 \), all other coefficients are integer value. For a system size of \( L = 2m - 1 \), the largest component is equal to the number of alternating sign matrices in \( m \times m \) dimensions \( A_m \) (see appendix 7.0.1):

**Conjecture 1.2** (Razumov Stroganov 1).

\[ \psi_{00101011...} = \prod_{j=0}^{(i-1)} \frac{(3m + 1)!}{(m+j)!} := A_m \]  

(1.47)

This is one of the so-called Razumov Stroganov conjectures first described in [13]. Its validity has been proven in [4]. An analogous expression applies to even system sizes \( L = 2m \) [14]. The authors also conjectured:

**Conjecture 1.3** (Razumov Stroganov 2). The norm of the ground state eigenvector normalized such that \( \psi_{00...011...1} = 1 \) is given by:

\[ N_m = \sqrt{\frac{3\pi}{2m} \frac{(3m-1)!}{(2m-1)!}} A_m \]  

(1.48)

where the 3 is a generalization of the factorial given by:

\[ (N)\alpha := (N)!!...! = N(N - \alpha)(N - 2\alpha)...(N \mod \alpha) \]  

(1.49)

As mentioned before, the conjectures made above are also valid for ground states of the O(n = 1) loop model. Furthermore, in [11] similar conjectures have been made for the O(n = 1) loop model with open boundary conditions:

**Conjecture 1.4** (Nienhuis, Batchelor, De Gier [11]). The largest element of the ground state wave function of the O(n = 1) loop model with open boundary conditions is given by:
1.2 Equivalence to other models

\[ \psi_{01...01(0)} = \begin{cases} N_8(L) & L = 2m \\ A_v(L) & L = 2m - 1 \end{cases} \] (1.50)

where \( N_8(2m) \) are the number of cyclically symmetrical transpose complement plane partitions and \( A_v(2m + 1) \) are the number of unique ASM’s that are symmetrical about the vertical axis given by:

\[
N_8(2m) = \prod_{i=1}^{m-1} (3i + 1) \frac{(2i)!(6i)!}{(4i)!(4i + 1)!} 
\]

\[
A_v(2m + 1) = (-3)^m \prod_{i=1}^{2m+1} \prod_{j=1}^{m} \frac{3(2j - i) + 1}{2j - i + 2m + 1} 
\]

The above observations connecting the XXZ-chain to alternating sign matrices under some symmetry conditions also apply to the ground state of the \( O(n) \) loop model that the XXZ-chain is equivalent to. For more details, we refer to ??.

1.2.4 Critical bond percolation

Besides the link between the \( O(n) \) loop model and the spin models described above, the \( O(n = 1) \) loop model is equivalent to bond percolation on the square lattice. This is defined by drawing a point on every vertex on a square lattice and connecting each point to its neighboring points with a probability \( p \). A dual lattice can then be defined by placing vertices in the center of a plaquette. Then two neighboring points in the dual lattice are connected if the line separating them in the original lattice is not connected. The equivalence is best seen by placing blue and yellow points on the vertices of the square lattice such that only yellow points neighbor blue points and vice versa. Two example plaquettes are depicted in Figure ??.

Connect two vertices if it is possible to do so without crossing any lines. Rotating the model by \( \pi/4 \), we can then identify two lattices corresponding to the conventional square lattice for bond percolation (the blue corners) and the dual lattice (the yellow corners). The probability of connecting two blue vertices throughout the cylinder always adds up to \( 1/2 \) regardless of the value for \( b \). To provide an intuitive argument for this, realize that two neighboring squares in the \( O(n) \) loop model have the color of their corners reversed. Hence, if one would need \[\square\] on one of both squares to connect the blue corners, one would need \[\square\] to connect the blue corners on the other square. The probability for any two blue corners to be connected is given by:
\[ P(\text{blue connected}) = P(\text{need } \begin{array}{c} \quad \end{array}) P(\begin{array}{c} \quad \end{array}) + P(\text{need } \begin{array}{c} \quad \end{array}) P(\begin{array}{c} \quad \end{array}) = \frac{1}{2}(1 - b) + \frac{1}{2}b = \frac{1}{2} \quad (1.52) \]

And similar for the probability for two blue corners to not be connected. While the given probability of two arbitrary neighboring blue points adds up to \(1/2\), the probability for two points to be connected given their orientation is dependent on \(b\): to connect two blue points horizontally, we always need \(\begin{array}{c} \quad \end{array}\) whereas \(\begin{array}{c} \quad \end{array}\) is always needed to connect them vertically. Thus, there is an asymmetry dependent on \(b\). Bond percolation on the square lattice is critical for connection probability \(1/2\) [7].

Finally, note that in systems of even size it is only possible to connect points of the same color to one another. To see this, take two neighboring points on the lattice and note that they are already separated by a red line since we fill every square of the lattice. If the red line separating does not end on the boundary, it must end in a loop enclosing either of the two points, but not both. If the red line does end on the boundary, it half-encloses either of the two points, thus making it so that it can never connect to the other one. For odd system sizes, there is always one cluster that winds around the cylinder and on which one can move from one lattice to the dual lattice. This is discussed in more detail below.

The different physical models presented above provide us with a motivation for studying the \(O(n)\) loop model. The conserved quantities we aim to calculate correspond exactly with conserved quantities in the Heisenberg \(XXZ\)-chain via mappings the mapping in equation 1.44a and to conserved quantities in the \(q\)-Potts model. Furthermore, we aim to find more connections between combinatorics and the ground state.
2 Conserved Quantities

Now that we introduced the model and have shown some equivalent physical models motivating our research, we move on to the main results of this thesis. They are divided in two parts: In the first part, conserved quantities in the O(n) loop model are calculated exactly up to tenth order. From the results, a recursive formula is conjectured and in turn used to create further conserved quantities. The second part explores the connection between the ground state of the O(n = 1) loop model and combinatorics.

From theorem 1.1 it would be logical to only generate words by chaining generators in a closed range such that there is at least one generator on each position in the range, as the result can only contain such terms resulting from taking the derivative of the terms 1/T(b) terms resulting from taking the derivative of the terms 1/T(b). However, this equality is only valid if we apply the TL-rules to reduce all words and add up and subtract equal reduced versions. Once the TL-rules are applied, it is no longer possible to discern whether or not it was constructed locally. Thus either we must accept that equation 2.2 is invalid, or we choose to give up relation 2.2 and explicitly keep track of the order of all derivatives of T. This also means we cannot freely change the position of the T^{-1}(b). We choose to take the derivative of one such terms such that the new d/db T(b) terms resulting from taking the derivative of the terms 1/T(b) are always trailing the already existing derivatives in the numerator.  Thus for the derivative of a term like equation 2.1 we get:

\[
\frac{d}{db} T(n_1, n_2, \ldots, n_i) = \sum_{k=1}^{i} T(n_1, n_2, \ldots, n_k + 1, \ldots, n_i) - T(n_1, n_2, \ldots, n_k, 1, n_k, \ldots, n_i) \tag{2.4}
\]

By taking the derivative of an expression like T(n_1, n_2, \ldots, n_i) every T^{-1}(b) generates a new first derivative via (d/db)T^{-1}(b) = -(T^{-1}(b))^2(d/db)T(b), so there is an additional first derivative entering the expression. Each (d/db)^n T(b) is accompanied by a T^{-1}(b) in this way. Using this procedure, the first four Q_k are given by:

\[
\begin{align*}
\tilde{Q}_1 &= T(1)
\bigg|_{b=0} \tag{2.5a} \\
\tilde{Q}_2 &= T(2) - T(1, 1)
\bigg|_{b=0} \tag{2.5b} \\
\tilde{Q}_3 &= T(3) - 2T(2, 1) - T(1, 2) + 2T(1, 1, 1) \tag{2.5c} \\
\tilde{Q}_4 &= T(4) - 3T(3, 1) - T(1, 3) + 6T(2, 1, 1) + 3T(1, 2, 1) + 3T(1, 1, 2) - 3T(2, 2) - 6T(1, 1, 1, 1) \tag{2.5d}
\end{align*}
\]

Note that we are enumerating all partitions of the number k with all their permutations in this way.

---

5We will henceforth write T(b = i) explicitly if the value of b is meant and T(n_1, \ldots) always denotes derivatives of T(b)

6Either choosing the T^{-1}(b) always trailing or always leading would lead to a correct description.
With these observations, a procedure is created to calculate the \( \tilde{Q}_k \) for \( k \) as high as possible. First, the expansion of \( \tilde{Q}_k \) in derivatives of \( T(b) \) is calculated by applying equation 2.4 to the expression for \( \tilde{Q}(k-1) \) starting at \( \tilde{Q}_1 = T(1) \). Then all words that are locally constructed in a term \( T(n_1, \ldots, n_i) \) are calculated. This is done in parallel for all different terms in the expansion, which allows for a factor of three speedup as we had access to three different cores. The procedure is described in appendix 7.1.2. The resulting words are put in a reduced form using the algorithm described in appendix 7.1.1. All equivalent forms of a reduced word are produced and saved in a hash table using the procedure described in appendix 7.1.3. Using this table, all equivalent versions are mapped to one version and all results can be added together. Using this procedure it was possible to calculate \( \tilde{Q}_k \) up to and including \( k = 10 \). For \( \tilde{Q}_{10} \) the computations could no longer be performed on a normal machine due to memory issues. Instead they were performed on a cloud computer with 100GB of RAM.

### 2.1 Conjectured formula

Using the expressions we found for conserved quantities up to tenth order, it is possible to conjecture a general formula. Let us first define a mapping \( F : w \in TL_n \mapsto w' \in TL_n \):

\[
F(e_{i_1}e_{i_2}e_{i_3} \ldots) = e_{(L+1)−i_1} e_{(L+1)−i_2} e_{(L+1)−i_3} \ldots
\]  

(2.6)

This corresponds to a reflection in a vertical axis of the cylinder.\(^7\) Define a quantity that is mapped to itself under the action of \( F \) symmetrical and a quantity that is mapped to minus itself to be anti-symmetrical.

The \( \tilde{Q}_k \) are usually not completely (anti-)symmetrical; only \( \tilde{Q}_1 \) is symmetrical. We can however add and subtract other \( \tilde{Q}_k \) freely, as the result still commutes with \( H \). In this way, we observe that is possible to make every \( \tilde{Q}_k \) even completely anti-symmetrical by subtracting only lower \( \tilde{Q}_{j<k} \) and every \( \tilde{Q}_k \) odd completely symmetrical in the same way. Furthermore, since the terms in each \( \tilde{Q}_k \) are formed by taking \( k \) derivatives of the transfer matrix and setting \( b \) to zero afterwards, every term in the \( \tilde{Q}_k \) must be constructed by taking the product of exactly \( k \) monoids \( e_i \). Consequently, the longest terms in \( \tilde{Q}_k \) are a product of \( k \) monoids on different positions. Furthermore, there are only two possible rules for the \( e_i \) with which we can reduce a product of \( k \) monoids to a product with fewer generators: \( e_i^2 = ne_i \) and \( e_i e_{i+1} e_i = e_i \). The first rule decreases the length of a product of monoids by one and adds a factor of \( n \), the second rule decreases the length by two and does not add an \( n \). Hence, if \( k \) is even (odd), this means all terms of odd (even) length \( l \) can only be an odd (even) polynomial \(^8\) in \( n \). In other words, for a word of length \( l \) in conserved quantity \( \tilde{Q}_k \) the parity of the polynomial is always equal to the parity of \( (k−l) \). The leading order term is always \( n^{k−1} \).

The longest words in \( \tilde{Q}_k \) (those made up of exactly \( k \) different monoids) cannot appear in lower \( \tilde{Q}_{j<k} \), but we observe that they always appear in (anti-)symmetrical combinations. Furthermore, we observe that if \( \tilde{Q}_k \) is divided by a factor of \((k−1)!\), the coefficients for the longest words all become 1. For simplification, we can just absorb this coefficient by defining the coefficient \( C_k \) in equation 1.11. For \( k \) even, all coefficients are then of integer value whereas for \( k \) odd some coefficients are of half-integer value. Thus we get for the constant in equation 1.11:

\[
C_k = \frac{1}{(k−1)!}
\]  

(2.7)

We will define the canonical form \( Q_k \) of the \( k \)-th conserved quantity to be the sum of \( \tilde{Q}_k \) and the minimal number of lower conserved quantities \( \tilde{Q}_{j<k} \) such that the total is either completely symmetrical or completely anti-symmetrical:

\(^7\) The factor \((L+1)\) was chosen to ensure the indices are always positive. Since the system is periodic and the words are summed over all positions, we could have chosen any constant and the result would be equally valid.

\(^8\) E.g. creating the term \( e_1 e_2 \) from a product of five generators always yields an odd polynomial. For example: \( e_1 e_1 e_2 e_3 e_2 = n^2 e_1 e_2 \) or \( e_1 e_2 e_1 e_2 e_2 = n e_1 e_2 \)
Table 1: Canonical forms $Q_k$ up to $k = 10$

<table>
<thead>
<tr>
<th>$Q_k$</th>
<th>Expansion in $Q_{j&lt;k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_1$</td>
<td>$Q_1$</td>
</tr>
<tr>
<td>$Q_2$</td>
<td>$Q_2 + nQ_1$</td>
</tr>
<tr>
<td>$Q_3$</td>
<td>$Q_3 + \frac{1}{2}nQ_2$</td>
</tr>
<tr>
<td>$Q_4$</td>
<td>$Q_4 + 2nQ_3 - n^3Q_1$</td>
</tr>
<tr>
<td>$Q_5$</td>
<td>$Q_5 + \frac{1}{2}nQ_4 - \frac{1}{2}n^3Q_2$</td>
</tr>
<tr>
<td>$Q_6$</td>
<td>$Q_6 + 3nQ_5 - 5n^3Q_4 + 3n^6Q_1$</td>
</tr>
<tr>
<td>$Q_7$</td>
<td>$Q_7 + \frac{1}{2}nQ_6 - \frac{5}{4}n^3Q_4 + \frac{1}{4}n^5Q_2$</td>
</tr>
<tr>
<td>$Q_8$</td>
<td>$Q_8 + 4nQ_7 - 14n^3Q_5 + 28n^5Q_3 - 17n^7Q_1$</td>
</tr>
<tr>
<td>$Q_9$</td>
<td>$Q_9 + \frac{1}{2}nQ_8 - \frac{5}{4}n^3Q_6 + \frac{5}{4}n^5Q_4 - \frac{15}{2}n^7Q_2$</td>
</tr>
<tr>
<td>$Q_{10}$</td>
<td>$Q_{10} + 5nQ_9 - 30n^3Q_7 + 126n^5Q_5 - 255n^7Q_3 + 155n^9Q_1$</td>
</tr>
</tbody>
</table>

Conjecture 2.1. [(anti-)symmetrical forms] Conserved quantities given by $\hat{Q}_k$ can be made completely symmetrical for odd $k$ and anti-symmetrical for even $k$ under the transformation $w \rightarrow F(w)$ by adding appropriate factors of previous quantities $\tilde{Q}_{j<k}$. We define the (anti-)symmetrical form of $Q_k$ to be $Q_k$.

The expression of $Q_k$ in terms of $\hat{Q}_k$ up to $k = 10$ can be found in Table 1. First of all, we note that in order to symmetrise the odd conserved quantities, we only need to add even terms. Conversely, to anti-symmetrise the even quantities, only odd terms are necessary. Furthermore, the coefficients are only monomials in $n$. This is remarkable: To (anti-)symmetrise the terms of length $(k-1)$ in the $k$-th quantity, we have to add a specific number of $n \times \hat{Q}_{k-1}$ (we have no freedom here, as terms of length $(k-1)$ only appear in $\tilde{Q}_k$ and $\tilde{Q}_{(k-1)}$). This fixes the symmetry for all terms up to a factor of $n$. Proceeding to higher orders in $n$, we see that all terms proportional to $n^2$ are automatically (anti-)symmetrised as well. Proceeding to terms of length $(k-3)$, we see that again only one monomial in $n^3$ times $Q_{(k-2)}$ is necessary to create a completely (anti-)symmetrical combination of these terms. Doing so automatically fixes the symmetry for all terms of length $(k-4)$ as well etc.

Looking at the sequence $Q_k = \tilde{Q}_k + C_1n^3\hat{Q}_{k-1} + C_2n^5\hat{Q}_{k-3} + \ldots$, we see that $C_1 = \frac{1}{3}\binom{k}{1}$, $C_2 = \frac{1}{4}\binom{k}{3}$, and $C_3 = \frac{1}{5}\binom{k}{5}$. However, the apparent consistency seems to stop there, as the next coefficient $C_4$ differs significantly from $\frac{1}{4}\binom{k}{3}$. A general formula for constructing $Q$ from $\hat{Q}$ was not found. But it turns out easier to find the regularities in the resulting (anti-)symmetrized form $Q_k$, from the exact computer calculations up to $Q_{10}$.

2.1.1 Connected words

We now turn to the analysis of the (anti-)symmetrised versions of the conserved quantities. Firstly, we notice that words with double generators do not appear in $Q_k$.

Conjecture 2.2 (Double generators). Reduced words that contain a certain generator twice or more (e.g. $e_2e_1e_3e_2$) always have a coefficient of 0 in $Q_k$.

This simplifies our description: While the number of times one generator can appear in a reduced word is finite, there are a lot of variations one can make. Thus not needing a description for these words is a nice simplification.

Footnote: It is capped at the integer part of $(L+1)/2$ where $L$ is the dimension of the Temperley-Lieb algebra $\text{TL}_a$. To see this, start with the word $e_2e_1e_3e_2$. If we try to add another $e_2$ the same way $e_2e_1e_3e_2e_2$, we see that we must prevent the second set of $e_1e_3$ from annihilating the middle $e_2$ via $e_1e_2e_1$. The only way to do that is by adding $e_Le_4$ in between, resulting in $e_2e_1e_3e_2e_Le_4e_1e_3e_2$. Each time we try to add another $e_2$, we need another two new generators.
Since we observe that many coefficients are the same, we begin by investigating what properties of a word determine what its coefficient in $\tilde{Q}_k$ will be. Define the p-value\(^{10}\) of a word:

**Definition 7 (p-value).** The p-value $P(w)$ of a reduced word $w$ is defined as the number of times an $e_{i+1}$ appears to the left of an $e_i$.

Note that the p-value is constant on different reduced forms of a word. Conceptually, the p-value of a word is the answer to the question: “How many times must one swap an $e_{i+1}$ with an $e_i$ starting from all generators in the word sorted from low to high”. One can only apply TL-rule 1.21c to alter a reduced word to a different reduced version, and commuting two $e_i, e_j$ with $|i - j| > 1$ does not change the p-value. We will assume for any word that the highest index of any generator is smaller than the system size. This is to ensure the definition works correctly under rotations of the system, as otherwise a term like $e_1 e_L$ contributes to the p-value if we consider the term $\tilde{e}_2 \tilde{e}_{L+1} = \tilde{e}_2 \tilde{e}_1$ in the implicit sum, but would not if we consider the term $\tilde{e}_2 \tilde{e}_L$. Let $w$ be a reduced word of length $l$ and $w' = F(w)$. Then:

$$P(w) = p \Leftrightarrow P(w') = (l - p - 1) \tag{2.8}$$

Hence, only reduced words with $p = (l - p - 1) \Rightarrow p = (l - 1)/2$ can be symmetrical, as reduced words with different p-values can never be equivalent. Since $P(w)$ must be integer, words of even length can never by symmetrical.

The conserved quantities $Q_k$ are described by assigning a coefficient to each unique reduced word that can be constructed by a product of $k$ generators $e_i$. We will start by looking at connected words. It turns out there is a way of relating the coefficients for the polynomials of connected words of length $l$ to the coefficients of connected words of length $(l + 1)$. For connected words:

**Conjecture 2.3.** Let $cw$ be a connected word of length $l$ in $Q_k$. The coefficient of that word in $Q_k$ is uniquely determined by the properties $k, l$, and $p = P(cw)$. Denote by $C_{k,l}^p$ the coefficient for all connected words of length $l$ with $P(cw) = p$ in the $k$-th conserved quantity.

Note that these properties do not uniquely define a connected word, but $C_{k,l}^p$ is equal for all different reduced words with equal $p$ and $l$.

Each word in $Q_k$ was constructed by taking a product of $k$ generators. As noted above, the coefficients are given by either odd or even polynomials in $n$ dependent on the parity of the difference $(k - l)$. This is still valid in $Q_k$. Explicitly writing the coefficients as a polynomial, we get:

$$C_{k,l}^p = \sum_{i=0,2,4}^{[k-l]} Z_{k,l}^i n^{(k-l-i)} \tag{2.9}$$

Where the $Z_{k,l}^i$ denote the coefficients of the polynomial. Now let us start at the connected terms of highest length in $Q_k$. Their coefficient is just a numerical factor, as the rule 1.21a decreases the length of the word and cannot have been applied at all. We observe:

**Conjecture 2.4.** The coefficient for connected words of length $l = k$ in $Q_k$ with $P(cw) = p$ is given by:

$$C_{k,k}^p = Z_{k,k}^0 n^0 = (-1)^p \tag{2.10}$$

For connected words of smaller length $l < k$, we observe a recursive relation where a large part of $C_{k,l}^p$ is equal to $C_{k,(l+2)}^{p+1}$. In fact, knowing $C_{k,(l+2)}^{p+1}$, we only have to add a single factor to obtain $C_{k,l}^p$:

**Conjecture 2.5 (recursive relation).**

$$C_{k,l}^p = Z_{k,l}^0 n^{(k-l)} + C_{k,(l+2)}^{p+1} \tag{2.11}$$

Or, equivalently:

$$Z_{k,l}^2 = Z_{k,(l+2)}^{p+1,0} \tag{2.12}$$

Where all $C_{k,l > k}^p = 0$ (of course, there are no terms of length $k + 1$ or higher in $Q_k$).

---

\(^{10}\)For lack of a better term, $p$ was slightly inspired by the word ‘permutation’. 
Let us look at $e_1 e_2 e_3$ in $Q_{10}$ as an example. Its total polynomial coefficient is observed to be $C_{10,3}^0 = -1n^1 - 10n^3 - 55n^5 - 155n^7$. The monomial as added in equation 2.11 in this polynomial is $Z_{10,3}^{0,0} n^{10-3} = -155n^7$. The rest of the polynomial is just equal to the polynomial for words with $l+2$ and $p+1$ (e.g. the word $e_2 e_1 e_3 e_4 e_5$; there are 4 nonequivalent connected words of length 5 with $p = 1$). Now the coefficient for this set of words is $C_{10,5}^1 = -1n^1 - 10n^3 - 55n^5$ where again the terms $-1n^1 - 10n^3$ are equal to the coefficient for connected words like $e_3 e_2 e_1 e_4 e_5 e_6 e_7$ (there are $\binom{6}{2}$ nonequivalent connected words with $l = 7$ and $p = 2$) for which the polynomial coefficient is given by $C_{10,7}^2 = -1n^1 - 10n^3$, etc. We only need to find a description for all values $Z_{k,l}^{p,0}$, the coefficient of the leading term in $n$. Then the recursive relation 2.11 gives the rest of the polynomial for arbitrary $l$. From now on, we will drop the index $i$ such that any $Z_{k,l}^p$ then denotes the coefficient leading in $n$ where $i = 0$. In summary, the problem has now been reduced to the challenge of finding a description for the leading term $Z_{k,l}^p$ in the polynomial coefficient for words of length $l$ and $p$-value $p$ in the $k$-th conserved quantity.

It is possible to identify another recursive relation relating $Z_{k,l}^p$ and $Z_{k,l+1}^p$ to $Z_{k,l+1}^{p+1}$:

**Conjecture 2.6.**  

$$Z_{k,l}^p + Z_{k,l+1}^p = -Z_{k,l+1}^{p+1}$$ (2.13)

If $Z_{k,l+1}^p$ is known for all $p$ and we know one value $Z_{k,l+1}^{p=p'}$, we can construct all coefficients $Z_{k,l}^p$. We can make a table of all $Z_{k,l}^p$'s, where we put all values with equal $l$ on one row and increase $p$ from 0 to $(l-1)$. Then we can put all values for $(l-1)$ on the row below, such that it is shifted by half a cell. This choice is made to emphasize conjecture 2.6 taking the sum of two neighboring cells always equals minus the cell above. Then three $Z$-values are depicted in Figure 13 summarizing relation 2.13. The completed tables for $k = 10$ and $k = 9$ are shown in Figures 14a and 15a.

![Figure 13](image-url)

Figure 13: If two of these coefficients are known, the third can be calculated using conjecture 2.6 where the two cells on the bottom always sum to minus the top cell.

### 2.1.2 Even $k$

Let us first turn to the case where $k$ is even. As noted before in conjecture 2.1, $Q_{k \in (2N)}$ is anti-symmetrical under the action of $F$. Now consider the special words of odd length $l = (2m - 1)$:

$$sw_l = e_m e_{(m-1)} e_{(m-2)} \ldots e_1 e_{m+1} e_{m+2} \ldots e_{(2m-1)}$$ (2.14)

E.g. $sw_5 = e_3 e_2 e_1 e_4 e_5$. These words have $P(sw_l) = \frac{(l-1)}{2}$ and have the property that $F(sw_l) = sw_l$: they are symmetrical. Since even conserved quantities can be made completely anti-symmetrical, we know these coefficients must be zero in even conserved quantities:

$$Z_{k,l}^{p=\frac{(l-1)}{2}} = 0 \quad \forall \ k \text{ even} \ l \text{ odd}$$ (2.15)

Note that not every word with $p = \frac{(l-1)}{2}$ is always symmetric. However, according to conjecture 2.3 a word’s length and $p$-value uniquely determines its coefficient in the conserved quantities, thus it is only necessary to show that one word with $p = \frac{(l-1)}{2}$ has coefficient zero to assume all others with the same values also have coefficient zero.

Thus, we now have an initial $Z_{k,l}^p$ value for all odd values of $l$. For even values of $l$, we observe the simple relation:

$$Z_{k,l}^{(l+1)} = Z_{k,l}^{(l+1)}$$
2.1 Conjectured formula

\[ Z_{k,l}^{p=0} = -Z_{k,l+1}^0 \quad \forall k, l \text{ even} \]  

(2.16)

Combined with the initial values for \( Z_{k,k}^p \) given by equation 2.10, this gives a complete description of the connected terms in conserved quantities for even \( k \).

2.1.3 Odd \( k \)

For odd \( k \), a similar argument can be made. We know \( Q_k \) is symmetrical for odd \( k \). Additionally, we know for connected words \( P(F(cw)) = (l - 1) - P(cw) \). Thus, consider the set of coefficients \( Z_{p,k,l}^{(l-2)/2} \) for words with \( p = (l - 2)/2 \) and \( l \) even. Applying \( F \) to words of this type maps them to words of where \( p = (l - 1) - (l - 2)/2 = l/2 = (l - 2)/2 + 1 \), so it raises the \( p \)-value by one. But we are considering a symmetrical quantity and we just applied a reflection, thus we also know the coefficients of these two sets of words must be equal:

\[ Z_{p,k,l}^{(l-2)/2} = Z_{p,k,l}^{(l-2)/2+1} \]  

(2.17)

Combining this with the conjecture 2.6, we get an initial \( Z_{k,l}^p \) for even \( l \): \n
\[ Z_{k,l}^{p=(l-2)/2} = Z_{k,l}^{p=(l-2)/2+1} = -Z_{k,l+1}^{l/2} \quad \forall k \text{ odd}, \ l \text{ even} \]  

(2.18)

Now for odd values of \( l \), we observe a similar relation as equation 2.16:

\[ Z_{k,l}^{p=0} = -Z_{k,l+1}^{p=0} \quad \forall k, l \text{ odd} \]  

(2.19)

Hence, equations 2.19 and 2.16 can be summarized into:

Conjecture 2.7.

\[ Z_{k,l}^{p=0} = -Z_{k,l+1}^{0} \quad \forall (k + l) \text{ even} \]  

(2.20)

These initial conditions for each \( Z_{k,l}^p \) together with the initial condition for \( Z_{k,k}^p \) (equation 2.10) and the recursive relation given by equation 2.13 completely define the coefficients \( C_{p,k,l}^p \) for connected words in \( Q_k \). We can now describe disconnected words (words with one or more \( e_i \) missing in the range 1..\( l \)).

2.1.4 Disconnected words

Now let us look at words with holes in them:

Definition 8. A hole in a word is defined as the absence of an \( e_i \) for \( i_1 < i < i_h \) in a reduced form, where \( i_1 \) and \( i_h \) denote respectively the highest and the lowest index of any generator in a word as usual.

Note that similar to the \( p \)-value, the number of holes is equal on different reduced versions of a word. We will use \( hw_1 = e_1 e_5 e_4 \) and \( hw_2 = e_1 e_3 e_5 \) as examples. Holes can only be created from an uninterrupted product of \( k \) generators by applying equation 1.21b. Both \( hw_1 \) and \( hw_2 \) have two holes. However, \( P(hw_1) = 1 \) whereas \( P(hw_2) = 0 \). The locations of the holes matter: If the holes are placed next to each other, the values \( P(w) \) can extend over a different range than if the holes are not next to each other. We observe that the coefficient of reduced words is still dependent on \( l \) and \( p \), but we will add two properties for words with holes in them: the number of holes \( h \) and the number of gaps \( g \). A gap is defined as a maximally chosen sequence of neighboring holes such that their neighboring generators are in the word. For example, if one encounters three holes where two of them are neighboring and the third one is not, the word contains two gaps.

To finalize our description, we note that the coefficient for disconnected words without double generators is directly related to the coefficient for connected words:
2.2 Constructing new $Q_k$

The description found from the observations above can now be used to generate $Q_{k>10}$. As an example we will construct the quantities $Q_{10}$ and $Q_9$. We need to find all coefficients $Z^p_{k,l}$ such that the polynomial coefficient for every unique word can be calculated using relation (2.11). A complete table of all the $Z$-values and how they are constructed is shown in Figure 14a. The shape of this table is chosen such that the central axis contains the coefficients for symmetrical words. This means if a word is described by a coefficient on the left of the central axis, the reflected version of the word is described by the coefficients on the same position to the right of the central axis. Hence, the table in this form must be symmetrical about the central axis for $k$ odd and anti-symmetrical for $k$ even.

Starting out from conjecture 2.4, we can write down $Z^p_{10,10}$ (the blue row in Figure 14a). Furthermore, we can immediately fill in all $Z^p_{10,10}/2 = 0$ for $l$ odd as they describe the coefficient for i.a. symmetric words (equation (2.15) the green entries in Figure 14a). With one entry in the row for $l = 9$ and the row above for $l = 10$ complete, we can use conjecture 2.6 repeatedly to fill in the complete row. This is true for all rows in the table: if the row above is completely known, just one entry is needed to fill in the full row. For $l = 8$, we can now use conjecture 2.7 to find the initial entry. This procedure can be repeated: for rows where $l$ is even, we make use of conjecture 2.7 and for $l$ odd, we use conjecture 2.15. Thus the whole table can be calculated. For odd $k$ the procedure is similar, but equation 2.15 cannot be used to find an initial value for $l$ even. Equation 2.18 is used instead, as is highlighted for $k = 9$ in Figure 15a.

Once the table is complete, Coefficients for specific words can be obtained using equation 2.11 and conjecture 2.8 for disconnected words. For example, say we want to know the coefficient for the word $e_2e_1e_4e_3$. Its length is 4 and its $p$-value 2, so we refer to the correct position in the table as shown in Figure 14b. Then move up by two rows, add that coefficient times the correct power of $n$ (starting with $n^0$, add 2 to the exponent for every two steps up) and so on until it is not possible to move further up. Then all that is left is to generate all unique reduced words with $k$ or fewer generators and assign them the correct coefficient. The algorithm used to generate all reduced words of some length is given in Appendix 7.1.5.

The above procedure is used to generate quantities $Q_{11}$ .. $Q_{17}$. The number 17 was chosen for the practical reason that it became too computationally straining for higher orders on an average computer.

To check if the generated $Q_k$ are conserved, the commutator with $e_1$ is computed. Note that the sum over all positions was implicit, and we did not specify a cylinder width $L$. But $e_1$ automatically commutes with all words that contain only generators in the ranges $[i + 2, i - 2]$. Therefore, we only need to check for a word $w'$ if it commutes with $e_1$ for $i$ chosen in the range $[i - 1, i_0 + 1]$, where $i_0$ and $i$ are the lowest and highest index of the generators in $w'$. Furthermore, if a word on one position of the cylinder commutes with the Hamiltonian, it commutes on all positions of the cylinder, so we only need to check if the combination of all words in $Q_k$ on one position (replacing all $e_i$ in the word by $e_i$) commutes. The generated quantities indeed commute with the Hamiltonian.

While it is computationally costly to produce all unique words in the algebra (the number of unique reduced words in $T_{L_0}$ is given by the Catalan number $\frac{1}{2}$ scaling factorially), creating these tables following the above procedure is not hard. Hence, if one just wants to know the coefficient of some word in a conserved quantity of some size $k$, one would just have to create the table below and select the right column. This can easily be done up to $k = 1000$. 

Conjecture 2.8.

$$C^p_{k,l} = (-n)^g C^p_{k,l+2}$$

(2.21)

Note that if a word has one hole, the number of gaps $g$ is also one. The coefficient is related to the coefficient of connected words greater in length by at least 3 monoids (given by the $l + 2h + g$ in the relation above). Hence, disconnected words only start appearing in $Q_k$ at length $l = (k - 3)$. There is a slightly intuitive explanation for the $2h$ in this relation: To create a hole from a local combination of generators, we have to make use of the rule $e_i e_{i+1} e_i = e_i$, which removes two generators from a word. It is no surprise that the coefficient of those words are related to coefficients of connected words with an increased length of 2.
2.2 Constructing new $Q_k$

(a) Table with $Z_{p,k,l}$ for $k = 10$. Note that for each row, the leftmost entry corresponds to $p = 0$, thus lines of constant $p$ follow a diagonal from top left to bottom right. The blue entries correspond to the conjecture 2.4 ($Z_{p,k,l}^p = (-1)^p$). The green entries correspond to equation 2.15 ($Z_{p,k,l}^n = (l - 1)^2$ for $k,l$ even). The red entries correspond to conjecture 2.7 ($Z_{p,k,l}^n = -Z_{p,k,l}^n$ for $k,l$ even). Two neighboring cells always sum to minus the cell directly above them.

(b) The entries selected to obtain the coefficient for the word $e_2e_1e_4e_3$ in the canonical anti-symmetrical form of conserved quantity $Q_{10}$. 
2.2 Constructing new $Q_k$

(a) Table with $Z_{p}^{k}$ for $k = 9$. Note that for each row, the leftmost entry corresponds to $p = 0$, thus lines of constant $p$ follow a diagonal from top left to bottom right. The blue entries correspond to the conjecture 2.4 ($Z_{p}^{k,l} = (-1)^{p}$). The green entries correspond to equation 2.18 ($Z_{p}^{k,l} = (l - 2)/2 + 1$) $\forall k \text{ odd}$, $\forall l \text{ even}$). The red entries correspond to conjecture 2.7 ($Z_{p}^{k,l} = 0$ $\forall k,l \text{ even}$). Two neighboring cells always sum to minus the cell directly above them.

(b) The entries selected to obtain the coefficient for the word $e_3 e_2 e_1$ in the canonical antisymmetrical form of conserved quantity $Q_9$. 

<table>
<thead>
<tr>
<th>$l \cdot k$</th>
<th>$I(w)$</th>
<th>$P(w)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi^{1/2}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$k \cdot l$</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>$-1$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$-1$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$-1$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>

$Z_{p}^{k,l}$ for $k = 9$.
3 Ground state statistics

3.1 Ground state vector

Motivated by the connections between the $O(n)$ loop model and physical models described in section 1.2, we want to find the ground state of the $O(n = 1)$ loop model on the cylinder. Additionally, we consider the $O(n = 1)$ loop model on a strip with reflecting boundary conditions introduced below.

We want to find the ground state vector of $H = -e_1$. As noted before, terms in expansion 1.14 of the transfer matrix commute. Let $A, B$ be two commuting linear operators and $\psi_a$ an eigenvector of $A$ with eigenvalue $\lambda_a$. For the product $A(B\psi)$, we get:

$$A(B\psi) = BA\psi = B\lambda_a\psi = \lambda_a(B\psi)$$  \hspace{1cm} (3.1)

Thus, the vector $B\psi$ is also an eigenvector of $A$ with the eigenvalue $\lambda_a$. This means that either the eigenvalue $\lambda_a$ is degenerate, or $B\psi = \lambda_b\psi$ and $\psi$ is also an eigenvector of $B$. Take the set of link patterns on an $L$-dimensional cylinder as a basis. The Hamiltonian $H$ maps one link pattern to a sum of $L$ new ones each with coefficient $-1$. Thus, in matrix form, $H$ is a real square matrix with only negative entries. Hence, its negative $Q_1$ is a positive square matrix, and it satisfies the Perron-Frobenius theorem:

**Theorem 3.1** (Perron-Frobenius). Let $A$ be a real positive square matrix. Then $A$ has a nondegenerate eigenvalue $\lambda_{pf}$ that is larger than all other eigenvalues. The eigenvector corresponding to this eigenvalue has only positive components.

Since $H = -Q_1$, the eigenvector of $Q_1$ with maximum eigenvalue corresponds to the eigenvector of $H$ with lowest eigenvalue, i.e. the ground state. The Perron-Frobenius theorem states that the eigenvalue $\lambda_{pf}$ is unique, which from the statements above also implies that all quantities $Q_k$ share the same eigenvector. The groundstate is thus also the eigenstate of the transfer matrix with highest weight. To find this state, the matrix power method is used where we apply $Q_1$ repeatedly to a vector in $LP_L$ and normalize after each step. The ground state will be expressed in the basis of link patterns. We could take our initial guess to be a sum of each unique link pattern in $LP_L$. However, from the equivalence to the Heisenberg $XXZ$-chain described in section 1.2.3, we expect all rotated versions of a link pattern to have the same coefficient. Furthermore, $Q_1$ itself is a sum over all positions of a monoid. Thus applying $Q_1$ to two link patterns that are rotated versions of one another yields the same result. Hence, it would not make much sense to calculate the action of $H$ on rotated versions of one link pattern separately. Rather, we identify all rotated versions of link patterns and only apply $Q_1$ to one of them. The result is then multiplied by the multiplicity $m(\pi^i)$ of that link pattern, defined as the number of times a link pattern can be rotated before it turns into itself. Note that the multiplicity of every link pattern for $L$ odd is equal to $L$. Let $ULP_L$ be the set of link patterns in dimension $L$ containing only one of all rotated versions of a link pattern and let $\psi_i$ denote our guess for the eigenstate after $i$ iterations of the power method. The initial guess is just a sum of all link patterns in $ULP_L$ times their multiplicity:

$$\psi_i = \sum_{\pi_j \in ULP_L} m(\pi_j)\alpha^i_{\pi_j} \pi^i$$ \hspace{1cm} (3.2a)

$$\alpha^0_0 = 1$$ \hspace{1cm} (3.2b)

![Figure 16: Link patterns with lowest weight.](image)

(a) Even system size. (b) Odd system size.
We want to normalize the ground state vector such that the link pattern in the O(n = 1) loop model corresponding to the state $\psi_{00...011...1}$ in the equivalent XXZ-chain is 1. Those link patterns are given in Figure 16. The vector is normalized after every application of $Q_1$ by dividing by the coefficient these states pick up. Let $\alpha_i^j$ be the coefficient of link pattern $\pi_j \in ULP_L$ after $i$ iterations. Furthermore, let $\tilde{\alpha}_i^{j+1}$ denote the coefficient of link pattern $\pi_j$ after application of $Q_1$ on $\psi_i$ but before normalization and $\tilde{\alpha}_0^{j+1}$ be the specific coefficients of the link patterns shown in Figures 16a and 16b. The final procedure is then given by:

$$\tilde{\psi}_{i+1} = \sum_{j=1}^{ULP_L} m(j) - H\tilde{\alpha}_i^{j+1}\pi_j := \sum_{j=1}^{ULP_L} \tilde{\alpha}_i^{j+1}\pi_j$$

$$\alpha_i^{j+1} = \frac{1}{\tilde{\alpha}_0^{j+1} m(j)} \tilde{\alpha}_i^{j+1}$$

Note that the $m(j)$ in the first line is used to create the new coefficients $\tilde{\alpha}_i^{j+1}$. After all results are summed over, the result is properly normalized by dividing by $m(\pi_j)$ again. By repeating this procedure, the vector converges to the Perron-Frobenius eigenvector. However, convergence can be slow. Furthermore, it is possible that this convergence is only achieved if the coefficients are tracked sufficiently precise. In our case, it is necessary to represent the coefficients as a bigfloat of 128 bits. After every procedure, the vector converges to the Perron-Frobenius eigenvector. However, convergence can be slow.

### 3.1.1 Cluster probabilities

We are now in position to analyze the results. Indeed for both odd and even system size the Razumov-Stroganov conjectures are verified. Furthermore, we are interested in how one can move from one point to any of them without crossing a red line. Figures 23a and 23b show two clusters of size 3. Note that the blue cluster extends all the way down until it meets a line that goes all the way around the cylinder, thus blocking any cluster from reaching further down. In the case of odd $L$, however, this cluster would extend down over the whole cylinder. Furthermore, it is possible to go all the way around the cylinder in this cluster, which is not possible in any other cluster (e.g. the green cluster in Figure 23b). There is always only one cluster that winds around the whole cylinder in this way (and extends infinitely far down if $L$ is odd). In the mapping to the disk, we can visualize this by adding a point in the relevant cluster marking the infinity. One can see this point as an opening, thus mapping a configuration to a link pattern on a punctured disk.

We want to know what the probability is to find a cluster of size $k$ given a system size of $L$ in the ground state of the system. Since the coefficients of the ground state are of integer value, these probabilities are rational numbers. The numerator is just given by $L$ times the sum of all entries in the ground state vector given in conjecture 13 times the multiplicity of the entry. We will assume the numerator to be of a similar form, i.e., given by a fraction of two products over some factorial terms dependent on $k$ and $L$. We can make use of this assumption in two ways. Firstly, we note that a factorial has a simple prime decomposition given by the decomposition of the terms in the product. Thus while the actual numbers we calculate are large, their prime decomposition results in a number of small primes typically between 1 and $3L$. Secondly, we can divide the probability on a cluster of size $k$ in a cylinder of size $L$ by the probability on a cluster of size $k$ in $L + 2$. If the probability contains a term such as $(a \cdot L + b)!!$, almost all terms will cancel and we are only left with the final term $(a \cdot (L + 2) + b)$.

In summary, we want to know the probability for a cluster of size $k$ appearing in a system of size $L$: 

$$\text{Probability} = \frac{\text{Coefficient in ground state}}{\text{Coefficient in } ULP_L}$$

...
3.1 Ground state vector

Figure 17: Difference between a winding cluster and a nonwinding cluster.
\[ P(L, k) = \frac{N(L, k)}{D(L)} \tag{3.4} \]

where the probability has been explicitly written as a division of two terms. \(N(L, k)\) stands for the total weight of clusters of size \(k\) in the ground state vector (i.e. the total number of clusters of size \(k\) in a link pattern times the weight of that link pattern in the ground state vector, summed over all entries in the ground state) and \(D(L)\) for the normalization constant (the total number of clusters in a link pattern times the weight of that link pattern in the ground state vector summed over all entries of the ground state vector). We expect these terms to be similar in form to the formula from the Razumov-Stroganov conjectures, i.e. a large division of several factorial terms dependent on \(L\) state vector). We expect these terms to be similar in form to the formula from the Razumov-Stroganov times the weight of that link pattern in the ground state vector summed over all entries of the ground state vector.

\[ D = \sum_{\text{patterns}} \text{weight of that link pattern in the ground state vector} \]

and

\[ N = \sum_{\text{all \ L-term in F}} \text{weight of clusters of size \(k\)} \]

Our strategy for finding exact forms for \(N(L, k)\) will be to first find a formula for a small component of the full formula by decomposing \(N(L, k)\) into prime factors and then calculate the fraction:

\[ F(L, k) = \frac{N(k, (L + 2))}{N(L, k)} = \frac{\Lambda(L + 2)M(L + 2, k)}{\Lambda(L)M(L)} \tag{3.6} \]

Thus the terms independent of \(L\) cancel out. Then we display all \(F(L, k)\) for \(L = 2\) to \(L = 26\) (even \(L\)) or \(L = 3\) to \(L = 23\) (odd \(L\)) at a given \(k\). We then have a list of divisions of relatively small prime terms in which we aim to observe a pattern. For example, for odd \(L\) we might observe a \(3\) in \(F(k, 3)\), a \(5\) in \(F(k, 5)\), a \(7\) in \(F(k, 7)\), \(3^2\) in \(F(k, 9)\) and so on. This leads us to the assumption that there is a factor of \(L\) in \(F(L, k)\). We can then make a guess \(G(L, k)\) and divide \(F(L, k)\). If most terms in the sequence get shorter, we keep our guess and continue this process until we find \(G(L, k) = F(L, k)\). After doing this for one value of \(k\), we move to \(k + 1\) and do the same. Here we might find a \(5\) in \(F(L, k)\), a \(7\) in \(F(k, 5)\) etc, from which we conclude the general function \(F(L, k)\) might contain a term like \((L + 2k)\). If it is possible to do this up to high enough \(k\)'s, we can construct a general \(F(L, k)\), from which the \(L\)-dependent part of \(N(L, k)\) can be retrieved. Then, a similar procedure is employed for the \(k\)-dependent part. Define:

\[ H(L, k) = \frac{N(k + 1, L)}{N(L, k)} \tag{3.7} \]

Again, we divide out guesses for terms in the numerator and the denominator until we are left with the exact expression for \(H(L, k)\). Note that it is only necessary to do this process at one specific \(L\), as all \(L\)-dependence is already captured in \(F(L, k)\). It is preferable to take the largest \(L\) available, as the size of a cluster \(k\) can only range from 1 to the integer part of \((L + 1)/2\): the cluster of maximum size is always the largest cluster in the diagram depicted in Figure 18.

![Figure 18: The largest possible cluster in odd and even system sizes is the integer part of \((L + 1)/2\).](image)

Having found both \(H(L, k)\) and \(F(L, k)\), we finally get the recursive relation:

\[ P(L, k) = F(k, (L - 2))P(k, (L - 2)) = F(k, (L - 2))F(k, (L - 4))P(k, (L - 4)) = \ldots \tag{3.8a} \]

\[ = P(k = 1, L = 2, 3) \prod_{m=1}^{(k-1)} H(L, m) \prod_{j=2}^{(L-2)} F(L, k) \tag{3.8b} \]

Where \(P(k = 1, L = 2, 3)\) denotes the initial term for \(L\) even or odd (either should be chosen, not both), and we just fill this factor in from the numerically found probabilities.
The main results are presented below. But first some remarks on the process described above. Firstly, the sequence $F(L,k)$ gets shorter for every increase in $k$: a system of size $2n$ can only contain clusters of maximum size $n$. Therefore, if we want to know the probability ratios for a high value of $k$, we can only use the probabilities $P(k,L = 2k)$ and up. However, the sequence $F(L,k)$ must be sufficiently large to be able to recognize common terms. Thus it is quite important to obtain the ground state vector for sufficiently large systems. Secondly, when one is formulating a guess $G(k,L)$ for the probability ratio $F(L,k)$ it is almost never possible to make $F(L,k)$ simpler for all $L$, as terms in the numerator can be canceled terms in the denominator in some $F(L,k)$. One just searches for a general trend that makes most of the terms simpler. Hence, this process is susceptible to error and it was necessary to remove terms from $G(L,k)$ again later on that turned out to be wrong. Finally, our assumption that $F(L,k)$ consisted of only small primes was not quite correct, as there were some larger primes of order $L^2$. To solve this, as many of the other terms were guessed such that a sequence containing mostly the larger primes remained. This sequence was entered in the On-Line Encyclopedia of Integer Sequences (OEIS) which identified the term.

A detailed example of the process of finding a probability formula is given in Appendix 7.0.2. The probability on a cluster of size $k$ in an even system size $L = 2n$ is given by:

$$P(L = 2n, k) = \frac{3 \left( \frac{L^2}{4} + \frac{k}{2} \right) (3k^2 - 1) (2k - 3)!! (L - k - 1)! (L + 2k - 2)!!}{(2k - 2)!! (k + 1) (L + k)! (L - 2k)!!}$$

(3.9)

where the term $L^2/4 + (k/2)(3k^2 - 1)$ was found using the OEIS. In this probability, it is not specified whether or not the cluster is the winding cluster or not. Let the parameter $d$ denote the minimum number of lines one would have to cross to go from a cluster to the winding cluster in a link pattern (thus the winding cluster itself has $d = 0$). We can split the probabilities for a cluster of size $k$ in a system of size $L$ ($P(L,k)$) to also specify this parameter, thus resulting in the probability on a cluster of size $k$ in a system size $L$ given that one has to cross a minimum of $d$ lines to go from that cluster to the winding cluster ($P(L,k,d)$). For the winding cluster, we get:

$$P(L = 2n, k, d = 0) = \frac{3 k (2k - 1)!! (L - k - 1)! (L + 2k - 2)!!}{2 (2k - 2)!! (L + k)! (L - 2k)!!}$$

(3.10)

Hence the probability to have a cluster of size $k$ that is not the winding cluster $P(L,k,d > 0)$ is the difference of these two equations ($P(L,k,d > 0) = P(L,k) - P(L,k,d = 0)$).

For odd system sizes, the same procedure is applied. Note that there are always two edges on the boundary that are in the winding cluster. Therefore the maximum number of edges that not in the winding cluster is $(L - 1)/2$, whereas the maximum number of edges containing the cluster is one higher.

The following formula was found for $d = 0$:

$$P(L,k,d = 0) = \frac{4^{k-1} (3L - 1)6 (6k - 8)6 (2k - 3)!! 3k (L - k - 1)! (2k - 1) (L + 2k - 5)!! (3k - 1)! (L + 2k - 3)}{3^{k-1} (3L - 5)6 (6k - 4)6 (3k - 4)3^2 L (k - 2)! (4k - 4)!! (L + k)! (L - 2k + 1)!!}$$

And for $d > 0$:

$$P(L,k,d > 0) = \frac{3 \cdot (2k - 3)!! \cdot k \cdot (L + 2k - 1)!! \cdot (L - k)!}{(2k + 2)!! \cdot L \cdot (L - 2k - 1)!! \cdot (L + k - 1)!}$$

(3.12)

### 3.2 Open boundary conditions

The above procedure was repeated for the $O(n = 1)$ loop model on a strip with reflecting boundary conditions. This model achieved by placing a triangle on the sides of every two layers connecting the boundary lines from those two layers (see Figure 19). Consequentially, operators can no longer be of even layer size, as this would take a system of size $L$ to a system of size $L + 2$. Thus, the transfer matrix is now a two-layer operator:

\[ \text{https://oeis.org} \]
3.2 Open boundary conditions

The Hamiltonian is again defined by the sum of all terms in order $b$ in the transfer matrix, giving:

$$H_{\text{strip}} = \sum_{i=1}^{L}$$

Link patterns now live on a line instead of a disk:

Figure 20: Link pattern on a strip

Link patterns can no longer be rotated to produce equivalent versions with equal coefficients in the ground state eigenvector. Rather, we identify all link patterns generated by mirroring a link pattern in the vertical axis. Thus the multiplicity of a pattern is either 1 or 2. The link pattern with the lowest coefficient is now given by:

Figure 21: The link pattern with minimal coefficient in the ground state vector.

Using the same algorithm as before, we are able to obtain the ground state vector for system sizes up to $L = 20$. The conjectures in [1] are verified.
3.2 Open boundary conditions

(a) Even case: one infinite cluster that connects both boundaries.

(b) Odd case: two disconnected infinite clusters.

The equivalent of a winding cluster in this system differs for $L$ even and $L$ odd: For even $L$, there is one cluster that connects both boundaries, depicted in Figure 22a that is infinite in size. However, for the odd case, the unconnected line separates two different infinite clusters that can now no longer connect the other way around the cylinder. Thus, there are two winding\(^{12}\) clusters per diagram, shown in Figure 22b.

The same method was employed in an attempt to find similar probabilities $P(L, k, d)$ on the strip. The first result is promising: using this method, it is possible to find the probability on a cluster of size $k$ given that it is connected to the boundary for both even $L$ and odd $L$ $P(L, k, d = 0)$. We get:

$$P(L = 2n - 1, k, d = 0) = \frac{(3L + 1)!^6 (L + 2k - 2)! (L - k)!}{(3L - 4)!^6 (L - 2k + 1)! (L + k)! (L + 1)}$$

(3.15)

And for even system sizes:

$$P(L = 2n, k, d = 0) = \frac{2^{(k-1)}(3L + 2)!^6 (L + 2k - 3)! (L - 1)! (L - k)! (3k - 2)! (2k - 3)!^k}{3^{(k-1)}(3L - 5)!^6 (6k - 4)!^6 (2k - 3)! (k - 2)! (L + 1)! (L + k)! (l - 2k + 2)! (3k - 4)^3}$$

(3.16)

However, when we try to apply this method to cases $d > 0$, it breaks down. Unfortunately, the prime decomposition for the probability on a cluster of size $k$ that is not the winding cluster contains very large prime factors, e.g. $F(L = 14, k = 3, d > 0)$ contains the prime 269958407. This leads us to the conclusion that it is not a simple product of factorial terms. The large primes can be the result of a sum of such terms, as this can lead to arbitrary prime decompositions and is not 'simple' as a factorial term. Our first guess would be to split the probability $P(L, k, d > 0)$ into all separate values for $d$. From the groundstate eigenvector we can compute the new probabilities keeping track of the distance from each cluster to the winding cluster. Then, we employ the same method for $P(L, k, d = 1)$, $P(L, k, d = 2)$, etc. Unfortunately this again does not always produce simple prime factorizations. However, we observe that some terms in these probabilities do always produce simple prime factorizations. Given a certain cluster size $k$, there is a maximum number of lines that cluster can be separated from the winding cluster given by:

\(^{12}\)Winding cluster is no longer an appropriate name if the cluster does not actually wind around, but we will continue using it for consistency.
3.2 Open boundary conditions

\[(k + d) = \begin{cases} 
\left(\frac{L}{2} - 1\right) & L = 2n \\
\frac{L+1}{2} & L = 2n - 1
\end{cases} \tag{3.17}\]

I.e. if we have some cluster of size \(k\) that is maximally nested within other clusters and thus has a maximum number of lines one would need to cross to get to the winding cluster. This specific case does always produce a simple prime decomposition and the algorithm described above can be employed. Thus, on the strip it is possible to find the probability on a cluster of size \(k\) given that it is either the winding cluster itself, or it has a maximal distance \(d\) from the winding cluster. Note that there is always only one link pattern (plus its rotations/reflections) that has a cluster that satisfies equation 3.17. Examples are shown in Figure 23.

![Figure 23: The most nested cluster for \(L = 10\) and different \(k\) and \(d\) for periodic boundary conditions. Note that \(k + d = \left(\frac{L}{2} + 1\right)\), to which only one link pattern contributes.](image)

Using the method described above, we are able to find a general formula for the probabilities on these clusters. However, it was slightly more complicated, as the probability ratios \(F(L, k, d) = P(L + 2, k, d)/P(L, k, d)\) were still very complicated. It was necessary to go one step further and take ratios of the ratios to obtain terms that had a simple prime decomposition, i.e. \(H(L, k, d) = F(L + 2, k, d)/F(L, k, d)\). This resulted in not just a division of several factorial terms, but a product over such terms as the total result. These computations were also performed for the system with closed boundary conditions.

Since \(k\) is uniquely determined by the value of \(d\), the probability only depends on one of both parameters. \(d\) was chosen in this case, but using equation 3.17, the equations below can be rewritten to functions of \(k\). We arrive to the following formula:
For periodic boundary conditions, $L$ even:

$$P(L, d, k = (\frac{L}{2} - d)) = \frac{1}{2} \prod_{\alpha=1,2,...}^{(d-1) \quad \text{(3.2)}} \frac{\text{3} \quad \text{2} \quad (2\alpha + 2)! \quad (2\alpha + 1)! \quad (2\alpha - 1)!^2}{(\alpha - 1)!^2 \quad (3\alpha + 3)! \quad (3a)!} \times \prod_{\beta=(2d), (2d+2),...}^{(L-2)} \frac{16 \quad (\beta + 1)!! \quad (3\beta + 2d - 2)!! \quad (3\beta - 2d)!! \quad (\beta + 1)!! \quad (\beta - 2d + 4) \quad (\beta - 2d - 1)!! \quad (3\beta - 2)!! \quad (3\beta + 4)!! \quad (L + 2d - 1)!! \quad (3L - 6d + 6)}{\text{(3.2)}}$$

For periodic boundary conditions, $L$ odd:

$$P(L, d, k = (\frac{L-1}{2} - d)) = \frac{1}{4} \prod_{\alpha=1,2,...}^{(d-1) \quad \text{(4.4)}} \frac{(4\alpha + 5)! \quad (4\alpha + 3)! \quad (2\alpha + 2)!^2}{(3\alpha + 6)! \quad (3\alpha + 4)! \quad (3\alpha + 2)! \quad (3\alpha + 3)! \quad (6\alpha + 5)! \quad (6\alpha + 1)!} \times \prod_{\beta=(2d+1), (2d+3),...}^{(L-2)} \frac{(2\beta + 3)! \quad (\beta - 2d + 3)!! \quad (3\beta + 2d - 2)!! \quad (3\beta - 2d - 2)!! \quad (\beta + 1)!! \quad (\beta + 1)!! \quad (\beta - 2d + 1) \cdot (\beta + 3)}{(2\beta - 1)! \quad (\beta - 2d)!! \quad (3\beta + 5)!! \quad (3\beta + 2)!! \quad (\beta + 2d + 1)!! \quad (\beta + 2d - 1) \cdot (\beta + 3)}$$

For the strip with reflecting boundary conditions, $L$ even:

$$P(L, d, k = (\frac{L}{2} - d)) = \frac{1}{3} \prod_{\alpha=1,2,...}^{(d-1) \quad \text{(3.2)}} \frac{3 \quad (4\alpha + 3)! \quad (4\alpha + 1)! \quad (4\alpha + 2)!^2}{2 \quad (6\alpha + 9)! \quad (6\alpha + 1)! \quad (3\alpha + 2)! \quad (3\alpha + 1)!} \times \prod_{\beta=(2d), (2d+2),...}^{(L-2)} \frac{(\beta - 2d + 4)!! \quad (3\beta + 2d + 2)!! \quad (\beta - 1)!! \quad (\beta + 2)!! \quad (3\beta - 2d - 1)!! \quad (\beta - 2d - 1)!! \quad (L + 2d)!! \quad (3L - 1)!!}{\beta \quad (2\beta - 1)! \quad (3\beta + 6)! \quad (L - 2d - 1)!! \quad (L + 2d)!! \quad (3L + 1)!!}$$

For the strip with reflecting boundary conditions, $L$ odd:

$$P(L, d, k = (\frac{L-1}{2} - d)) = \frac{1}{4} \prod_{\alpha=1,2,...}^{(d-1) \quad \text{(4.4)}} \frac{(4\alpha + 5)! \quad (4\alpha + 3)! \quad (2\alpha + 2)!^2}{(3\alpha + 6)! \quad (3\alpha + 4)! \quad (3\alpha + 2)! \quad (3\alpha + 3)! \quad (6\alpha + 5)! \quad (6\alpha + 1)!} \times \prod_{\beta=(2d+1), (2d+3),...}^{(L-2)} \frac{(2\beta + 3)! \quad (\beta - 2d + 3)!! \quad (\beta + 1)!! \quad (\beta + 1)!! \quad (\beta - 2d - 2)!! \quad (\beta + 2d + 1)!! \quad (\beta + 2d - 1)!! \quad (\beta + 3)}{(2\beta - 1)! \quad (\beta - 2d - 2)!! \quad (3\beta + 5)!! \quad (3\beta + 2)!! \quad (\beta + 2d + 1)!! \quad (\beta - 2d + 1) \cdot (\beta + 3)}$$
4 Conclusion

The main result of this thesis is a conjecture of a recursive description for conserved quantities in the dense $O(n)$ loop model with periodic boundary conditions. Using computational methods, it was possible to generate these quantities up to $\tilde{Q}_{10}$. Each $\tilde{Q}_k$ was put in an (anti-)symmetrical form $\tilde{Q}_k$ by adding appropriate combinations of previous quantities $\tilde{Q}_{k'}$. This was done by hand and we have not found a general formula for this procedure. Instead, several observations on the coefficient of reduced words in $\tilde{Q}_k$ were made leading to a full description:

- Reduced words that contain a generator in one position twice always have coefficient zero (conjecture 2.2);
- Words of maximum length $k$ are connected and have coefficient $(-1)^{p(w)}$ in $Q_k$ (conjecture 2.4);
- There is a recursive relation between coefficients of connected words of length $l$ and connected words of length $l + 1$ (conjecture 2.5);
- Using this recursive relation and if all coefficients of connected words of some length $l + 1$ are known, we can find the coefficients for connected words of length $l$ generating an initial condition either from symmetry arguments (equations 2.15 and 2.17) or conjecture 2.7;
- Coefficients for disconnected words can be obtained from coefficients of connected words via conjecture 2.8.

Using this description, we can generate the full table of $Z$-coefficients. To assign the correct polynomial coefficient to a given (reduced) word, one needs to select the correct terms and add factors of $n$ as described in section 2.2. This should be done for every unique word of length $\leq k$. The procedure used to generate all unique reduced words of some length $k$ can be found in the appendix 7.1.5. Further conserved quantities up to and including $\tilde{Q}_{17}$ were generated and it is verified that they indeed commute with the Hamiltonian.

Unfortunately, a closed formula for the $Z$-coefficients is still lacking. The only way to generate them as of now are by doing it recursively starting from $Z_{k,k}^p = (-1)^p$. Additionally, there are many variants on the dense $O(n)$ loop model. One can consider different boundary conditions or variants where (half) empty plaquettes are allowed. The methods employed in this thesis are capable of generating conserved quantities in all such models that are described by a commuting family of transfer matrices. However, the results may not always yield a simple description, and as such it might not be possible to guess the correct form of $Q_k$ for other variants of these models. Furthermore, there might be other (nonlocal) conserved quantities algebraically independent from the ones we found.

Driven by connections between the $O(n = 1)$ loop model, several other (more physical) models such as the Heisenberg XXZ-chain, and combinatorics, the groundstate vector was computed for the O($n = 1$) loop model. This was done both on a semi-infinite cylinder and on a semi-infinite strip with reflecting boundary conditions. The Von Mises power method was used to obtain these vectors for system sizes up to $L = 28$ (cylinder) and $L = 20$ (strip). The conjectures made in [13] and [1] were verified.
By displaying the prime decompositions of ratios of probabilities, we achieved to conjecture a formula
for the following probabilities:

- **Cylinder, even size:**
  - The probability for a cluster of size $k$ in a system of size $L$: $P(L, k)$
  - The probability for a cluster of size $k$ in a system of size $L$ given that it is the winding cluster
    (number of lines one has to cross to get to the winding cluster $d = 0$): $P(L, k, d = 0)$
  - The probability for a cluster of size $k$ in a system of size $L$ given that it is not the winding
    cluster ($d > 0$): $P(L, k, d > 0) = P(L, k) - P(L, k, d = 0)$
  - The probability for a cluster of size $k$ in a system of size $L$ given that the number of lines one
    has to cross to get to the winding cluster is maximal: $P(L, (k + d) = (\frac{l + 1}{2}))$

- **Cylinder, odd size:**
  - The probability for a cluster of size $k$ in a system of size $L$: $P(L, k)$
  - The probability for a cluster of size $k$ in a system of size $L$ given that it is the winding cluster
    ($d = 0$): $P(L, k, d = 0)$
  - The probability for a cluster of size $k$ in a system of size $L$ given that it is not the winding
    cluster ($d > 0$): $P(L, k, d > 0) = P(L, k) - P(L, k, d = 0)$
  - The probability for a cluster of size $k$ in a system of size $L$ given that the number of lines one
    has to cross to get to the winding cluster is maximal: $P(L, (k + d) = (\frac{l + 1}{2}))$

- **Strip with reflecting boundary, even size:**
  - The probability for a cluster of size $k$ in a system of size $L$ given that it is the winding cluster
    ($d = 0$): $P(L, k, d = 0)$
  - The probability for a cluster of size $k$ in a system of size $L$ given that the number of lines one
    has to cross to get to the winding cluster is maximal: $P(L, (k + d) = (\frac{l}{2} + 1))$

- **Strip with reflecting boundary, odd size:**
  - The probability for a cluster of size $k$ in a system of size $L$ given that it is the winding cluster
    ($d = 0$): $P(L, k, d = 0)$
  - The probability for a cluster of size $k$ in a system of size $L$ given that the number of lines one
    has to cross to get to the winding cluster is maximal: $P(L, (k + d) = (\frac{l + 1}{2}))$

Unfortunately, it was not possible to find a formula $P(L, k, d > 0)$ for both even and odd system
sizes on the infinite strip with reflecting boundary conditions. This was mainly due to the large prime
factors that appeared in the prime decomposition for $P((L + 2), k, d > 0)/P(L, k, d > 0)$. It could be
true that these large primes from lumping together clusters with certain properties in the same manner
that we were lumping together $P(L, k, d = 1)$, $P(L, k, d = 2)$ etc. While decomposing $P(L, k, d > 0)$ into
these specific terms did not solve our problem, it might very well be that there exists some property by
which we can separate these probabilities such that a description in terms of products of factorial terms
is possible. If so, the main challenge is to ask the correct questions. For example, we have tried to look
at the distance of a cluster to the boundary not in terms of lines one has to cross before reaching the
boundary $d$, but rather as the index of the closest edge in a cluster to the boundary. Unfortunately this
did not produce better results.
5 Discussion

While the conjectures made in this thesis are thoroughly checked by using them to generate new examples, they remain conjectures. A proof that they are indeed valid is still missing. Furthermore, for these conserved quantities it was always assumed that the system size was sufficiently large that \((L > k)\) such that all terms \((d/db)^n T(b)\) were nonzero. An analysis should be made for cases where this is not valid. However, it is only necessary for the number of conserved quantities in a system to be unbounded in the infinite size limit of the system \([5]\), which is certainly the case.

Two different representations of the Temperley-Lieb algebra were used for computing the conserved quantities. Initially, operators were represented by a hash table linking points from one layer to points in the next layer (also called the diagrammatic representation, see Appendix 7.1.3 for more details). There are a few benefits to this method. When one applies two operators successively (equivalent to taking the product of two operators), the resulting operator can be found very easily by making a new hash table that implements the new point-to-point map directly. Furthermore, every equivalent version of a word produces the exact same hash table, whether or not the word used to construct the table is reduced. However, comparing if two hash tables are equal is a slow process, and it was not feasible to use this method for higher order quantities than \(Q_7\). The second representation of operators in \(TL_a\) were lists of integers where an integer denoted the position of an \(\tilde{e}_i\). While this produced a faster algorithm, it has some downsides. In this representation, it is not immediately clear whether or not a word is reduced. So an algorithm was necessary to map a word to a reduced form (see Appendix 7.1.1). Furthermore it was necessary to keep track of all equivalent reduced versions of all unique reduced words in the algebra. This was done by making one large hash table, linking every form of a reduced word to one arbitrary equivalent version for every unique reduced word in the algebra.

To calculate the groundstate vector of a system, the Lanczos-method could provide a more efficient algorithm than the Von Mises power method. They are similar in that they are based on repeated application of the Hamiltonian and normalization of the resulting vector, but the Lanczos-method is in general more efficient. One could use this to produce the groundstate eigenvector for larger system sizes. For our purposes the obtained results were sufficient.
6 Acknowledgements

I wish to thank Bernard Nienhuis for all his guidance and patience throughout the process of this thesis. His advice helped me not only on the topic of physics but in my life in general and I always felt welcome to discuss any problems and questions I had. I wish to thank my parents Paul and Tiemke for their moral and financial support throughout the years, as well as providing access to a virtual machine with enough memory to calculate $Q_{10}$. Furthermore I wish to thank Twan Koperberg for his thorough and useful feedback on this thesis. All of the code and all of the images used in this thesis were created using plt racket. Finally, thanks to the makes of the OEIS for their great site.

Producing this thesis has taken a long time (substantially longer than the one year that was advised). There are several reasons. Firstly, I have not always been able to put as much work in this project as I might have wanted for reasons ranging from responsibilities in activities not related to study to combining part of the writing with a PhD. Secondly, producing conserved quantities proved to be a much harder task than initially thought. Even after lots of bug fixes, the outcomes of the code produced wrong results for a long time. It was only once we checked the results for relatively high order ($Q_{5}$) by hand that we discovered to our surprise one of our main assumptions was incorrect\footnote{We first assumed $[T^n(b), T^m(b)] = 0$ which is not valid if we only take the locally constructed terms into account.} causing wrong results rather than a bug in the code. Finally, the focus of this thesis was only shifted to conserved quantities after we found that the results for calculating probabilities on the strip with reflecting boundary conditions (the second part of this thesis) were not as good as we had hoped. While the time line might have not been ideal for me and the people around me, I am very satisfied with the final results.
7 Appendix

The appendix is split into two main parts. The first part provides some background on topics discussed in the thesis. The second part outlines some of the most important algorithms that were used to calculate the conserved quantities.

7.0.1 Alternating sign matrices

An alternating sign matrix is a square \((m \times m)\) matrix with entries \(-1, 0,\) and \(1\) with the properties:

- Every row and every column sums to 1;
- The nonzero elements alternate between 1 and \(-1\) in each row and each column.

The number of unique AMS's of size \(n \times n\) is given by:

\[
A_n = \prod_{j=0}^{(n-1)} \frac{(3j + 1)!}{(n + j)!}
\]  

7.0.2 Example calculation of probability formula

In this section, we will perform in detail the calculation of \(P(L = 2n+1, k, d = 0)\) for periodic boundary conditions, i.e. the probability of having a cluster of size \(k\) appearing given that it is the winding cluster in a system of odd size. All formula for probabilities on clusters of a certain size were found via this technique. The ground state vectors for odd system sizes were calculated up to and including \(L = 23\). Firstly, note that for odd system sizes, the winding cluster is at least of size 2, since the two edges on both sides of the line connected to the infinity are always connected to each other. Thus \(P(L = 2n+1, k = 1, d = 0) = 0\) and we start with \(k = 2\). The sequence \(F(L, k = 2) = P(L + 2, k = 2, d = 0)/P(L, k = 2, d = 0)\) is given by:

\[
\begin{align*}
F(3, 2) &= \frac{3}{5^2} \\
F(5, 2) &= \frac{5^2}{2^1 \times 3 \times 7} \\
F(7, 2) &= \frac{5 \times 7 \times 13}{3^2 \times 11^2} \\
F(9, 2) &= \frac{2^3 \times 3^2}{11 \times 13} \\
F(11, 2) &= \frac{3 \times 11 \times 19}{5 \times 13 \times 17} \\
F(13, 2) &= \frac{11^2 \times 13}{2 \times 3 \times 5^2 \times 17} \\
F(15, 2) &= \frac{3 \times 5^1 \times 13}{17 \times 19 \times 23} \\
F(17, 2) &= \frac{2 \times 5 \times 17}{13 \times 19} \\
F(19, 2) &= \frac{17 \times 19 \times 31}{3 \times 7 \times 23 \times 29} \\
F(21, 2) &= \frac{3 \times 7 \times 17 \times 19}{2^4 \times 5^2 \times 23}
\end{align*}
\]  

Now we try to formulate a guess for this sequence \(G(L, k = 2)\) and divide \(F(L, k = 2)\) by this guess to see if the result becomes simpler. We repeatedly add terms to \(G(L, k = 2)\) until \(F(L, k = 2)/G(L, k = 2)\) becomes unity. Firstly, we note that the term \((L - 2)\) seems to appear in the numerator of the above terms. In some \(F\) it is only partly present, e.g. for \(L = 17\), we notice a 5 in the numerator but no 3. The assumption is that the 3 canceled a 3 in the denominator. Thus we only look for terms that appear in most terms in the sequence. We can also identify an \(L\) in the numerator and \((L + 2)\) in the denominator. So our first guess will be:
\[ G(L, k = 2) = \frac{L \times (L - 2)}{(L + 2)} \] (7.3)

Then the sequence \( F(L, k = 2) / G(L, k = 2) \) becomes:

\[
\begin{align*}
\frac{F(3, 2)}{G(3, 2)} &= \frac{1}{5} \quad (7.4a) \\
\frac{F(5, 2)}{G(5, 2)} &= \frac{5}{2^2 \times 3^2} \quad (7.4b) \\
\frac{F(7, 2)}{G(7, 2)} &= \frac{13}{11^2} \quad (7.4c) \\
\frac{F(9, 2)}{G(9, 2)} &= \frac{2^2}{7 \times 13} \quad (7.4d) \\
\frac{F(11, 2)}{G(11, 2)} &= \frac{19}{3 \times 5 \times 17} \quad (7.4e) \\
\frac{F(13, 2)}{G(13, 2)} &= \frac{11}{2 \times 5 \times 17} \quad (7.4f) \\
\frac{F(15, 2)}{G(15, 2)} &= \frac{5^2}{19 \times 23} \quad (7.4g) \\
\frac{F(17, 2)}{G(17, 2)} &= \frac{2}{3 \times 13} \quad (7.4h) \\
\frac{F(19, 2)}{G(19, 2)} &= \frac{31}{23 \times 29} \quad (7.4i) \\
\frac{F(21, 2)}{G(21, 2)} &= \frac{17}{2^4 \times 5^2} \quad (7.4j)
\end{align*}
\]

(7.4k)

Only a few terms remain in the numerators. For \( L = 19 \), there is a 31 in the numerator and for \( L = 21 \) a 17. Then we presume the 17 resulted from a term close 31, and the most obvious term we can think of is \( 2 \times 17 = 34 \) where the 2 was canceled with a term in the denominator. This would be described by the sequence \((3L + 5)/2\). Indeed, we find lots of evidence in other terms as all remaining factors in the numerator are part of the decomposition of such a term. Dividing out \((3L + 5)/2\), only terms in the denominators remain and it is easy to identify them as \((L + 4)\) and \((3L + 1)/2\). In summary:

\[ F(L, k = 2) = \frac{L \times (L - 2) \times (3L + 5)}{(L + 2) \times (L + 4) \times (3L + 1)} \] (7.5)

From this formula we can recreate the probability on a winding cluster of size 2. The lowest system size that can have a winding cluster of size 2 is \( L = 3 \) and from our numerical results we know \( P(L = 3, k = 2) = \frac{2}{3} \) (omitting the \( d = 0 \) henceforth), thus we get:

\[
\begin{align*}
P(L, k = 2) &= P(L = 3, k = 2) \prod_{i=3}^{(L-2)} F(i, 2) \\
&= P(L = 3, k = 2) F(3, 2) F(5, 2) ... F((L - 2), 2) \quad (7.6a) \\
&= \frac{2}{3} \frac{(3 \times 5 \times \ldots \times (L - 2)) (1 \times 3 \times \ldots \times (L - 4) (14 \times 20 \times \ldots \times (3(L - 2) + 5))}{(5 \times 7 \times \ldots \times L)(7 \times 9 \times \ldots \times (L + 2)(10 \times 16 \times \ldots \times (3(L - 2) + 1))} \\
&= \frac{2 \times 3 \times (3 \times 5) \times 4 (L - 2)! \times (L - 4)! \times (3L - 1)!}{3 \times (2 \times 8) \times L! \times (L + 2)! \times (3L - 5)!} . \quad (7.6d)
\end{align*}
\]

(7.6d)

where the numerical factor in front is needed since not every factorial term starts at 1.
Now we can repeat the same procedure for \( k = 3 \). Note that a winding cluster of size 3 can only appear in system sizes 5 and up. We find the exact same procedure we find up to \( k \) by 2 thus raising the lowest index of the product in equation 7.10a by two. This is what generates the following generalization:

\[
P(L, k = 3) = P(5, 3) \prod_{i=5}^{(L-2)} F(i, 3)
\]

(7.7c)

\[
P(L, k = 4) = \frac{5 \times 7 \times 9 \times 11 (3L - 1)6 \times (L - 2)! \times (L - 5)! \times (L - 6)! \times (L + 5)!}{2^3 \times 3 \times (3L - 5)!6 \times (L - 7)! \times L! \times (L + 3)! \times (L + 4)!}
\]

(7.7a)

\[
P(L, k = 5) = \frac{3^2 \times 5^2 \times 7^2 (3L - 1)6 \times (L - 7)! \times (L - 6)! \times (L - 2)! \times (L + 7)!}{2^5 \times (3L - 5)!6 \times (L - 9)! \times L! \times (L + 4)! \times (L + 5)!}
\]

(7.7b)

\[
P(L, k = 6) = \frac{(3L - 1)6 \times (L - 7)! \times (L - 6)! \times (L - 2)! \times (L + 7)!}{(3L - 5)!6 \times (L - 9)! \times L! \times (L + 4)! \times (L + 5)!}
\]

(7.7c)

(7.10b)

For higher \( k \) we do not find the same function \( F(L, k) \), but additional terms appear. Using the same procedure we find up to \( k = 6 \) (where the numerical terms independent of \( L \) have been simplified):

\[
P(L, k = 7) = F(L, k = 7)
\]

(7.9)

This leads to the following generalization:

\[
P(L, k) = P((2k - 1), k) \prod_{i=(2k-1)}^{(L-2)} F(i, k)
\]

(7.10a)

\[
P(L, k) = (3L - 1)6 \cdot (6k - 8)6 \cdot (L - k - 1)! \cdot (2k - 1)! \cdot (L + 2k - 3)! \cdot (3k - 1) \cdot (L + 2k - 3)
\]

As noted above, for each increasing \( k \) the lowest system size that can contain such a cluster increases by 2 thus raising the lowest index of the product in equation 7.10a by two. This is what generates the \( k \)-dependent terms. Take the term \((L - 2)!\) as an example. It resulted from a factor \( L \) in \( F(L, k) \), but for increasing \( k \) the initial index for the product is raised by two thus leading to a term \((L - 2)!/(2k - 3)!!\).

We can expand further equation 7.10a by finding an expression for \( P((2k - 1), k) \). These terms should only depend on \( k \), as all \( L \)-dependence is already captured in the above equation. To find these terms we employ the same method where we define the ratio of two such subsequent terms:
\[ K(k) = \frac{P(2(k + 1) - 1, (k + 1))}{P(2(k + 1), k)} \]  

(7.11)

Then we display \( K(k) \) for increasing \( k \) where we decompose the numerator and the denominator and make a guess of what terms appear in the expression. Then we divide out our guess and repeat this process until we have a full expression. This gives:

\[
K(k) = \frac{4 \times (2k - 1)^2 \times (k + 1)}{3 \times (3k - 1)^2 \times k} \quad (7.12a)
\]

\[
P((2k - 1), k) = \frac{4^{(k-1)} \times (2k - 3)!^2 \times k}{3^{(k-1)} \times (3k - 4)^2} \quad (7.12b)
\]

Combining these gives the final expression:

\[
P(L, k) = \frac{4^{k-1} \cdot (3L - 1)6 \cdot (6k - 8)6 \cdot (2k - 3)!^2 \cdot k \cdot (L - k - 1)! \cdot (2k - 1) \cdot (L + 2k - 5)!! \cdot (3k - 1) \cdot (L + 2k - 3)}{3^{k-1} \cdot (3L - 5)6 \cdot (6k - 4)6 \cdot (3k - 4)^2 \cdot L \cdot (k - 2)! \cdot (4k - 4)! \cdot (L + k)! \cdot (L - 2k - 1)!!} \quad (7.13b)
\]

This method was used in all expressions of probabilities that were found in this thesis, although it was sometimes necessary to take the ratio of two probability ratios \( F(L + 2, k)/F(l, k) \) instead of just the ratio.

### 7.1 Algorithms

In this appendix, a brief overview is given of the most essential the algorithms that were used in this thesis. All code was written in PLT Racket.

#### 7.1.1 Reducing a word

Given a general word, we want an algorithm that puts it in a reduced form. Note that any way to decrease the number of generators in a word involves the action of two generators with the same index. This algorithm is based on identifying the positions of those double generators and attempting to push them towards each other such that either TL-rule 1.21a or 1.21b can be used to rewrite them as a single generator. There are some difficulties involved. Take for example the word \( e_1 e_2 e_3 e_4 \). Looking at the two \( e_2 \)'s in the word, it looks impossible to push them towards each other, as \( e_2 \) does not commute with both \( e_1 \) and \( e_3 \), and the part of the word that blocks this operation \( e_2 e_3 \) cannot be simplified. However, looking at the two \( e_2 \)'s in the word, we can commute them to get the word \( e_2 e_1 e_3 e_4 e_2 \), which reduces to \( e_2 e_1 e_2 \). If we look at the two \( e_2 \)'s now, we see that the whole word can be reduced to \( e_2 \). Thus it matters in which order we consider pairs of generators.

First, we write an algorithm to identify the position of all double generators. This code is not very advanced thus we will provide no further details. The output looks like\(^{[14]}\)

\[
\text{Doubles}(w) : w \rightarrow ((i_1 \ p_1 \ p_2)(i_2 \ p_1 \ p_2)\ldots(i_n \ p_1 \ p_2)) \quad w \in TL_a
\]

(7.14)

where \( i_j \) stands for the index of a generator that appears at least twice, \( p_1 \) for the first position and \( p_2 \) for the second position \( (p_1 < p_2) \). If a generator appears more than twice, this function returns all neighboring pairs of that generator. E.g. for the word \( e_1 e_2 e_3 e_4 e_2 e_1 \), we get as output the list \( '((1 \ 1 \ 4)\ (1 \ 4 \ 6)\ (2 \ 2 \ 5)) \).

Then the following algorithm gives a reduced version of a word:

\(^{[14]}\)Using LISP notation for lists, where the brackets ‘(’ denotes a list (and all nested brackets are also lists).
Let \( w \) be a word and \( d = D(w) \) be the list of double entries in that word provided above. Let \( o \) be the order of \( n \) that \( w \) is proportional to.

1. If \( d \) is empty, \( w \) is the reduced word we are looking for.

2. Take the first index \( i_1 \) and position pairs \( p_1 \) and \( p_2 \) from \( d \). Let \( sw \) be the subword of \( w \) given by the generators on the positions \((p_1 + 1)...(p_2 - 1)\), i.e. the part in between the positions \( p_1 \) and \( p_2 \). Let \( b \) be all generators in the subword with index \( e_j = e_{i+1} \).

   (a) If the length of \( |b| = 0 \), there are no generators preventing \( e_i \) position \( p_{i1} \) to commute to the one on position \( p_2 \). Let \( w' \) be the same word as \( w \) minus the generator on position \( p_2 \). Repeat this algorithm from the top with \( w_{new} = w' \), \( o_{new} = o + 1 \) and \( d_{new} = D(w') \).

   (b) If the length of \( b = 1 \), there is only one \( e_i \) in between the \( e_i \) such that we can use equation 1.21b.

      Let \( w' \) be equal to \( w \) missing the generator \( e_i \) on position \( p_1 \) and the word given by \( b(sw, i) \).

      Repeat this algorithm from the top with \( w_{new} = w' \) and \( d_{new} = D(w') \).

   (c) If the length of \( b \) is larger than one, repeat this algorithm with \( d_{new} = rest(d) \).

In words, once we find a pair of generators with the same index that we can use to decrease the length of a word, we again check all pairs of generators with the same index in the new word until it is not possible to use any pair to decrease the length. Then \( w \) is in a reduced form.

### 7.1.2 Calculating all words in \( T(n_1, ..., n_i) \)

This algorithm lies at the very core of calculating \( \tilde{Q}_k \). It is relatively straightforward to express the conserved quantities in terms of the \( T(n_1...n_i) \), see equations 2.5. Calculating every word in such a term is a lot more difficult. As an example, let us look at the construction of all words contained in \( T(3, 2) = (d^3/db^3)T(b)T^{-1}(b)(d^2/db^2)T(b)T^{-1}(b) \) in \( \tilde{Q}_5 \). First note that we have:

\[
\left( \frac{d^3}{db^3}T(b) \right)^{T^{-1}(b)} \bigg|_{b=0} = \sum_{i=1}^{L} \tilde{e}_i \tilde{e}_{i+1} \tilde{e}_{i+2} + \sum_{i=1}^{L-2} \sum_{j=i+3}^{L-2} \tilde{e}_i \tilde{e}_{i+1} \tilde{e}_j + \sum_{i=1}^{L-4} \sum_{j=i+3}^{L-4} \sum_{m=j+3}^{L} \tilde{e}_i \tilde{e}_j \tilde{e}_m \quad (7.15a)
\]

\[
= e_1 e_2 e_3 + \sum_{i=1}^{L} \sum_{j=i+3}^{L-2} \tilde{e}_i \tilde{e}_{i+1} \tilde{e}_j + \sum_{i=1}^{L-4} \sum_{j=i+3}^{L-4} \sum_{m=j+3}^{L} \tilde{e}_i \tilde{e}_j \tilde{e}_m \quad (7.15b)
\]

This already contains a lot of different terms. Fortunately, we can throw away a majority of these terms. To see this, note that we are calculating a term in \( \tilde{Q}_k = 5 \). As such, every word must be constructed by a total of 5 generators in different places. Furthermore, from Theorem 1.1, we know that every word is constructed locally, i.e. any word must be constructed in such a way that every \( e_i \) in the range \([i_1, i_b]\) appeared at least once before reducing the word. The words in the equation above all contain 3 generators, thus we can only use 2 more generators to add onto the words that are formed (also denoted by the fact that we are multiplying this term by a term \((d/db)^2T(b)T^{-1}(b) \) in which every term consists of exactly two generators). Therefore, all words that have a ‘gap’ of more than two missing generators in equation 7.15 can be thrown away. We are left with the following terms:

\[
\left( \frac{d^3}{db^3}T(b) \right)^{T^{-1}(b)} \bigg|_{b=0} = e_1 e_2 e_3 + e_1 e_2 e_4 + e_1 e_2 e_5 + e_1 e_3 e_4 + e_1 e_4 e_5 + e_1 e_2 e_5 + e_1 e_3 e_5 \quad (7.16)
\]

Where now all double and triple sums are replaced by single sums that are taken implicitly. The second term is given by:

\[^{15}\text{Note that if we contained a different pair of double generators } e_{i_2} \text{ that was already checked using this algorithm and could not be used to reduce the word, they will again be checked when the algorithm is recurred.}\]
\[
\left( \frac{d^2}{db^2} T(b) \right) T^{-1}(b) \bigg|_{b=0} = e_1 e_2 + \sum_{i=1}^{L-2} \sum_{j=1}^{L} \hat{e}_i \hat{e}_j = e_1 e_2 + e_1 e_3 + e_1 e_4 + \ldots = \sum_{a=1}^{4} e_i e_{i+a} \quad (7.17)
\]

Where in the last equality, all terms with more holes that 3 are tossed away, since they always violate the locality Theorem. This equality is thus only valid in these specific terms, not in general. Let us no gaps, thus as long as we satisfy the locality theorem we can place the second term keeping only the terms that are constructed locally, we get:

\[
e_1 e_2 e_3 \cdot e_1 e_2 = e_1 e_2 e_3 e_1 e_2 + e_1 e_2 e_3 e_2 e_3 + e_1 e_2 e_3 e_3 e_4 + e_1 e_2 e_3 e_4 e_5 \quad (7.18a)
\]

\[
e = e_3 e_4 e_5 e_2 e_2 + e_2 e_3 e_4 e_2 e_2 + e_1 e_2 e_3 + e_1 e_2 e_3 + ne_1 e_2 e_3 e_4 + e_1 e_2 e_3 e_4 e_5 \quad (7.18b)
\]

Where we translated the words such that the lowest index is always 1 (since we have periodic boundary conditions, \(e_1, e_2, \ldots e_m = e_{1+n}, e_{2+n}, \ldots e_{m+n}\) for an arbitrary integer constant \(n\)) and put the words in a reduced form. Since the word coming from equation \(7.16\) did not contain a gap, we just connect \(e_1 e_2\) on all positions to the already existing term. Now multiplying the second term in equation \(7.16\) by the same term, we get:

\[
e_1 e_2 e_4 \cdot e_1 e_2 = e_1 e_2 e_4 e_2 e_3 + e_1 e_2 e_4 e_3 e_4 \quad (7.19a)
\]

\[
e = ne_1 e_2 e_4 e_3 + e_1 e_2 e_4 \quad (7.19b)
\]

Since the first word \(e_1 e_2 e_4\) contains a hole (it misses generator \(e_3\), defined in Definition 8), we can only add all terms of the second word such that it contains an \(e_3\). Thus, we keep only two terms. Outlining the complete algorithm:

**Algorithm 1.**

1. Start with the first term \((\frac{d}{db})^{n_1} T(b) T^{-1}(b)\). Generate all words such that the total number of holes never exceeds \((k - n_1)\), the number of generators that are to be placed afterwards in subsequent terms.

2. Move to the next term \((\frac{d}{db})^{n_2} T(b) T^{-1}(b)\). For each word generated in the previous step, check how many holes must be filled. This number is given by \(f = (\# \text{ holes} - (k - n_1 - n_2))\), i.e. the number of holes minus the number of generators that will still be placed after this step. If this number is negative, it means we construct new words while allowing to create \(f\) new holes. If \(f = 0\), it is not necessary to place the words in this term such that they cover a hole (but it is an allowed operation) as long as no new holes are created. If this number is positive, it restricts the placement of new words such that at least \(f\) words are covered. Note that if this is not possible for two words we are multiplying, we cancel the whole term.

3. Repeat the previous step for each term \((\frac{d}{db})^{n_i} T(b) T^{-1}(b)\) defining \(f_i = (\# \text{ holes} - (k - \sum_{j=1}^{i} n_i))\) at each step. At many occasions, it will be possible to place a new word on several positions, in which case each possible position should be occupied once and subsequent terms should be attempted to be placed on each of the resulting words.
Using this algorithm, we finish the example calculation:\footnote{Note that the actual computer algorithm computed a list of possible positions that tracked where a new word could be placed such that the minimum number of holes necessary were covered. Hence the zeroes in these equations were not hard to compute, they just produced an empty list of outcomes.}

\[
\begin{align*}
e_1e_2e_3 \cdot e_1e_2 &= e_1e_2e_3e_3e_4 \\
e_1e_3e_4 \cdot e_1e_2 &= e_1e_3e_4e_1e_2 + e_1e_3e_4e_2e_3 = ne_1e_3e_2e_4 + e_1e_3e_4e_2e_3 \\
e_1e_4e_5 \cdot e_1e_2 &= e_1e_4e_5e_2e_3 \\
e_1e_2e_5 \cdot e_1e_2 &= e_1e_2e_5e_3e_4 \\
\end{align*}
\]

\[e_1e_2 \text{ cannot cover two holes}\]

\[
\begin{align*}
e_1e_2e_3 \cdot e_1e_3 &= e_1e_2e_3e_1L_2 + e_1e_2e_3e_1e_3 + e_1e_2e_3e_2e_4 = e_2e_1e_3 + ne_1e_3 + e_1e_2e_4 \\
e_1e_2e_4 \cdot e_1e_3 &= e_1e_2e_4e_1e_3 + e_1e_2e_4e_3e_5 = e_1e_4e_3 + e_1e_2e_4e_5 \\
e_1e_2e_5 \cdot e_1e_3 &= 0 \\
e_1e_3e_4 \cdot e_1e_3 &= e_1e_3e_4e_1e_2 + e_1e_3e_4e_2e_4 = e_2e_4e_5e_1e_3 + ne_1e_3e_2e_4 \\
e_1e_4e_5 \cdot e_1e_3 &= 0 \\
e_1e_2e_5 \cdot e_1e_3 &= 0 \\
e_1e_3e_5 \cdot e_1e_3 &= e_1e_3e_5e_2e_4 \\
e_1e_2e_3 \cdot e_1e_4 &= e_1e_2e_3e_1e_3 + e_1e_2e_3e_1e_4 = ne_2e_1e_3e_4 + e_1e_3e_4 \\
e_1e_2e_4 \cdot e_1e_4 &= e_1e_2e_4e_1e_3 = e_1e_2e_5e_1e_4 \\
e_1e_2e_5 \cdot e_1e_4 &= 0 \\
e_1e_3e_4 \cdot e_1e_4 &= e_1e_3e_4e_2e_5 \\
e_1e_4e_5 \cdot e_1e_4 &= 0 \\
e_1e_2e_5 \cdot e_1e_4 &= 0 \\
e_1e_3e_5 \cdot e_1e_4 &= 0 \\
e_1e_2e_3 \cdot e_1e_5 &= e_1e_2e_3e_1e_4 = e_2e_3e_4e_1e_5 \\
e_1e_2e_4 \cdot e_1e_5 &= 0 \\
e_1e_2e_5 \cdot e_1e_5 &= 0 \\
e_1e_3e_4 \cdot e_1e_5 &= 0 \\
e_1e_4e_5 \cdot e_1e_5 &= 0 \\
e_1e_2e_5 \cdot e_1e_5 &= 0 \\
e_1e_3e_5 \cdot e_1e_5 &= 0
\]
7.1 Algorithms

7.1.3 Diagrammatic representation of $TL_a$

In this thesis, two representations for operators in $TL_a$ have been used. This section deals with the hash table representation, where a word is defined by how it affects a configuration on the of the $O(n)$ loop model on the cylinder. Remember that acting with an operator on a configuration of the cylinder means adding a layer to the top of the cylinder (in the $O(n = 1)$ loop model this is a mapping from one link pattern to a new one). Thus, we can label the points on the top of the cylinder and the the points that they will be mapped to by the operator $^{17}$.

Then, we can define an operator as a hash table linking the points on top to the points on the bottom. E.g. for a monoid, we get:

\[ \begin{array}{c}
\text{top} & \text{bottom} \\
1 & 2 \\
3 & 4 \\
\end{array} \]

Then we can construct general operators in $TL_a$ by chaining together monoids, following strands all the way from the top to the bottom and storing the resulting links into one hash. For example, the word $e_2e_3$ for $L = 4$ becomes:

\[ \begin{array}{c}
\text{top} & \text{bottom} \\
1 & 2 \\
3 & 4 \\
\end{array} \]

7.1.4 Equivalent forms of words

In order to add all results from calculations with words, e.g. when calculating $Q_k$ or a commutator $[e_i, w]$, the results have to be added up in such a way that equivalent forms of words are recognized and added up to the same form. For this thesis, both the diagrammatic representation described above and the algebra itself have been used to perform calculations. In the diagrammatic representation, identifying equivalent forms of words is trivial, as they will automatically be mapped to the same diagram. All information contained in a diagram is how each strand connects two points. The path that any strand follows is not saved, thus we automatically end up with the same diagram for equivalent words.

$^{17}$To distinguish the points on the bottom and on top, the symbols ‘s’ for ‘south’ and ‘n’ for north were used. One could also use one single index $i$ where if $1 \leq i \leq L$ the index denotes the bottom and if $L < i \leq 2L$ the index $i - L$ denotes the top.
For the algebra itself, a database was created where each equivalent reduced form of a word was linked to one arbitrary version. When given an arbitrary form of a reduced word \( w \), all its equivalent reduced forms can be generated using the following algorithm:

**Algorithm 2.**

1. Let \( L_{\text{old}} = L_{\text{new}} = (w) \) be a list containing only \( w' \). Let \( i = 0 \).
2. Take the first element in the list \( L_{\text{old}} \). Check if the generator on position \( i \) can be freely commuted to the right. If so, add the new word that is created by this operation to the new list \( L_{\text{new}} \). Subsequently check if the generator that has now been shifted to position \( i + 1 \) can again be commuted to the right, and so on until it can not be commuted to the right anymore. Then continue with other elements in \( L_{\text{old}} \) starting at the same index \( i \) until all elements in \( L_{\text{old}} \) have been visited once.
3. Increase \( i \) by one and set \( L_{\text{old}} = L_{\text{new}} \) and repeat this process until \( i > \text{length}(w) \).

After this process \( L_{\text{new}} \) is a list with all equivalent reduced forms of \( w \).

### 7.1.5 Generating all unique words in \( \mathcal{TL}_a \)

To check the conjectures in section 2.1, \( \tilde{Q}_k \) was generated for \( 10 < k < 18 \). Based on conjecture 2.2, words that contain one \( e_i \) twice in a reduced form will not be generated. Thus, the goal is to produce every unique word of length \( 0 < l \leq k \) excluding words with double generators but including words with holes.

For \( \tilde{Q}_1 \), the problem is trivial as \( e_1 \) is the only reduced word possible. For higher \( k \), we start by looking at connected words. We note that they can be translated to a list with ones and zeroes (henceforth named instruction) in the following way:

**Algorithm 3.** Translate a connected word into an instruction:

1. Let \( w \) be the word we want to translate and let \( I(w) \) be the corresponding instruction. Set \( I(w) \) to be empty initially. Let \( i = 2 \).
2. If \( e_i \) is to the left of \( e_{i-1} \) in \( w \), append a 0 to the end of \( I(w) \). Otherwise, add a 1.
3. Set \( i_{\text{new}} = i_{\text{old}} + 1 \) and repeat step 2 and 3 until \( i_{\text{new}} > l \) where \( l \) is the length of the word.

For higher \( k \), we define the following mapping from a word to a list with ones and zeroes (henceforth called an instruction \( I(w) \)):

**Algorithm 4.** Translate a word into an instruction:

1. Let \( w \) be the word we want to translate and let \( I(w) \) be the corresponding instruction. Set \( I(w) \) to be empty initially. Let \( i = 2 \).
2. Check if \( e_i \) is present in the word \( w \). If not, append a 0 to the end of \( I(w) \). If \( e_i \) is to the left of \( e_{i-1} \) in \( w \), append a −1 to the end of \( I(w) \). Otherwise, append a 1.
3. Set \( i_{\text{new}} = i_{\text{old}} + 1 \) and repeat step 2 and 3 until \( i_{\text{new}} \) is larger than the index of the highest generator in \( w \).

This provides a unique instruction for each nonequivalent word: for two equivalent but different reduced versions of a word, only the relative ordering between \( e_i \) and \( e_{i \pm 1} \) s.t. \( |i - j| > 1 \) can differ. In other words, if two reduced words differ by an ordering of a \( e_i \) and \( e_{i \pm 1} \), those words are nonequivalent and will be mapped to a different instruction. Furthermore, this process is reversible. We can use the instruction \( I(w) \) to construct a reduced word equivalent to our initial word \( w \) in the following way:

---

18 Usually the first form of a word encountered in a calculation that was not yet in the database was chosen as the canonical form.
Algorithm 5. Translate an instruction to a word:

1. Let $w' = e_1$. Let $i = 2$.

2. Let $j$ be the first entry of $I(w)$. If $j = 0$, set $w'_{\text{new}} = w'_{\text{old}}$. If $j = 1$, set $w'_{\text{new}} = w'_{\text{old}} \cdot e_i$. Set $w'_{\text{new}} = e_i \cdot w'_{\text{old}}$ otherwise.

3. Set $i_{\text{new}} = i_{\text{old}} + 1$ and $I(w)_{\text{new}} = \text{rest}(I(w)_{\text{old}})$ and repeat step 2 and 3 until $I(w)$ is empty. $w'$ is now equivalent to $w$.

We can now construct all nonequivalent words of a certain length $l$ by generating all instructions with the two restrictions:

- The total number of $\pm 1$ in $I(w)$ must be exactly $l$
- The last element of $I(w)$ cannot be a 0.
References


