Generalized Belief Propagation Algorithms for Decoding of Surface Codes

Josias Old[1,2] and Manuel Rispler[1,2,3]

[1]Institute for Quantum Information, RWTH Aachen University, Aachen, Germany
[2]Institute for Theoretical Nanoelectronics (PGI-2), Forschungszentrum Jülich, Jülich, Germany
[3]QuTech, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands

Belief propagation (BP) is well-known as a low complexity decoding algorithm with a strong performance for important classes of quantum error correcting codes, e.g. notably for the quantum low-density parity check (LDPC) code class of random expander codes. However, it is also well-known that the performance of BP breaks down when facing topological codes such as the surface code, where naive BP fails entirely to reach a below-threshold regime, i.e. the regime where error correction becomes useful. Previous works have shown, that this can be remedied by resorting to post-processing decoders outside the framework of BP. In this work, we present a generalized belief propagation method with an outer re-initialization loop that successfully decodes surface codes, i.e. opposed to naive BP it recovers the sub-threshold regime known from decoders tailored to the surface code and from statistical-mechanical mappings. We report a threshold of 17% under independent bit-and phase-flip data noise (to be compared to the ideal threshold of 20.6%) and a threshold value of 14% under depolarizing data noise (compared to the ideal threshold of 18.9%), which are on par with thresholds achieved by non-BP post-processing methods.

1 Introduction

Quantum computing devices suffer from operational errors and decoherence. Methods to keep errors in check and advance towards fault-tolerant quantum computing involve *quantum error correcting codes*. A code is formally defined as a k-dimensional subspace of some n-dimensional Hilbert space. In quantum error correcting codes called *(quantum) stabilizer codes*, the error correction procedure consists of three main elements: the (non-destructive) measurement of stabilizer operators, the decoding of this measurement to obtain a suitable recovery operation, and the application of the latter to correct the errors on the codestate. In order to achieve *fault-tolerant quantum computation*, all three stages have to be done in a fast and efficient way to avoid the accumulation of errors.

A class of codes which recently received a lot of attention are *Quantum Low-Density Parity-Check codes* (qLDPC codes) [1]. Given an asymptotically constant rate $r = \frac{k}{n}$, they are proven to achieve fault-tolerance with only constant overhead. That is, the ratio of total number of qubits used in the fault-tolerant protocol to the number of qubits in a non-fault tolerant circuit is asymptotically constant for increasing code size [2]. An important quantity for error correction is the *minimum distance*. It is the minimum weight (*i.e.* the number of qubits involved) of a logical operation on the codespace. Naturally, a large distance implies a better protection against errors. Quantum error correcting codes are called *good* if their rate is constant and the minimum distance scales linearly with increasing code size. The existence of such codes, however non-qLDPC, has been proven for a long time [3, 4]. In a recent seminal work, Panteleev and Kalachev showed that it is actually possible to construct asymptotically good qLDPC codes [5]. Shortly after that breakthrough, several other constructions achieved a similar scaling behavior [6, 7].

A key ingredient of a fault-tolerant protocol, which will be the focus of this work, is an efficient

Josias Old: j.old@fz-juelich.de
Manuel Rispler: rispler@physik.rwth-aachen.de

decoding algorithm. Proofs of fault-tolerance often consider the classical processing of the error correction cycle as free. In practice, this is not the case and decoders should be fast enough to prevent additional errors from occurring. This typically results in a trade-off between decoder accuracy, i.e. how well the decoding algorithm finds a good, ideally the most likely correction and decoder computational complexity, i.e. how fast the algorithm can be executed as a function of the input size. There are a variety of decoders that are highly adapted to the quantum code in use. In this paper, we consider a generalization of a classical decoding algorithm called *belief propagation* (BP) or *sum-product algorithm*. For classical LDPC codes, it is among the best practical decoding algorithms: it allows to achieve error correction close the theoretical upper bound on the information transfer, the *Shannon capacity* [8], while remarkably maintaining low algorithmic complexity. In principle, any classical decoding algorithm can be used to decode quantum codes, which stems from the fact that any quantum code can be formulated as a so-called CSS (Calderbank-Shor-Steane) code, which decouples into two classical codes that can (in principle) be decoded independently. However, this has many pitfalls. The most relevant one for us here is the observation that belief propagation on the surface code can completely fail to converge or converge to decoding decisions that do not remove the error. This is thought to be the underlying reason for the observation that the surface code produces no sub-threshold regime (the regime of error rate, where the logical error rate can be arbitrarily suppressed by increasing the code size) when decoding with BP [9]. This is in stark contrast to tailored quantum decoders such as minimum weight matching, under which the surface code is typically celebrated for its high threshold value [10]. Recently introduced post-processing methods to BP showed that these can in principle decode various types of qLDPC codes [11, 12]. We will focus on a single decoder that achieves the same at a lower complexity. The Generalized Belief Propagation (GBP) algorithm was previously considered in the context of decoding quantum bicycle codes [13].

This paper is structured as follows. First, we review the basics of stabilizer error correction and quantum LDPC codes. In Section 2 we show the relation of GBP to standard BP and adapt it for the decoding of quantum codes. Section 3 applies GBP to surface codes and shows numerical evidence of the emergence of a threshold.

1.1 Stabilizer Codes

Stabilizer codes are defined by an Abelian subgroup of the Pauli group, $\mathcal{S} \subseteq \mathcal{P}_n$ with $-\mathbf{1} \notin \mathcal{S}$. The (commuting) elements of the group are the *stabilizers* or *parity checks* $S \in \mathcal{S}$. The codespace $\mathcal{Q}$ is then defined as the subspace of the Hilbert space $\mathcal{H}^n$ that is stabilized by $\mathcal{S}$. For any codestate $|\psi\rangle$ it therefore holds that $S|\psi\rangle = |\psi\rangle \ \forall S \in \mathcal{S}$. If there are $n-k$ independent generators for $\mathcal{S}$, $|\langle \mathcal{S} \rangle| = n-k$, then the codespace is k-dimensional and encodes k qubits.

Consider a Pauli error $E \in \mathcal{P}_n$ occurring on a codestate, $|\psi\rangle \to E|\psi\rangle$. Such an error can be detected by measuring all stabilizer generators, if any of those anticommute with the error, $SE|\psi\rangle = -ES|\psi\rangle$ for some $S \in \mathcal{S}$. The binary outcome of all stabilizer measurements is also called the *syndrome* or *syndrome vector* $\mathbf{s} \in \mathrm{GF}(2)^{n-k}$ with

$$s_c = \langle E, S_c \rangle := \begin{cases} 0 & \text{if} \quad [E, S_c] = 0, \\ 1 & \text{if} \quad \{E, S_c\} = 0. \end{cases} \tag{1}$$

Here, s_c denotes the measurement outcome of stabilizer generator S_c for $c = 1, \ldots, n_c = n-k$. This gives rise to three different scenarios. We say there occurred a

1. *detectable error* if $\mathbf{s} \neq 0$,
2. *trivial error* if $\mathbf{s} = 0$ and $E \in \mathcal{S}$,
3. *logical error* if $\mathbf{s} = 0$ and $E \notin \mathcal{S}$.

The last case represents the operators that map non-trivially between codestates, the *logical operators*. They can formally be defined using the *centralizer* of $\mathcal{S}$ in $\mathcal{P}_n$, $\mathcal{C}_{\mathcal{P}}(\mathcal{S}) = \{L : LS = SL \ \forall S \in \mathcal{S}\}$, such that the logical operators are $\mathcal{L} = \mathcal{C}_{\mathcal{P}}(\mathcal{S}) \setminus \mathcal{S}$. The *(minimum) distance* of the stabilizer quantum code then corresponds to the minimal weight of a logical operator,

$$d = \min_{L \in \mathcal{L}} |L|. \tag{2}$$

1.2 Decoding of Stabilizer Codes

The *decoding problem* refers to the inference of a suitable correction from the measured syndrome. Because trivial errors have zero syndrome, they define an equivalence class for every detectable error. This feature called *degeneracy* implies that corrections only need to be found up to a trivial error. Due to the linearity of the codes, errors up to weight $t = \lfloor \frac{d-1}{2} \rfloor$ can be uniquely matched to a codestate and hence be corrected for. A simple decoder involves a lookup-table which stores a suitable correction, for example the lowest-weight error matching each measured syndrome. Making assumptions on the error probabilities can improve this approach.

To that end, we consider Pauli error channels,

$$\mathcal{E}(\rho) = \sum_{E \in \mathcal{P}_n} p(E) E \rho E^\dagger. \tag{3}$$

Furthermore, we assume that the qubits are memoryless and suffer from errors independently,

$$p(E) = \prod_{q=1}^{n_q} p(E_q). \tag{4}$$

We can write the probability of errors conditioned on the observation of a syndrome using Bayes' rule and the fixed "evidence" $p(\mathbf{s}) = 1$ as

$$p(E|\mathbf{s}) = p(E)p(\mathbf{s}|E) \tag{5}$$

$$= \prod_{q=1}^{n_q} p(E_q) \prod_{c=1}^{n_c} \delta(\langle E, S_c \rangle = s_c). \tag{6}$$

By $\delta(i = j)$ we denote the Kronecker delta δ_{ij}. It assigns zero probability to all error configurations E that have a commutation relation inconsistent with the measurements.

In quantum error correction, the ideal decoder returns an error guess, which is in the most likely error class, specified by all errors that are equivalent up to an element of the stabilizer group. Given a syndrome, this *maximum likelihood decoding* identifies

$$E^\star \in \arg\max_{E\mathcal{S}} p(E\mathcal{S}|\mathbf{s}) = \arg\max_{E\mathcal{S}} \sum_{S \in \mathcal{S}} p(ES|\mathbf{s}). \tag{7}$$

Note, however, that directly calculating the probabilities for an error class (or even just storing the lookup-table) quickly becomes intractable since there are $2^{(n-k)}$ different syndromes. For example, storing all syndromes of a code defined by 42 independent stabilizer generators requires approximately 550GB of memory.

Possibly more efficient decoding strategies rely on relaxed constraints such as finding the

- most likely error, *i.e.* identifying

$$E^\star = \arg\max_{E \in \mathcal{P}_n} p(E|\mathbf{s}) \tag{8}$$

or the

- qubit-wise most likely error, *i.e.* identifying

$$E^\star = \{\arg\max_{E_q \in \mathcal{P}_1} p_q(E_q|\mathbf{s})\}_{q=1}^{n_q}, \tag{9}$$

where $p_q(E_q|\mathbf{s}) = \sum_{q' \neq q} p(E|\mathbf{s})$ are the single-qubit marginal probabilities.

Note that these equations are agnostic of the quantum nature of the underlying problem and have been studied extensively in various settings including classical decoding. Finding the most likely error is already less involved than finding the most likely error class, but is in general still NP-complete [14]. Calculating $p(E|\mathbf{s})$ directly and even inferring the marginal probabilities $p_q(E_q|\mathbf{s})$ still involves an exponential number of components. However, there exist algorithms that can - under certain conditions - calculate the marginals in linear complexity $\mathcal{O}(n)$. This comes at the cost that the qubit-wise most likely error might globally be inconsistent with the observed syndrome. Before introducing such an algorithm, the *belief propagation* algorithm, we fix the notation and graphical representations used throughout this paper.

1.3 Representation of Stabilizer Codes

Algebraic representation The stabilizer group and most operations used in stabilizer error correction can be mapped from the Pauli group to vector spaces over finite fields (or *Galois Fields*), denoted by $\mathrm{GF}(q)$ with $q = \{2, 4\}$ [15].

In both cases, the Pauli word E is mapped to a vector $\mathbf{e} \in \mathrm{GF}(q)$ of length $(3 - \frac{q}{2})n$. The group operation is mapped to element-wise addition on the finite fields, $EE' \mapsto \mathbf{e} + \mathbf{e}'$. Commutation of two Paulis E, E' can be checked using the *symplectic product* denoted by $\star$,

$$\langle E, E' \rangle \mapsto \mathbf{e} \star \mathbf{e}'. \tag{10}$$

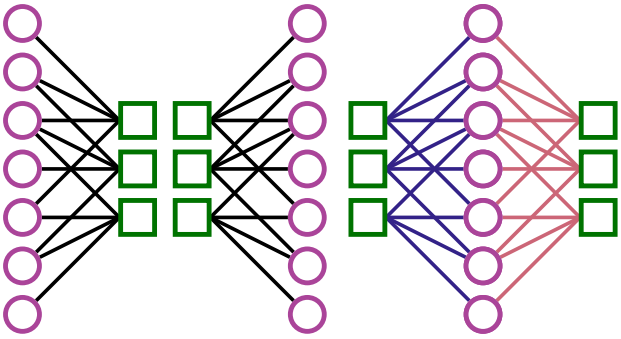

Figure 1: Tanner graph representation of the Steane code in binary (left) and quaternary (right) representation. Purple circles correspond to qubits, green squares to parity checks. In the quaternary representation, blue edges correspond to X-type checks and red edges to Z-type checks. Note that since the Steane code is a CSS-code, the binary Tanner graph splits into two disjoint graphs $\mathcal{T} = \mathcal{T}_X \sqcup \mathcal{T}_Z$.

The stabilizer generators are put in a *parity check matrix* $\mathbf{H} \in \mathrm{GF}(2)^{(n-k)\times 2n}$ or $\mathbf{H} \in \mathrm{GF}(4)^{(n-k)\times n}$, such that the measurement of all stabilizers can be represented by the symplectic matrix-vector product

$$\mathbf{H} \star \mathbf{e} =: \mathbf{s}. \tag{11}$$

The actual implementation in the binary and quaternary framework can be found in App. B.

Tanner graph representation Stabilizer codes can be graphically represented as *Tanner graphs*, similar to classical codes [16]. These are bipartite graphs with two vertex sets Q and C representing the qubits and the stabilizer measurements/ parity checks respectively. The edge set E consists of edges $e = (q, c)$ drawn between vertices $q \in Q$ and $c \in C$ if qubit q is involved in the parity-measurement of stabilizer c. Different types of Paulis can be distinguished by coloring the edges. With this correspondence, the parity-check matrix $\mathbf{H}$ is the *biadjacency matrix* or *reduced adjacency matrix* of the Tanner graph $\mathcal{T} = (Q \cup C, E)$.

Example: The Steane Code The Steane code [17] is a $[[7, 1, 3]]$- quantum code with stabilizer generators

$$\langle \mathcal{S} \rangle = \{X_0X_1X_2X_4, X_1X_2X_3X_5, X_2X_4X_5X_6, \tag{12}$$

$$Z_0Z_1Z_2Z_4, Z_1Z_2Z_3Z_5, Z_2Z_4Z_5Z_6\}. \tag{13}$$

In the algebraic representations, the parity check matrices are

$$\mathbf{H}_{\mathrm{GF}(2)} = \begin{bmatrix} \mathbf{H}_X & 0 \\ 0 & \mathbf{H}_Z \end{bmatrix} = \begin{bmatrix} 1\,1\,1\,0\,1\,0\,0 & & \\ 0\,1\,1\,1\,0\,1\,0 & & \\ 0\,0\,1\,0\,1\,1\,1 & & \\ & & 1\,1\,1\,0\,1\,0\,0 \\ & & 0\,1\,1\,1\,0\,1\,0 \\ & & 0\,0\,1\,0\,1\,1\,1 \end{bmatrix} \tag{14}$$

and

$$\mathbf{H}_{\mathrm{GF}(4)} = \begin{bmatrix} \mathbf{H}_X \\ \omega\mathbf{H}_Z \end{bmatrix} = \begin{bmatrix} 1\,1\,1\,0\,1\,0\,0 \\ 0\,1\,1\,1\,0\,1\,0 \\ 0\,0\,1\,0\,1\,1\,1 \\ \omega\,\omega\,\omega\,0\,\omega\,0\,0 \\ 0\,\omega\,\omega\,\omega\,0\,\omega\,0 \\ 0\,0\,\omega\,0\,\omega\,\omega\,\omega \end{bmatrix} \tag{15}$$

and Tanner graphs are shown in Fig. 1.

1.4 Low-Density Parity-Check Codes

Low-Density Parity-Check (LDPC) codes are families of classical codes with a sparse parity-check matrix. The most successful classical LDPC codes rely on random or pseudo-random constructions of the parity-check matrix, most notably Sipser and Spielman's *Expander Codes* [18]. Their properties include a constant rate, a linear distance and efficient decoders, which is often referred to as *good code* [19]. This has led to try and construct quantum versions of LDPC codes (qLDPC codes) in the hope of obtaining *good* quantum codes with a constant rate and distance linear in the number of qubits. In addition to having good decoding properties, it was famously shown by Gottesman that such codes, if they exist, enable fault-tolerant quantum computation with only constant overhead [2].

In general, qLDPC codes can be defined similarly to classical LDPC codes as codes with a sparse parity check matrix. This corresponds to quantum stabilizer codes with stabilizer generators of low weight that is upper bounded by a constant. In particular, a (d_c, d_q)-qLDPC code ensemble has parity checks measuring at most d_c qubits and every qubit is involved in at most d_q syndrome measurements. This broad definition includes a range of well known codes like the surface codes. They are defined on a lattice, therefore exhibit a high degree of symmetry and have only nearest-neighbor interaction [20]. They have a minimum distance $d \propto \sqrt{n}$ but suffer from a vanishing rate $r \to 0$ as the number of qubits $n \to \infty$. A more general construction, the hypergraph product (HGP) codes, can achieve a constant rate $r \to 1 - \frac{d_c}{d_q}$, when based on good (e.g. random) classical codes [21].

Very recently, a construction that builds on a *G-lifted product* of expander codes over non-abelian groups G by Panteleev and Kalachev were proven to achieve constant rate and linear distance [5]. Similar constructions like the *Quantum Tanner Codes* or codes from balanced product of lossless expanders achieve the same [6, 7].

The advantageous properties of good qLDPC codes manifest at large qubit numbers. For near future applications, a moderate number of qubits in the order of a few hundred is realistic. It is therefore reasonable to focus on the less intricate hypergraph product construction, which we will briefly recap in the following.

Hypergraph Product Codes The hypergraph product construction uses graph based arguments to derive quantum codes from classical codes. For details refer to [21], in the following considerations the construction rule for the parity check matrices shall be sufficient.

Let $\mathbf{H}_1 \in \mathrm{GF}(2)^{m_1 \times n_1}, \mathbf{H}_2 \in \mathrm{GF}(2)^{m_2 \times n_2}$ be parity check matrices of classical codes $\mathcal{C}_1, \mathcal{C}_2$ with dimension k_1 and k_2. The Hypergraph Product (quantum) Code $\mathcal{Q} = \mathrm{HGP}(\mathcal{C}_1, \mathcal{C}_2)$ with parity check matrix $\mathbf{H}$ is a quantum CSS code with parameters

$$[[n_1 n_2 + (n_1 - k_1)(n_2 - k_2), k_1 k_2, \min(d_1, d_2)]] \tag{16}$$

which has its parity check matrix constructed from the classical parity check matrices as

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_X & 0 \\ 0 & \mathbf{H}_Z \end{pmatrix}, \tag{17}$$

$$\mathbf{H}_X = \begin{pmatrix} \mathbf{1}_{n_1} \otimes \mathbf{H}_2 & \mathbf{H}_1^T \otimes \mathbf{1}_{m_2} \end{pmatrix}, \tag{18}$$

$$\mathbf{H}_Z = \begin{pmatrix} \mathbf{H}_1 \otimes \mathbf{1}_{n_2} & \mathbf{1}_{m_1} \otimes \mathbf{H}_2^T \end{pmatrix}. \tag{19}$$

If we choose the base codes to have minimum distance linear in its length, the HGP construction gives quantum codes with minimum distance $\Omega(\sqrt{n})$. The construction trivially preserves sparsity and therefore translates a classical LDPC property to a quantum LDPC property. Some choices of base codes give well known quantum codes.

- Taking the classical (cyclic) *repetition codes* as base gives the topological *(toric) surface codes* [20]. They have vanishing rate and a distance $d \propto \sqrt{n}$. The graph based construction is shown in Fig. 2.

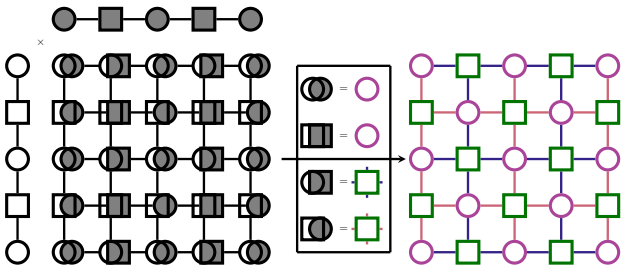

Figure 2: Graphical representation of the hypergraph product construction.The Cartesian graph product of the Tanner graphs of two repetition codes (left) yields the distance-3 surface code (right) using the rules shown in the middle and explained in app. C

- Taking the product of two (good) *classical expander codes* yields the *quantum expander codes* [22]. These codes have constant rate and a minimum distance $d \propto \sqrt{n}$. A slightly simplified version uses random classical codes that with, some known probability, have specific expansion properties. For such codes, a well known and widely used method for decoding is *belief propagation*.

2 Generalized Belief Propagation

We now introduce Generalized Belief Propagation due to Yedidia, Freeman and Weiss (YFM) [23]. We then show how a decoder for quantum codes can be constructed from that.

The Tanner graph introduced in Sec. 1.3 can also be thought of as an instance of a *factor graph* representing a joint probability distribution over *factors* f [24],

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c=1}^{n_c} f_c(\mathbf{x}_c). \tag{20}$$

In a physical system in thermal equilibrium, *Boltzmann's Law* gives the probability of a state,

$$p(\mathbf{x}) = \frac{1}{Z} e^{-\beta E(\mathbf{x})} \quad \text{with} \quad Z = \sum_{\mathbf{x} \in P} e^{-\beta E(\mathbf{x})}. \tag{21}$$

P is the space of all possible states $\mathbf{x}$ and β the inverse temperature which we set to 1 in the following. A connection between the two can be drawn by identifying the probability distributions and therefore defining an energy $E(\mathbf{x})$ of a state

$\mathbf{x}$ of the factor graph to be

$$E(\mathbf{x}) = -\sum_{c=1}^{n_c} \ln f_c(\mathbf{x}_c). \tag{22}$$

The ultimate goal in quantum decoding is to maximize the probability distribution of the errors, which can be seen to correspond to finding the minimum free energy configuration. While this generally will be computationally unfeasible, one can construct a tractable variational ansatz. In the following, we will present GBP appealing to a general intuition for variational methods and refer the reader to the appendix A for background on variational methods and a detailed derivation of GBP.

2.1 Derivation of the GBP Algorithm

Generalized Belief Propagation relies on region-based approximations to the free energy. These are a class of approximations to $F(b)$, where the approximate free energy is a function of beliefs over sets of variables, called *regions*. From now on, we will call the factor graph the *Tanner graph*, the factors *check nodes* and variables *qubit nodes*, to facilitate the transition to the decoder based on GBP.

We can define a region r of a Tanner graph as a set of qubit nodes $\mathcal{Q}_r$ and a set of check nodes $\mathcal{C}_r$ such that if a check node c is in $\mathcal{C}_r$, then all neighboring qubit nodes $\{\Gamma(c)\}$ are in $\mathcal{Q}_r$. The general idea is to define regions of the factor graph and then approximate the overall free energy with the sum of the free energies of all the regions, subject to the conditions ensuring validity which are shown in the following.

The thermodynamic quantities of a region are defined as

- region energy

$$E_r(\mathbf{x}_r) = -\sum_{c\in\mathcal{C}_r} \ln[f_c(\mathbf{x}_c)] - \sum_{q\in\mathcal{Q}_r} \ln[p(x_q)] \tag{23}$$

- region average energy and region entropy

$$U_r(b_r) = \sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) E_r(\mathbf{x}_r) \tag{24}$$

$$S_r(b_r) = -\sum_{\mathbf{x}_r} b_r(\mathbf{x}_r) \ln[b_r(\mathbf{x}_r)] \tag{25}$$

- region free energy

$$F_r(b_r) = U_r(b_r) - S_r(b_r). \tag{26}$$

When constructing regions, every check and qubit node should be contained in some region. Since check and qubit nodes might appear in multiple regions, it is necessary to introduce *counting numbers* c_r in order to ensure that every check and qubit is only counted once when summing over regions. They need to be chosen such that for a set of regions $\mathcal{R}$ of a Tanner graph

$$\sum_{r\in\mathcal{R}} c_r\ \delta(c \in \mathcal{C}_r) = 1\ \ \forall c, \quad \sum_{r\in\mathcal{R}} c_r\ \delta(q \in \mathcal{Q}_r) = 1\ \ \forall q. \tag{27}$$

The overall, region-based approximate thermodynamic quantities are then given by

- region-based average energy and region-based entropy

$$U_{\mathcal{R}}(\{b_r\}) = \sum_{r\in\mathcal{R}} c_r\ U_r(\{b_r\}), \tag{28}$$

$$S_{\mathcal{R}}(\{b_r\}) = \sum_{r\in\mathcal{R}} c_r\ S_r(\{b_r\}), \tag{29}$$

- region-based free energy

$$F_{\mathcal{R}}(\{b_r\}) = U_{\mathcal{R}}(\{b_r\}) - S_{\mathcal{R}}(\{b_r\}). \tag{30}$$

Note that not for every choice of regions, a valid set of counting numbers can be found. To understand that, consider a valid choice of regions $\nabla = \{r_i\}$ with the corresponding counting numbers $\{c_i\}$, such that for qubit q, Eq. 27 holds. Now adding any qubit q to a region r' with counting number $c_{r'}$ and $q \notin r'$ results in

$$\sum_{r\in\mathcal{R}} c_r\ \delta(q \in \mathcal{Q}_r) = 1 + c_{r'}, \tag{31}$$

which is only true iff $c_{r'} = 0$. Also note that different choices of regions can yield approximations of different quality [25].

2.1.1 Region graph

Similarly to the Tanner graph, the relations of different regions can be formalized in a graph theoretical framework. To that end, YFM introduce the *region graph* (RG). It is a labeled, directed graph $\mathcal{G} = (\mathcal{R}, \mathcal{E}, L)$ in which each vertex corresponds to a region $r \in \mathcal{R}$. A directed edge $e \in \mathcal{E}$ may exist from region r_p to r_c if $(\mathcal{Q}(r_c) \cup \mathcal{C}(r_c)) \subset (\mathcal{Q}(r_p) \cup \mathcal{C}(r_p))$, *i.e.* if the set of constituents of the *child* is a subset of the *parents* constituents. An example of a valid region graph for the Steane code is shown in Fig. 3.

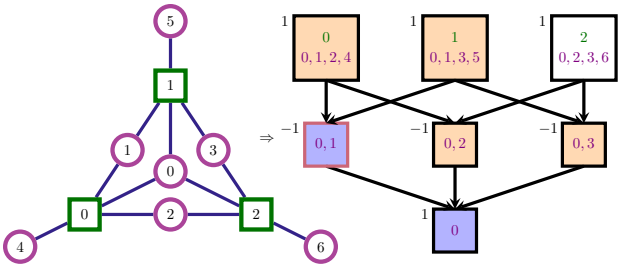

Figure 3: Example of a region graph for one part of the Steane code (corresponding to the classical Hamming code). Green and violet numbers represent checks and qubits respectively, counting numbers of the regions at top left corner. The "shadow" of the red bordered region containing qubits $0, 1$ is shaded in blue, the "blanket" in orange. See 2.1.2 for the definitions of shadow and blanket.

2.1.2 Notation

We adopt the notation from [26] (which differs slightly from YFM [23]) and adapt it to the error correction setting,

- $\mathcal{R}$: set of all vertices (=regions) of the region graph,
- $\mathcal{E}$: set of all (directed) edges of the region graph,
- $P(r), C(r), A(r), D(r)$: set of all parents, children, ancestors, descendants of region r,
- $\mathcal{Q}_r, \mathcal{C}_r$: qubits and checks in region r,
- $S(r) = D(r) \cup r$: "shadow" of the region r,
- $B(r) = P\left(S(r)\right) \setminus S(r)$: "blanket" of the region r.

2.1.3 Algorithm

We can recover the beliefs as approximations of the true marginal probabilities by minimizing the region-based free energy $F_{\mathcal{R}}(\{b_r\})$. To that end, a Lagrangian is constructed with the constraint that the beliefs shall be consistent between every parent and child region, *i.e.*

$$\forall r, p \in \mathcal{R}, r \subset p \implies \sum_{\mathbf{x}_p \setminus \mathbf{x}_r} \mathbf{b}_p(\mathbf{x}_p) = \mathbf{b}_r(\mathbf{x}_r), \tag{32}$$

and a normalization constraint

$$\forall r \in \mathcal{R}, \quad \sum_{\mathbf{x}_r} \mathbf{b}_r(\mathbf{x}_r) = 1. \tag{33}$$

Setting the derivatives of the Lagrangian with respect to the beliefs equal to zero gives implicit equations for the Lagrange multipliers and the beliefs, as shown in [27]. They can be solved iteratively and for that reason, the Lagrange multipliers (or functions thereof) are often called *messages* and the corresponding algorithms are referred to as *message-passing algorithms* [19].

In the following we show a more intuitive approach. For that, assume that a belief of a region first contains all *local* factors in that region. Messages from parent regions p into child regions r will be of the form $m_{p\to r}(\mathbf{x}_r)$. In order to catch all possible dependencies, we consider all messages that include some variables that are contained in that region. However, over-counting of factors should be prevented. This can be achieved by first considering all messages from parents into r. Secondly, we take into account all messages from regions which are not descendants of r but point into its descendants $D(r)$. This corresponds to the shadow and blanket of a region, such that an ansatz for the belief of a region r can be written as

$$\mathbf{b}_r(\mathbf{x}_r) \propto \prod_{q\in\mathcal{Q}_r} p_q(x_q) \prod_{c\in\mathcal{C}_r} f_c(\mathbf{x}_c) \prod_{\substack{a\in B(r),\\ b\in S(r)}} m_{a\to b}(\mathbf{x}_b). \tag{34}$$

By $\mathbf{x}_c$ we denote the variables in the support of check c. The message update rules follows from demanding consistency between parent and child regions (Eq. 32), giving the *Parent-to-Child-Algorithm*. In the following, the upper index (i) denotes the iteration step in order to formalize the iterative procedure. Note that with a uniform initialization of the messages, the beliefs of parent- and child regions p, r are incompatible at step $i = 0$. We therefore update the message from p to r by that mismatch,

$$\frac{\sum_{\mathbf{x}_p\setminus\mathbf{x}_r} \mathbf{b}_p^{(i)}(\mathbf{x}_p)}{\mathbf{b}_r^{(i)}(\mathbf{x}_r)} \xrightarrow{i\to\infty} 1 \tag{35}$$

$$\implies m_{p\to r}^{(i+1)}(\mathbf{x}_r) = m_{p\to r}^{(i)}(\mathbf{x}_r) \frac{\sum_{\mathbf{x}_p\setminus\mathbf{x}_r} \mathbf{b}_p^{(i)}(\mathbf{x}_p)}{\mathbf{b}_r^{(i)}(\mathbf{x}_r)}. \tag{36}$$

The overlap of messages in the numerator and denominator can then be canceled out to reduce

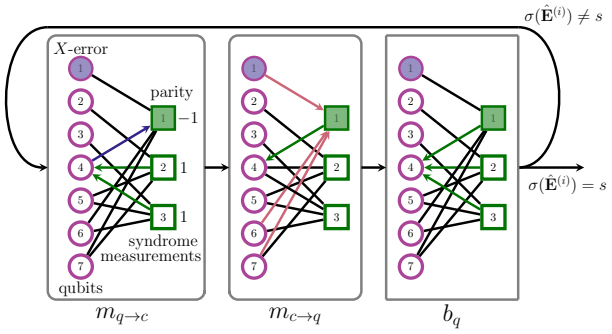

Figure 4: Belief Propagation decoding procedure. Shown is the X-Tanner graph of the $[[7,1,3]]$-Steane code. An error on the first qubit violates parity check 1. Messages are indicated as arrows. From left to right: qubit to check messages based on incoming messages, check to qubit messages, marginal probabilities or belief and hard decision.

the number of calculations. In that form, it becomes clear that consistent beliefs correspond to converged messages.

2.2 GBP as a Decoder for Quantum Codes

In order to draw the connection to quantum decoding, we identify the variables with the qubits. The factors, *i.e.* functional relations between qubits correspond to Kronecker delta tensors with entries according to the measured syndrome outcomes,

$$f_c(\mathbf{x}_c) = \delta[\mathbf{H}_c \star \mathbf{x}_c = s_c]. \tag{37}$$

The state of the system $\mathbf{x}$ coincides with the Pauli error E such that in the binary representation $\mathbf{x} = (x_1, x_2, \ldots, x_{2n_q}) \in \mathrm{GF}(2)^{2n}$ with $x_q \in \{0,1\}$. In the following, we denote the errors by $\mathbf{x}$ in order to avoid confusion with the energy.

Note that the algorithm introduced above includes both an implementation separating the X- and Z- part of a CSS quantum code (GF(2)) and a combined (GF(4))- implementation. The differences then lie in the length of the vectors, the set of messages used to calculate region beliefs and in the syndrome function. The latter uses implementations of the symplectic product Eq. 53 and Eq. 57 in the GF(2)- and GF(4)- framework respectively.

At each iteration, the parent-to-child algorithm defined by Eq. 36 gives an estimate of the marginal probability distribution of region r. These beliefs can be used to infer a guess of the error inflicted on the qubits.

2.2.1 How to make a hard decision

The argumentum maximum (argmax) of the belief of a region gives the most probable error on the region's qubits. This is often referred to as *making a hard decision*. In the end, a hard decision has to be taken for every qubit individually, whereas every qubit might be part of multiple regions. Consistency of hard decisions from different regions is only guaranteed if all messages are converged. This is in general not the case while iterating and also not guaranteed to happen at all. We therefore have to find a strategy to combine the different contributions from the region beliefs to the overall hard decision. A straightforward strategy to choose a hard decision is to focus on the highest level regions $\mathcal{R}_0 : \{r \in \mathcal{R} : c_r = 1\}$ and make a hard decision as

$$\hat{\mathbf{x}}^{(i)} = \bigcup_{r\in\mathcal{R}_0} \arg\max_{\mathbf{x}_r} \mathbf{b}_r^{(i)}(\mathbf{x}_r). \tag{38}$$

Naturally if the region-beliefs are not compatible, the overall error guess will not be consistent with the observed syndrome. In order to improve upon that, we find a more promising strategy. Whenever the error guesses from different regions on a single qubit are incompatible, we compare the beliefs of their respective regions and settle for the one with the largest belief. This corresponds to decoding as

$$\begin{aligned}\hat{\mathbf{x}}^{(i)} &= \{\hat{\mathbf{x}}_q^{(i)}\}_{q=1}^{n_q},\\ \hat{\mathbf{x}}_q^{(i)} &= \max_{x_q \in \mathbf{b}_r^{(i)}(\mathbf{x}_r)} \arg\max_{\mathbf{x}_r} \mathbf{b}_r^{(i)}(\mathbf{x}_r).\end{aligned} \tag{39}$$

In the following, we first show how standard Belief Propagation can be recovered from this more general approach and then show two different strategies to cope with further obstacles.

2.2.2 Bethe approximation

One choice of regions called *Bethe approximation* gives an approximation equivalent to the standard BP algorithm. It was originally introduced as *sum-product decoding* for (classical) LDPC codes by Gallager [28]. Later it was independently rediscovered by Pearl as a method to efficiently calculate single variable marginals on factor trees [29]. Poulin and Chung first applied BP to the decoding of quantum codes [30]. For an introduction on the BP algorithm, see for example [31, 12]. Here, we show how to get the standard BP equations from the generalized ansatz.

For this purpose, construct two types of regions, large and small. The large regions each contain a single check node and its neighboring qubit nodes. The small regions comprise single qubit nodes such that all qubit nodes that have more than one parent have their own small region. The counting numbers are $c_{r,\text{large}} = 1$ and $c_{r,\text{small}} = 1 - \sum_{q\in\mathcal{Q}_r} |P(q)|$. The beliefs of small regions $\{l_i\}$ and large regions $\{L_i\}$ are given by

$$\mathbf{b}_{c\in\{L_i\}}(\mathbf{x}_c) \propto \prod_{q\in\mathcal{Q}_c} p_q(x_q)\delta[\mathbf{H}_c \star \mathbf{x}_c = s_c] \prod_{\substack{a\in P[C(c)]\setminus c \\ b\in C(c)}} m_{a\to b}(x_b) \tag{40}$$

$$\mathbf{b}_{q\in\{l_i\}}(x_q) \propto p_q(x_q) \prod_{c\in P(q)} m_{c\to q}(x_q). \tag{41}$$

Using the consistency constraint Eq. 36 we find for the message updates

$$m^{(i+1)}_{c\to q}(x_q) = m^{(i)}_{c\to q}(x_q) \frac{\sum_{\mathbf{x}_c\setminus x_q} \prod_{q'\in\mathcal{Q}_c} p_{q'}(x_{q'})\delta[\sigma(\mathbf{x}_{\mathcal{Q}_c}) = s_c] \prod_{a\in P[C(c)]\setminus c, b\in C(c)} m^{(i)}_{a\to b}(x_b)}{p_q(x_q) \prod_{a\in P(q)} m^{(i)}_{a\to q}(x_q)} \tag{42}$$

$$= \frac{\sum_{\mathbf{x}_c\setminus x_q} \prod_{q'\in\mathcal{Q}_c\setminus q} p_{q'}(x_{q'})\delta[\sigma(\mathbf{x}_{\mathcal{Q}_c}) = s_c] \prod_{a\in P[C(c)]\setminus c, b\in C(c)} m^{(i)}_{a\to b}(x_b)}{\prod_{a\in P(q)\setminus c} m^{(i)}_{a\to q}(x_q)} \tag{43}$$

$$= \sum_{\mathbf{x}_c\setminus x_q} \prod_{q'\in\mathcal{Q}_c\setminus q} \delta[\sigma(\mathbf{x}_{\mathcal{Q}_c}) = s_c] \underbrace{p_{q'}(x_{q'}) \prod_{c'\in P(q')\setminus c} m^{(i)}_{c'\to q'}(x_{q'})}_{=:m^{(i)}_{q\to c}(x_q)}. \tag{44}$$

The definition of the second type of messages (qubit to check) in the last line recovers the initial BP equations 60, 61 when identifying $\mathcal{Q}_c \equiv \Gamma(c)$ and $P(q) \equiv \Gamma(q)$. Note that, instead of following rule Eq. 38 or 39, standard BP thresholds the beliefs of the small regions Eq. 41.

Intuitively, the BP algorithm operates on the Tanner graph of the code by sending messages along its edges. Starting on the qubit site, it assigns initial probabilities of error (e.g. depending on the error channel) to the qubit nodes. This information is sent to the parity check nodes along the edges. The parity check nodes collect all incoming messages and send back a message to all adjacent qubits. In its components, this message contains the sum over all configurations compatible with the observed syndrome, excluding the target qubit. Subsequently, the marginal probability of the qubits is calculated according to the incoming messages. If not compatible or converged, these are sent back excluding the receiver's information. This procedure is shown graphically in Fig. 4.

2.2.3 Numerical results

Using this choice of regions, we obtain the decoding performance for hypergraph product codes based on random matrices and for topological surface codes shown in Fig. 5. The random codes are $(7, 4)$-qLDPC and the surface codes are $(4, 4)$-qLDPC. We see that the random codes show a good performance in the sense that increasing the distance of the code increasingly suppresses failures. The surface codes however show the opposite behavior. In both cases, the primary cause for a decoding failure is not a logical error, but a failure to return an error compatible with the syndrome or a failure of convergence.

2.2.4 Success and obstacles using BP

The disadvantages and problems involved in classical BP decoding are extensively covered in the literature. These mainly concern harmful patterns in the Tanner graphs of the code due to cycles or *trapping sets* [32]. Their existence in classical codes translates to quantum codes when

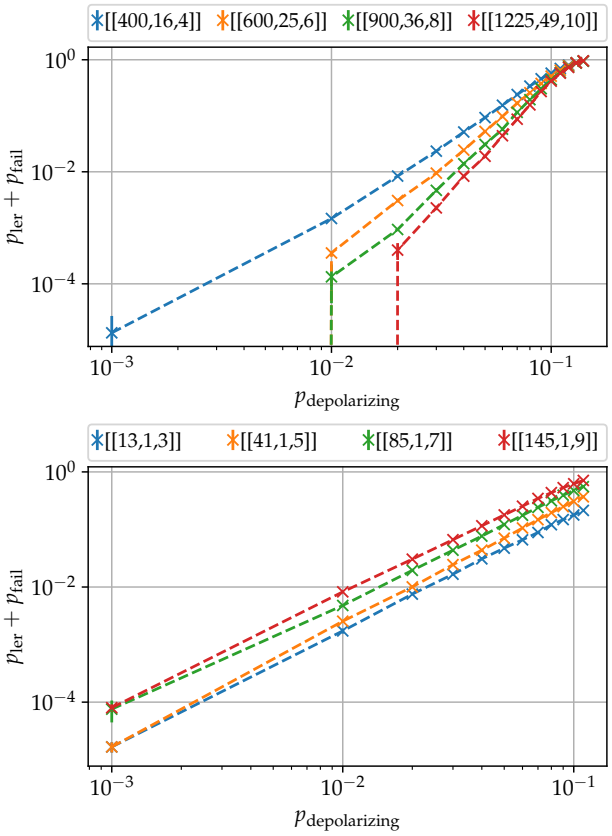

Figure 5: GBP with the Bethe approximation (standard BP), for HGP codes based on random classical codes (top) and topological surface codes (bottom). While the random codes show the emergence of a (pseudo-) threshold, the surface codes show decreasing decoding performance for increasing distance.

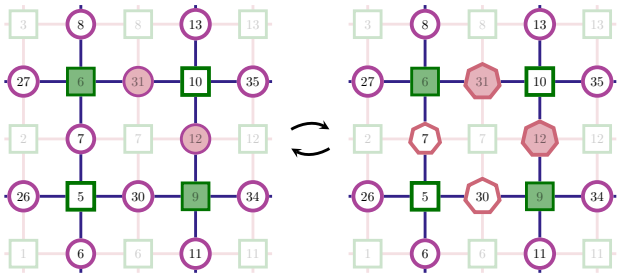

Figure 6: Split belief on a patch of the surface code, decoded with BP. Tanner graph representation with qubits as purple circles, parity-checks as green squares. X-Paulis are represented by blue and Z-Paulis by red edges. The initial error is indicated by filled qubits, the violated checks by filled squares. The error guess of the decoder by the heptagons. The Z-error on qubits $Z_{12}Z_{31}$ violates parity checks $\{6, 9\}$. The (degenerate) error pattern Z_7Z_{30} is symmetric to the original one. The standard BP decoder returns the union of all those qubits $Z_7Z_{30}Z_{12}Z_{31}$ with empty syndrome leading to a decoding failure.

using constructions based on the classical codes, like the hypergraph product construction. This also means that classical methods like message scheduling can be used to alleviate such problems [9]. The nature of quantum codes themselves introduces new obstacles to BP decoding. Certain syndromes allow for different error configurations, that can be translated to each other by symmetry transformations in the corresponding Tanner graph.

In a qubit-wise decoding fashion, the BP decoder assigns the same probability to each qubit involved in such configurations. With this *split belief*, the decoder thresholds all those qubits to the same error guess and therefore fails to converge. Some scheduling methods can break these symmetries but there is no general method to avoid them. Using irregular base codes and graphs of odd degree distribution also reduces the amount of symmetry, helping the decoder. An exemplary split belief is shown and explained in Fig. 6.

Because surface codes are highly symmetrical, lots of such split beliefs occur during decoding, which explains their bad performance. HGP codes with random base codes however show a good decoding performance because their local topology is inherited from the random classical codes that do not exhibit split beliefs.

There are two main strategies for improving the performance of BP. The first makes changes to the algorithm itself, *Memory Belief Propagation* for example shows good decoding performance at the cost of a slightly higher complexity [33]. A version of GBP was used to improve the decoding performance in quantum bicycle codes [13].

Other methods use the *soft* output of the BP algorithm (*i.e.* the marginal probabilities) and use them as input for post-processing methods. Notably *Ordered Statistics Decoding* allows to apply the combined BP+OSD decoder across a wide range of qLDPC codes, again at the cost of a higher complexity [11, 12].

3 GBP for the Surface Code

We use the same region graph as used for the Bethe approximation in Sec. 2.2.2, *i.e.* with large and small regions. The reason for that choice is that for surface codes, this construction method is equivalent to using the *cluster variational method*,

a standard method for constructing valid region graphs [27]. This choice of regions ensures that the smallest split beliefs, i.e. those related to two syndrome excitations, are resolved. The transition from the Tanner graph to the region graph is shown for a binary implementation in Fig. 7.

We implement the hard decision based on Eq. 38. We show how this helps with the prototypical split belief from Fig. 6 in Fig. 9. Using this hard decision procedure, we might still get a split belief within a single large region. However in our simulations, we observe that this is not the case, *i.e.* the arg max of the region is unique towards the end of the iterations.

3.1 Split and Repeat

We observe that while decoding based on the region beliefs using Eq. 39 improves upon standard BP, there still exist error guesses incompatible with the overall syndrome. However, applying the proposed correction usually reduces the syndrome weight. This is similar to one decoding iteration in the small-set flip algorithm (SSF), proposed by Leverrier, Tillich and Zémor [22]. In the SSF-algorithm, a configuration of qubits in the support of a stabilizer is flipped, if it decreases the syndrome weight. This works well for quantum codes with a sufficiently expanding Tanner graph. In surface codes however there are constant weight error patterns, that lead to failure of the SSF-algorithm. Our hard decision heuristic after the GBP inference that reduces the residual syndrome weight suggests that this issue can be overcome.

We use this insight to formulate a *split and repeat* procedure: After a run of GBP, we save the current error guess and reinitialize the decoding procedure with the syndrome of lower weight and a rescaled error probability. We repeat this until there is an empty overall syndrome. The overall error guess then is the sum of all intermediate errors. This algorithm is shown in Alg. 1. An exemplary run for a particularly harmful error pattern on a distance 9 surface code is shown in Fig. 8. We also see that the free energy is reduced during the course of decoding correspondingly.

Algorithm 1: GBP split repeat decoding for surface codes.

Input: Parity-Check matrix $\mathbf{H}$, syndrome s,
a-priori probability p_{init}, maximum number of iterations and repetitions $n_{\text{mi}}, n_{\text{mr}}$
Output: Error guess $\hat{e}$
$\mathcal{RG}$ = region graph constructed from $\mathbf{H}$ with Bethe approximation
$i = 0$
$\hat{e}_{\text{total}} = 0$
$s_i = s$
while $i < n_{mr}$ **do**
  $\tilde{p}_{\text{init}} = |p_{\text{init}} - \text{wt}(\hat{e}_{\text{total}})/n_{\text{qubits}}|$
  $\hat{e}_i = \text{GBP}(s_i, \mathcal{RG}, \tilde{p}_{\text{init}}, n_{\text{mi}})$
  $\hat{e}_{\text{total}} = \hat{e}_{\text{total}} + \hat{e}_i$
  if $\sigma(\hat{e}_{total}) =: s_{\text{GBP}} = s$ **then**
    return $\hat{e} = \hat{e}_{total}$
  else
    $s_{i+1} = s_i + s_{\text{GBP}}$
    $i = i + 1$
  end
end
return *Fail*

3.2 Dependence on Initial Probabilities

An additional parameter of the decoding procedure is the initial probability. As mentioned by Hagiwara et al., a linear decoder with a fixed initialization can correct a certain error set, independent of the error probability [34]. Kuo and Lai remark that choosing a fixed initialization can prevent fluctuations and increase decoding stability [33]. In our simulations, we observe that a fixed initialization error probability can lead to decoding failure. We therefore consider two further adaptions.

On the one hand, when re-initializing after a split procedure, we rescale the channel error probability by the weight of the current error guess, see Alg. 1.

On the other hand, when decoding still fails, we reinitialize the whole decoder with different initial probabilities. Assuming a good decoding performance when p_{init} is close to the channel error probability, we sample the new initial probability from a Gaussian with width 0.1 around the channel error probability.

3.3 Numerical Results

Remarkably, in our simulations we never observe a decoding failure and the decoder always returns

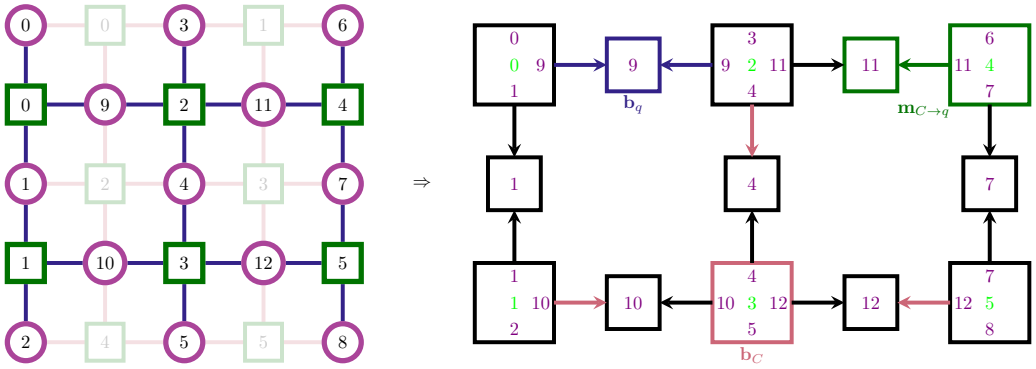

Figure 7: Tanner graph (left) to region graph (right) for the distance 3-surface code. Indicated in the region graph are the contributions to large region beliefs (red), small region beliefs (blue) and messages (green).

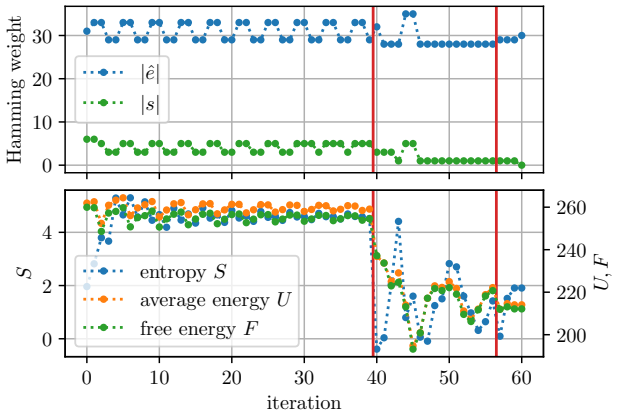

Figure 8: Weight of total error guess and residual syndrome (top) and thermodynamic quantities (bottom) during decoding of an error on a distance 9 surface code. Vertical lines represent the re-initialization after convergence or maximum number of iterations reached. We see that we can escape the oscillatory behavior by re-initializing.

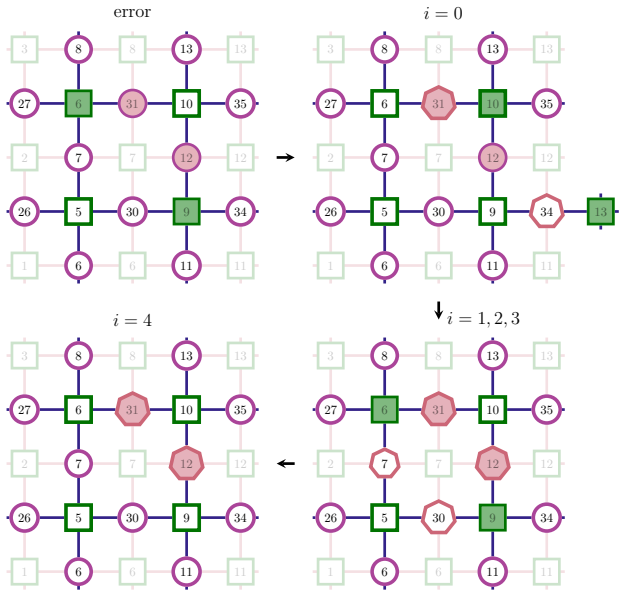

Figure 9: Split belief on a patch of the surface code, decoded with GBP. Tanner graph representation with qubits as purple circles, parity-checks as green squares. X-Paulis are represented by blue and Z-Paulis by red edges. The initial error is indicated by filled qubits, the violated checks by filled squares. The error guess of the decoder by the heptagons. The Z-error on qubits $Z_{12}Z_{31}$ violates parity checks $\{6, 9\}$. The (degenerate) error pattern Z_7Z_{30} is symmetric to the original one in the sense that they have equal weight. After $i = 4$ iterations, the GBP decoder returns the correct error, successfully decoding the split belief.

an error guess that puts the corrupted word back to the codespace. The results are shown for independent XZ- noise and depolarizing noise in Fig. 10 and 11 for the binary and quaternary implementation respectively. They generally show a decreasing logical error probability with increasing distance and a crossing indicating a threshold. The results for the threshold are summarized in Tab. 1.

Complexity In a naive implementation, directly adapting Eqs. 34 and 36, the GBP algorithm requires

- q^{d_c} multiplications per check region $c \rightarrow \leq n_c q^{d_c}$,
- q^{d_q} multiplications per qubit region $q \rightarrow \leq n_q q^{d_q}$,

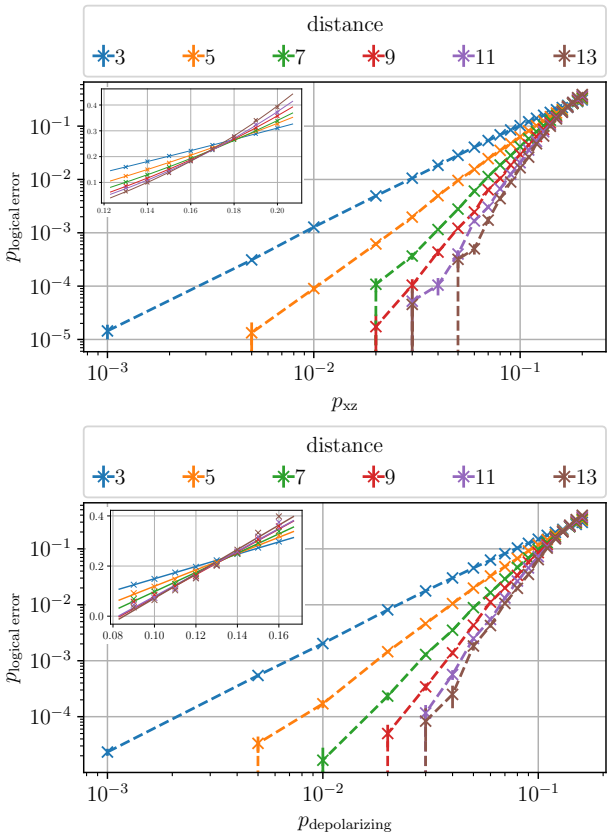

Figure 10: GBP for the surface code for independent XZ-noise (top) and depolarizing noise (bottom) in binary implementation. The logical error rate decreases with increasing distance. We estimate thresholds of $p_{\text{th}}^{\text{XY}} \approx 17\%$ and $p_{\text{th}}^{\text{depol.}} \approx 13.3\%$.

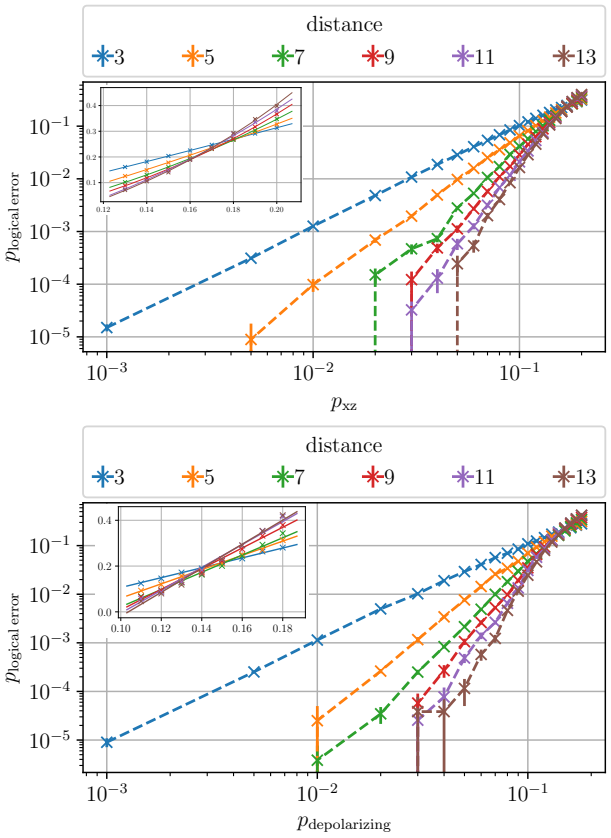

Figure 11: GBP for the surface code for independent XZ-noise (top) and depolarizing noise (bottom) in quaternary implementation. The logical error rate decreases with increasing distance. We estimate thresholds of $p_{\text{th}}^{\text{XY}} \approx 17\%$ and $p_{\text{th}}^{\text{depol.}} \approx 14\%$.

- q^{d_c-1} summations per marginalization of check region $c \rightarrow \leq n_c(q^{d_c-1})$,
- q multiplications and divisions per message calculation $\rightarrow \leq 2d_q n_q q$,

amounting to an overall asymptotic complexity of $\mathcal{O}(n_{\text{repetitions}} n_{\text{iterations}} q^{d_c} n_c)$. By implementing parallel the calculation of beliefs and messages, the explicit dependence on the code size can be omitted, $\mathcal{O}(n_{\text{repetitions}} n_{\text{iterations}})$. The amount of repetitions and iterations needed still depends on the code size and heuristically scale as shown in Tab. 2. They amount to an overall scaling of $\mathcal{O}(n_c^2)$ (GF(2)) and $\mathcal{O}(e^{\sqrt{n_c}} n_c^2)$ (GF(4)).

4 Summary and Outlook

We developed a decoder based on Generalized Belief Propagation using a specific hard decision method and an outer re-initialization loop. With these adaptations, the decoder is able to decode the surface code. The logical error probability decreases with growing distance below a certain qubit error probability indicating the emergence of a threshold of about 14% for depolarizing noise and 17% for independent bit- and phase-flip noise. As is typical for practical decoders, these values fall short of theoretical upper bounds but offer a lower decoding complexity. When comparing to known decoding algorithms, our decoder shows similar threshold values compared to BP-OSD. The main ingredient to the OSD-post processing is matrix inversion, which scales with the third power in the number of rows, *i.e.* $\mathcal{O}(n_c^3)$ [12]. Minimum weight perfect matching achieves a higher threshold but scales in general as $\mathcal{O}(n_q^3)$ [37]. The almost linear time Union find decoder also achieves a slightly higher threshold [38].

Future work can include a lower complexity implementation that is based on log-likelihood-ratios, which is frequently used in standard BP algorithms to reduce complexity. This would al-

Table 1: Thresholds from error sampling on the XZ-channel and the depolarizing channel for binary and quaternary implementation. The thresholds fall short of the optimal thresholds obtained by statistical mechanic methods but are similar to the ones from BP-OSD decoding. The optimal threshold for the XZ-channel is obtained from the single-Pauli threshold $p_{\text{th}}^X \approx 10.9\%$ as $p_{\text{th}}^{XZ} = 2p_{\text{th}}^X - (p_{\text{th}}^X)^2$.

Ch.	q	p_{th}	BP-OSD	optimal
XZ	2	17%	17.6% [12]	20.6% [35]
	4	17%		
depol.	2	13.3%		18.9% [36]
	4	14%		

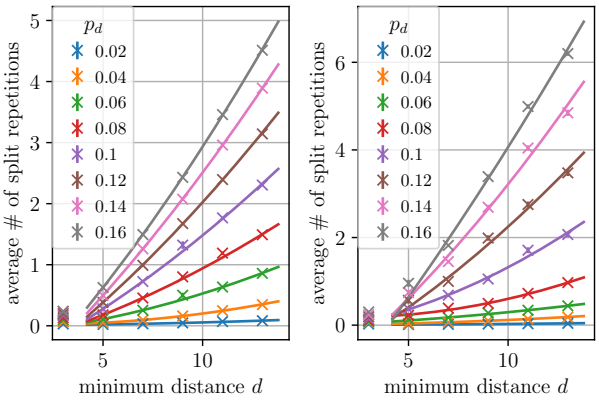

Figure 13: Average number of split repetitions scales quadratic with the code distance, shown is a quadratic fit for distances $d \geq 5$. (Left) Binary implementation in the depolarizing channel. (Right) Quaternary implementation in the depolarizing channel. The quaternary implementation needs on average more repetitions.

Table 2: Scaling of iterations and repetitions of the decoder with code distance , heuristically obtained from simulations.

	GF(2)	GF(4)
Iterations	$\mathcal{O}(d^2) = \mathcal{O}(n_c)$	$\mathcal{O}(d^2) = \mathcal{O}(n_c)$
Split rep.	$\mathcal{O}(d^2) = \mathcal{O}(n_c)$	$\mathcal{O}(d^2) = \mathcal{O}(n_c)$
p_{init} rep.	$\mathcal{O}(d^0) = \mathcal{O}(1)$	$\mathcal{O}(\exp(d))$

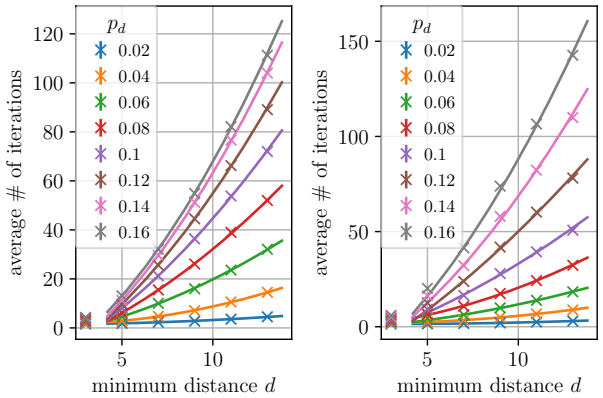

Figure 12: Average number of inner iterations scales quadratic with the code distance, shown is a quadratic fit for distances $d \geq 5$. (Left) Binary implementation in the depolarizing channel. (Right) Quaternary implementation in the depolarizing channel. The quaternary implementation needs on average more iterations.

low the decoder to be tested for more general quantum LDPC codes, where finding a fast and general decoder is ongoing research.

Additionally, all simulations were performed with code capacity noise, *i.e.* the data qubits experience noise through the quantum channel and the syndrome readout is assumed perfect. A next step on the road towards fault tolerance is to extend the decoding scheme to more realistic noise models, for example including faulty syndrome measurements. There are recent results suggesting that belief propagation is also suitable for syndrome noise when using repeated measurements and even in a single-shot decoding scheme [39, 40].

Simulation Methods

The decoder is implemented in a GF(q) formalism with both $q = 2$ and $q = 4$ in C++ making use of libraries libDai [41], the lemon Graph library [42], xtensor [43], NTL [44] and nlohmann JSON headers [45]. The code can be found on github [46].

Acknowledgements

We would like to thank B.M. Terhal and J. Knörzer for comments on the manuscript and M. Müller and D.P. DiVincenzo for facilitating this project. We acknowledge support from the EU Quantum Technology Flagship grant AQTION

under grant agreement number 820495 and by the BMBF project MUNIQC-ATOMS. This research is also part of the Munich Quantum Valley (K-8), which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus. MR was supported by ERC grant EQEC No. 682726 during the initial part of this work. Simulations were performed with computing resources granted by RWTH Aachen University under project *thes1045*.

References

[1] Nikolas P. Breuckmann and Jens Niklas Eberhardt. "Quantum Low-Density Parity-Check Codes". PRX Quantum **2**, 040101 (2021).

[2] Daniel Gottesman. "Fault-tolerant quantum computation with constant overhead". Quantum Information & Computation **14**, 1338–1372 (2014).

[3] A Robert Calderbank and Peter W Shor. "Good quantum error-correcting codes exist". Physical Review A **54**, 1098 (1996).

[4] Alexei Ashikhmin, Simon Litsyn, and Michael A Tsfasman. "Asymptotically good quantum codes". Physical Review A **63**, 032311 (2001).

[5] Pavel Panteleev and Gleb Kalachev. "Asymptotically good quantum and locally testable classical ldpc codes". In Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing. Pages 375–388. (2022).

[6] Anthony Leverrier and Gilles Zémor. "Quantum tanner codes". In 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). Pages 872–883. (2022).

[7] Ting-Chun Lin and Min-Hsiu Hsieh. "Good quantum ldpc codes with linear time decoder from lossless expanders" (2022). arXiv:2203.03581.

[8] David JC MacKay and Radford M Neal. "Near shannon limit performance of low density parity check codes". Electronics letters **33**, 457–458 (1997).

[9] Nithin Raveendran and Bane Vasić . "Trapping sets of quantum LDPC codes". Quantum **5**, 562 (2021).

[10] Robert Raussendorf and Jim Harrington. "Fault-Tolerant Quantum Computation with High Threshold in Two Dimensions". Phys. Rev. Lett. **98**, 190504 (2007).

[11] Pavel Panteleev and Gleb Kalachev. "Degenerate quantum LDPC codes with good finite length performance". Quantum **5**, 585 (2021).

[12] Joschka Roffe, David R White, Simon Burton, and Earl Campbell. "Decoding across the quantum low-density parity-check code landscape". Physical Review Research **2**, 043423 (2020).

[13] Nithin Raveendran, Mohsen Bahrami, and Bane Vasic. "Syndrome-generalized belief propagation decoding for quantum memories". In ICC 2019 - 2019 IEEE International Conference on Communications (ICC). Pages 1–6. (2019).

[14] E. Berlekamp, R. McEliece, and H. van Tilborg. "On the inherent intractability of certain coding problems (corresp.)". IEEE Transactions on Information Theory **24**, 384–386 (1978).

[15] Daniel Gottesman. "Stabilizer codes and quantum error correction" (1997). arXiv:quant-ph/9705052.

[16] R. Tanner. "A recursive approach to low complexity codes". IEEE Transactions on Information Theory **27**, 533–547 (1981).

[17] Andrew Steane. "Multiple-particle interference and quantum error correction". Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **452**, 2551–2577 (1996).

[18] M. Sipser and D. A. Spielman. "Expander codes". IEEE Transactions on Information Theory **42**, 1710–1722 (1996).

[19] David JC MacKay. "Information theory, inference and learning algorithms". Cambridge university press. (2003).

[20] A.Yu. Kitaev. "Fault-tolerant quantum computation by anyons". Annals of Physics **303**, 2–30 (2003).

[21] J. Tillich and G. Zemor. "Quantum ldpc codes with positive rate and minimum distance proportional to $n^{1/2}$". In 2009 IEEE International Symposium on Information Theory. Pages 799–803. (2009).

[22] A. Leverrier, J. Tillich, and G. Zémor. "Quantum expander codes". In 2015 IEEE

56th Annual Symposium on Foundations of Computer Science. Pages 810–824. (2015).

[23] Jonathan S Yedidia, William T Freeman, and Yair Weiss. "Generalized belief propagation". In NIPS. Volume 13, pages 689–695. (2000).

[24] Brendan J Frey, Frank R Kschischang, Hans-Andrea Loeliger, and Niclas Wiberg. "Factor graphs and algorithms". In Proceedings of the Annual Allerton Conference on Communication Control and Computing. Volume 35, pages 666–680. (1997).

[25] Max Welling. "On the choice of regions for generalized belief propagation" (2012). arXiv:1207.4158.

[26] Vladimir Fanaskov. "Gaussian belief propagation solvers for nonsymmetric systems of linear equations". SIAM Journal on Scientific Computing **44**, A77–A102 (2022).

[27] Jonathan S Yedidia, William T Freeman, and Yair Weiss. "Constructing free-energy approximations and generalized belief propagation algorithms". IEEE Transactions on information theory **51**, 2282–2312 (2005).

[28] R. Gallager. "Low-density parity-check codes". IRE Transactions on Information Theory **8**, 21–28 (1962).

[29] Judea Pearl. "Reverend bayes on inference engines: A distributed hierarchical approach". In Proceedings of the Second AAAI Conference on Artificial Intelligence. Pages 133–136. AAAI'82. AAAI Press (1982).

[30] David Poulin and Yeojin Chung. "On the iterative decoding of sparse quantum codes". Quantum Information & Computation **8**, 987–1000 (2008).

[31] Alex Rigby, J. C. Olivier, and Peter Jarvis. "Modified belief propagation decoders for quantum low-density parity-check codes". Physical Review A**100** (2019).

[32] Aiden Price and Joanne Hall. "A survey on trapping sets and stopping sets" (2017). arXiv:1705.05996.

[33] Kao-Yueh Kuo and Ching-Yi Lai. "Exploiting degeneracy in belief propagation decoding of quantum codes". npj Quantum Information**8** (2022).

[34] Manabu Hagiwara, Marc P. C. Fossorier, and Hideki Imai. "Fixed initialization decoding of ldpc codes over a binary symmetric channel". IEEE Transactions on Information Theory **58**, 2321–2329 (2012).

[35] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. "Topological quantum memory". Journal of Mathematical Physics **43**, 4452–4505 (2002).

[36] H. Bombin, Ruben S. Andrist, Masayuki Ohzeki, Helmut G. Katzgraber, and M. A. Martin-Delgado. "Strong resilience of topological codes to depolarization". Phys. Rev. X **2**, 021004 (2012).

[37] Vladimir Kolmogorov. "Blossom v: a new implementation of a minimum cost perfect matching algorithm". Mathematical Programming Computation **1**, 43–67 (2009).

[38] Nicolas Delfosse and Naomi H Nickerson. "Almost-linear time decoding algorithm for topological codes". Quantum **5**, 595 (2021).

[39] Antoine Grospellier, Lucien Grouès, Anirudh Krishna, and Anthony Leverrier. "Combining hard and soft decoders for hypergraph product codes". Quantum **5**, 432 (2021).

[40] Armanda O Quintavalle, Michael Vasmer, Joschka Roffe, and Earl T Campbell. "Single-shot error correction of three-dimensional homological product codes". PRX Quantum **2**, 020340 (2021).

[41] Joris M. Mooij. "Libdai: A free and open source c++ library for discrete approximate inference in graphical models". J. Mach. Learn. Res. **11**, 2169–2173 (2010).

[42] Balázs Dezső, Alpár Jüttner, and Péter Kovács. "Lemon – an open source c++ graph template library". Electronic Notes in Theoretical Computer Science **264**, 23–45 (2011).

[43] Johan Mabille, Sylvain Corlay, and Wolf Vollprecht (2016). code: xtensor-stack/xtensor.

[44] Victor Shoup. "Ntl: A library for doing number theory". (2021). url: `https://libntl.org`. code: libntl/ntl.

[45] Niels Lohmann (2022). code: nlohmann v3.10.5.

[46] Josias Old (2022). code: josiasold/gbp.

A Variational Methods in Statistical Mechanics

The paradigmatic example of a variational method in quantum mechanics is the Ritz method [1]. Here, the setting is that we are given a Hamiltonian and the task is to find its ground-state. Since this task is in general computationally hard already for the simplest non-trivial practically relevant Hamiltonians, it is fruitful and instructive to develop systematic methods to construct trial wavefunctions that approximate the ground state wavefunction while retaining computational feasibility. The fundamental insight here is that the energy expectation value of the problem Hamiltonian evaluated on *any* state is lower bounded by the expectation value of the true ground state,

$$\langle\psi_{\text{trial}}|\,\hat{H}\,|\psi_{\text{trial}}\rangle \geq \langle E_0|\,\hat{H}\,|E_0\rangle\,, \tag{45}$$

which essentially only relies on the fact that we can expand $|\psi\rangle$ in the Hamiltonian eigenbasis, upon which the statement follows immediately.

This paradigm of systematically constructing trial states can be extended to mixed states and finite temperature. The role of energy is taken over by the (Helmholtz) free energy $F = E - TS = -\frac{1}{\beta}\log Z$, where S is the entropy and $Z = \operatorname{tr}\exp\left(-\beta\hat{H}\right)$ the partition function. With respect to the free energy, any trial state fulfils the Bogoliubov inequality

$$\operatorname{tr}\hat{F}\rho_{\text{trial}} \geq \operatorname{tr}\hat{F}\rho_{\text{c}}. \tag{46}$$

Note that this contains the Rayleigh-Ritz inequality in the zero temperature limit. It can be proved by exploiting the Gibbs inequality

$$\operatorname{tr}\left[A\log A - A\log B\right] \geq 0, \tag{47}$$

which holds for any $A, B \geq 0$ provided $\operatorname{tr}A = \operatorname{tr}B$. Plugging in the canonical ensemble (or Gibbs state) $\rho_{\text{c}} = \exp\left(-\beta\hat{H}\right)/Z$ for B leads to

$$-S(\rho_{\text{trial}}) + \beta\operatorname{tr}\hat{H}\rho_{\text{trial}} + \log Z \geq 0, \tag{48}$$

which can be rearranged into the Bogoliubov inequality, stating that the free energy of any trial density matrix ρ_{trial} is lower bounded by the free energy of the canonical ensemble. This permits the extension of the variational principle to mixed states, where we now try to approximate the Gibbs state by a trial density matrix. In the context of the present work, we are dealing "only" with probability distributions, so let us point out the rather trivial fact that the above inequality in particular also holds when $\hat{H}$ is diagonal and the density operators are simply multivariate probability distributions.

[1] colloquially often known as Rayleigh-Ritz

A.1 Minimization of the Free Energy

For a variational ansatz, we introduce the trial probability distribution, the *belief* $b(\mathbf{x})$. Equivalent to Eq. 48, it holds that the free energy of such a trial probability distribution is lower bounded by the free energy of the real distribution, *i.e.*

$$F(b) := U(b) - S(b) \geq F_H \tag{49}$$

$$\sum_{\mathbf{x}\in P} b(\mathbf{x})E(\mathbf{x}) + \sum_{\mathbf{x}\in P} b(\mathbf{x})\ln b(\mathbf{x}) \geq -\ln Z. \tag{50}$$

Here, $U(b)$ is the *(variational) average energy* and $S(b)$ the *(variational) entropy* of the trial state. Plugging in the definition of the energy gives

$$F(b) = F_H + \sum_{\mathbf{x}\in P} b(\mathbf{x})\ln\frac{b(\mathbf{x})}{p(\mathbf{x})} = F_H + D(b||p) \tag{51}$$

with $F(b) \geq F_H$ and equality iff $b(\mathbf{x}) = p(\mathbf{x})$. $D(b||p)$ is the Kullback-Leibler divergence that gives a measure of how close the trial distribution b is to the "true" distribution p. Because $D(b||p) \geq 0$ with equality iff $b(\mathbf{x}) = p(\mathbf{x})$, a minimization procedure of $D(b||p)$ with respect to $b(\mathbf{x})$ can exactly compute F_H and recover $p(\mathbf{x})$.

Due to the intractability of a brute force approach, the trial functions are generally restricted or approximated by some factorized form.

B Representation of Stablizer Codes

Binary representation $q = 2$. The binary representation maps Pauli words of length n to binary vectors of length $2n$. We can represent any Pauli word (up to a phase) by $E = X^{\mathbf{e}_x}Z^{\mathbf{e}_z}$, where $\mathbf{e}_x, \mathbf{e}_z \in \mathrm{GF}(2)^n$. The binary representation then are the concatenations $\mathbf{H} = (\mathbf{H}_X, \mathbf{H}_Z)$

and $\mathbf{e} = (\mathbf{e}_x, \mathbf{e}_z)$ and it holds

$$E_i = \begin{cases} X & \text{if} \quad \mathbf{e}_i = 1 \quad \text{and} \quad \mathbf{e}_{i+n} = 0, \\ Y & \text{if} \quad \mathbf{e}_i = 1 \quad \text{and} \quad \mathbf{e}_{i+n} = 1, \\ Z & \text{if} \quad \mathbf{e}_i = 0 \quad \text{and} \quad \mathbf{e}_{i+n} = 1. \end{cases} \tag{52}$$

The addition in GF(2) corresponds to addition modulo 2, $1 + 1 = 0$. The symplectic product of to vectors $\mathbf{e} = (\mathbf{e}_x, \mathbf{e}_z), \mathbf{e}' = (\mathbf{e}'_x, \mathbf{e}'_z)$ is defined via

$$\mathbf{e} \star \mathbf{e}' := \mathbf{e}\mathbf{P}\mathbf{e}' = \mathbf{e}_x \cdot \mathbf{e}'_z + \mathbf{e}_z \cdot \mathbf{e}'_x$$
$$\text{with} \quad \mathbf{P} := \begin{pmatrix} 0 & \mathbf{1}_n \\ \mathbf{1}_n & 0 \end{pmatrix} \quad \text{and} \quad .$$

the scalar product, such that

$$\mathbf{H} \star \mathbf{e} = \mathbf{H}\mathbf{P}\mathbf{e}. \tag{53}$$

Quaternary representation $q = 4$. In GF(4), addition and multiplication can be defined via its addition and multiplication tables

$+$	0	1	ω	$\bar{\omega}$
0	0	1	ω	$\bar{\omega}$
1	1	0	$\bar{\omega}$	ω
ω	ω	$\bar{\omega}$	0	1
$\bar{\omega}$	$\bar{\omega}$	ω	1	0

$\times$	0	1	ω	$\bar{\omega}$
0	0	0	0	0
1	0	1	ω	$\bar{\omega}$
ω	0	ω	$\bar{\omega}$	1
$\bar{\omega}$	0	$\bar{\omega}$	1	ω

. (54)

The group operation is naturally represented by addition, if the Paulis are mapped via

$$I \to 0, \quad X \to 1, \quad Y \to \overline{\omega}, \quad Z \to \omega. \tag{55}$$

In order to define the symplectic product, two more definitions are needed, the conjugation and trace in GF(4),

- Conjugation: $\mathrm{GF}(4) \to \mathrm{GF}(4) : \alpha \to \overline{\alpha} = \alpha \times \alpha$,
- Trace: $\mathrm{GF}(4) \to \mathrm{GF}(4) : \alpha \to \mathrm{Tr}\{\alpha\} = \alpha + \overline{\alpha}$.

Then

$$\mathbf{e} \star \mathbf{e}' := \mathrm{Tr}\{\mathbf{e} \cdot \mathbf{e}'\} = \mathrm{Tr}\left\{\sum_q \mathbf{e}_q \times \overline{\mathbf{e}'_q}\right\} \tag{56}$$

such that

$$\mathbf{s} = \mathbf{H} \star \mathbf{e} = \left(\mathrm{Tr}\left\{\sum_q \mathbf{H}_{cq} \times \overline{\mathbf{e}_q}\right\}\right)_{c=0}^{n-k-1} . \tag{57}$$

C Hypergraph Product Construction

The Tanner graph of the hypergraph product (quantum) code $\mathcal{T}_\mathcal{Q}$ is based on the Cartesian product of the classical Tanner graphs $\mathcal{T}_{\mathcal{C}_1}$ and $\mathcal{T}_{\mathcal{C}_2}$. The Cartesian product of two graphs $\mathcal{T}_1 = (N_1 = V_1 \cup C_1, E_1)$ and $\mathcal{T}_2 = (N_2 = V_2 \cup C_2, E_2)$ is the bipartite graph $\mathcal{T}_{1\times 2} =: \mathcal{T}_1 \times \mathcal{T}_2 = (N_{1\times 2}, E_{1\times 2})$ with

$$N_{1\times 2} = \{n_1 n_2 | n \in N_1, n_2 \in N_2\} \tag{58}$$
$$E_{1\times 2} = \{(n_1 n_2, n'_1 n'_2) | (n_1 = n'_1 \wedge (n_2, n'_2) \in E_2) \vee (n_2 = n'_2 \wedge (n_1, n'_1) \in E_1)\}. \tag{59}$$

In words: the vertex set of the resulting graph is the Cartesian product of the vertex sets of the graph factors. There is an edge between vertices in the resulting graph if any of their partial vertices shared an edge in their graph factor. The graph constructed from the Cartesian product of two bipartite graphs is again bipartite. In order to derive a code, the vertex set of the new graph is partitioned into $N_{1\times 2} = Q \cup (C_X \cup C_Z)$ with

- qubits: $Q := \{n_1 n_2 | (n_1 \in V_1 \wedge n_2 \in V_2) \vee (n_1 \in C_1 \wedge n_2 \in C_2)\}$
- X-type stabilizers: $C_Z := \{n_1 n_2 | (n_1 \in C_1 \wedge n_2 \in V_2)\}$
- Z-type stabilizers: $C_X := \{n_1 n_2 | (n_1 \in V_1 \wedge n_2 \in C_2)\}$

Chosen like that, the graph corresponds to a Tanner graph of a quantum CSS code. The commutation condition is fulfilled since whenever $n_i \in V_i$ is adjacent to $n'_i \in C_i$ in $\mathcal{T}_{\mathcal{C}_i}$, there are exactly two vertices (qubits) in $\mathcal{Q}$ which are adjacent to the constructed X-type stabilizer $n'_i n_j$ and Z-type stabilizer $n_i n'_j$: $n'_i n'_j$ and $n_i n_j$. Twofold anticommutation then gives commutation.

D Belief Propagation Equations

We denote by $\Gamma(\bullet)$ the neighbors of node $\bullet$ and by $\sigma(\bullet)$ the parity of configuration $\bullet$. For a detailed description of the steps, see text.

- Qubit to check messages

$$m^{(i+1)}_{q\to c} \propto p_0(E_q) \prod_{c' \in \Gamma(q)\setminus c} m^{(i)}_{c'\to q}(E_q) \tag{60}$$

- Check to qubit messages

$$m_{c\to q}^{(i)} \propto \sum_{\mathbf{E}_{\Gamma(c)\setminus q}} \delta[\sigma(\mathbf{E})_c = s_c] \prod_{q'\in\Gamma(c)\setminus q} m_{q'\to c}^{(i)}(E_{q'}) \tag{61}$$

- Belief

$$b_q^{(i)}(E_q) \propto p_0(E_q) \prod_{c\in\Gamma(q)} m_{c\to q}^{(i)}(E_q) \tag{62}$$

- hard decision / error guess

$$\hat{E}_q^{(i)} = \arg\max_{E_q} b_q^{(i)}(E_q) \tag{63}$$