Tensor Network Decoding Beyond 2D

Christophe Piveteau, Christopher T. Chubb, and Joseph M. Renes

Institute for Theoretical Physics, ETH Zürich, Switzerland

Decoding algorithms based on approximate tensor network contraction have proven tremendously successful in decoding 2D local quantum codes such as surface/toric codes and color codes, effectively achieving optimal decoding accuracy. In this work, we introduce several techniques to generalize tensor network decoding to higher dimensions so that it can be applied to 3D codes as well as 2D codes with noisy syndrome measurements (phenomenological noise or circuit-level noise). The three-dimensional case is significantly more challenging than 2D, as the involved approximate tensor contraction is dramatically less well-behaved than its 2D counterpart. Nonetheless, we numerically demonstrate that the decoding accuracy of our approach outperforms state-of-the-art decoders on the 3D surface code, both in the point and loop sectors, as well as for depolarizing noise. Our techniques could prove useful in near-term experimental demonstrations of quantum error correction, when decoding is to be performed offline and accuracy is of utmost importance. To this end, we show how tensor network decoding can be applied to circuit-level noise and demonstrate that it outperforms the matching decoder on the rotated surface code. Our code is available at https://github.com/ChriPiv/tndecoder3d.

1 Introduction

The practical realization of quantum technologies is impeded by the inherent sensitivity of quantum systems to noise. Quantum error correction [1, 2] is generally considered the solution to this problem, as it allows for fault-tolerant processing and transmission of quantum information. The idea is to embed the quantum information into a larger system using a quantum code to increase its resilience against noise. The stabilizer formalism [3] has proven to be a convenient and useful framework to construct and describe quantum codes. The error correction procedure for a stabilizer code involves the measurement of certain stabilizer operators, and the measurement outcomes (the syndrome) are subsequently processed by a classical decoding algorithm (or decoder) to select a correction operation to be applied on the system.

Tensor network (TN) decoders operate by approximately contracting one or more relevant tensor networks and subsequently deciding on the correction operation from the contraction results. The relevant tensor networks describe the probability of a given logical error conditioned on the observed syndrome. If the tensor networks were contracted exactly, the resulting decoder would be optimal, so the only heuristic lies in the approximate contraction. Generally, the layout of the networks follows the locality of the code from which they were derived, and in the case of correlated noise from the locality of those correlations [4]. Previous work has focused mainly on 2D local codes, i.e. codes whose stabilizer operators are local when the qubits are laid out in a two-dimensional array. This family of codes includes topological codes, such as the 2D toric/surface code [5] and the 2D color code [6]. TN decoding performs remarkably well on all of these codes and has been shown to achieve essentially optimal decoding accuracy at competitive runtimes for bit-flip, phase-flip, and depolarizing noise [4, 7, 8]. Another approach is to base the tensor network on the encoding circuit of the code [9]; this has recently been shown to perform well on codes with local circuits of low depth [10].

However, while approximate contraction of 2D TNs is well understood and in most cases numerically well-behaved, approximate contraction of 3D networks is not and remains an active field of research. The fundamental obstruction is that it is not possible to define a canonical gauge of a tensor network that is not a tree [11, §5.2], so for example optimally truncating a projected entangled pair state (PEPS) network is more difficult than a matrix product state (MPS). This restriction is especially unfortunate for practical applications, since in experimental platforms the syndrome measurements are noisy and must be repeated multiple times (typically on the order of the distance of the code). The resulting decoding problem for a 2D local code under circuit level noise is fundamentally three-dimensional, and thus involves three-dimensional TNs. For this reason, existing TN decoding schemes are not applicable.

Here we introduce a family of techniques which extend the framework of TN decoding beyond two dimensions. The resulting decoding accuracy can in principle be made arbitrarily close to optimal, and thus we focus on

accuracy and eschew the issue of runtime. This approach lends itself to experimental demonstrations of error-corrected quantum memories on near-term hardware, when the decoding process is performed offline and runtime is not an immediate issue. Near-optimal decoders are also useful in probing the fundamental performance of codes independent of any decoder heuristics. They can thus be useful in the design of quantum codes [12].

We describe our main contributions in more detail below, by way of sketching the structure of the remainder of the paper. The following Section 2 reviews the properties of stabilizer codes, their optimal decoding as finding the most probable logical coset consistent with the syndrome, and the formalism of detector error models. It also gives a short overview of tensor networks.

In Section 3 we introduce two distinct representations (applicable to any stabilizer code) of the logical coset probabilities as TN contractions. The *generator picture* mathematically corresponds to the statistical mechanical mapping used in [4, 7, 8], but without the associated statistical mechanical language. The *detector picture* can be regarded as a dual representation to the generator picture. To our knowledge it has not been previously proposed in literature in this context. The idea of the detector picture is to choose the tensor network to represent a sum over all possible error patterns, and then remove the patterns that are not compatible with an observed syndrome and a given logical operator. In contrast, the generator picture takes the opposite approach, first fixing a representative Pauli error r that is compatible with the observed syndrome and logical operator, and then summing over the probabilities of errors $r \cdot s$ over all stabilizers s to obtain the total probability of the corresponding logical coset.

In Section 4 we propose a technique for approximate contraction of both detector and generator TNs: Sweep a two-dimensional TN from one side of the 3D TN to the other, contracting the tensors that are encountered along the way layer-by-layer, all while continuously truncating the bond dimensions of the 2D network to keep it below a given threshold. This can be seen as a higher-dimensional generalization of the sweep-line algorithm used in [8]. Unlike the case of sweeping with an MPS, one cannot define a canonical form for a 2D tensor network that includes loops, making it difficult to realize a numerically well-behaved truncation scheme. We make use of the *simple update* technique [13–15], commonly used in the condensed matter community to simulate the time evolution of spin chains. The idea is to keep track of a rank-one approximation of the environment of each tensor in the loopy tensor network, which allows for a more accurate truncation. This method was recently shown to be sufficient for fast and highly accurate simulations of 2D quantum systems with more than a thousand qubits [16]. Its theoretical justification lies in the fact that it makes the loopy network approach the *Vidal gauge* when the applied gates are close enough to the identity [17].

In Section 5 we present the results of numerical simulations of TN decoding. First, we consider independent bit- and phase-flip noise on the unrotated 3D surface code, differentiating the stabilizers into the *point sector* of weight-six X-type stabilizers and the *loop sector* of weight-four Z-type stabilizers. Note that decoding the point sector can equivalently be regarded as decoding the 2D surface code with a phenomenological noise model [18]. These two sectors are decoded with detector and generator TNs, respectively. As shown in Figures 1a and 1b, TN decoding outperforms the state of the art in both sectors, namely minimum-weight perfect matching for the point sector [19] and BP+OSD for the loop sector [20]. In a second step, we consider depolarizing noise on the unrotated 3D surface code, which involves both the point and loop sector in a correlated manner. We find that the detector TN decoder performs considerably better than the generator TN. As depicted in Figure 1c, it also significantly improves over the existing state-of-the-art decoder, which is BP+OSD [21].

We also develop a technique for TN decoding of circuit-level noise. One could make use of the circuit itself to construct the TN, but while the resulting TNs are three-dimensional, they are very large even for modest-sized decoding tasks. Instead, we make use of Stim's detector error models [22] to construct the TN, and consider the rotated surface code as a proof-of-principle. Even with this approach, the resulting 3D tensor network is notoriously large and exhibits a very complicated topology, as seen in Figures 8 and 10. To solve this problem, we first "compress" the TN to a cubic form via local approximate contraction. The result can then be globally contracted with the simple update method described above. Numerical results for the rotated surface code are depicted in Figure 1d and compared to PyMatching [23, 24].

Table 1 summarizes our observed thresholds and compares them with decoders in previous literature. The thresholds are estimated using the bootstrapping procedure explained in [8], see [25] for more details. We note that these thresholds should be taken with a grain of salt, since the considered code distances are rather small and therefore finite-size effects might have a significant effect. We argue that the improvements in logical error rates are more significant here, especially for the circuit-level noise where the exact threshold might be of small importance for experiments.

Finally, we discuss the ample directions for further research in Section 6. Our software implementation of the tensor network decoder is available at https://github.com/ChriPiv/tndecoder3d.

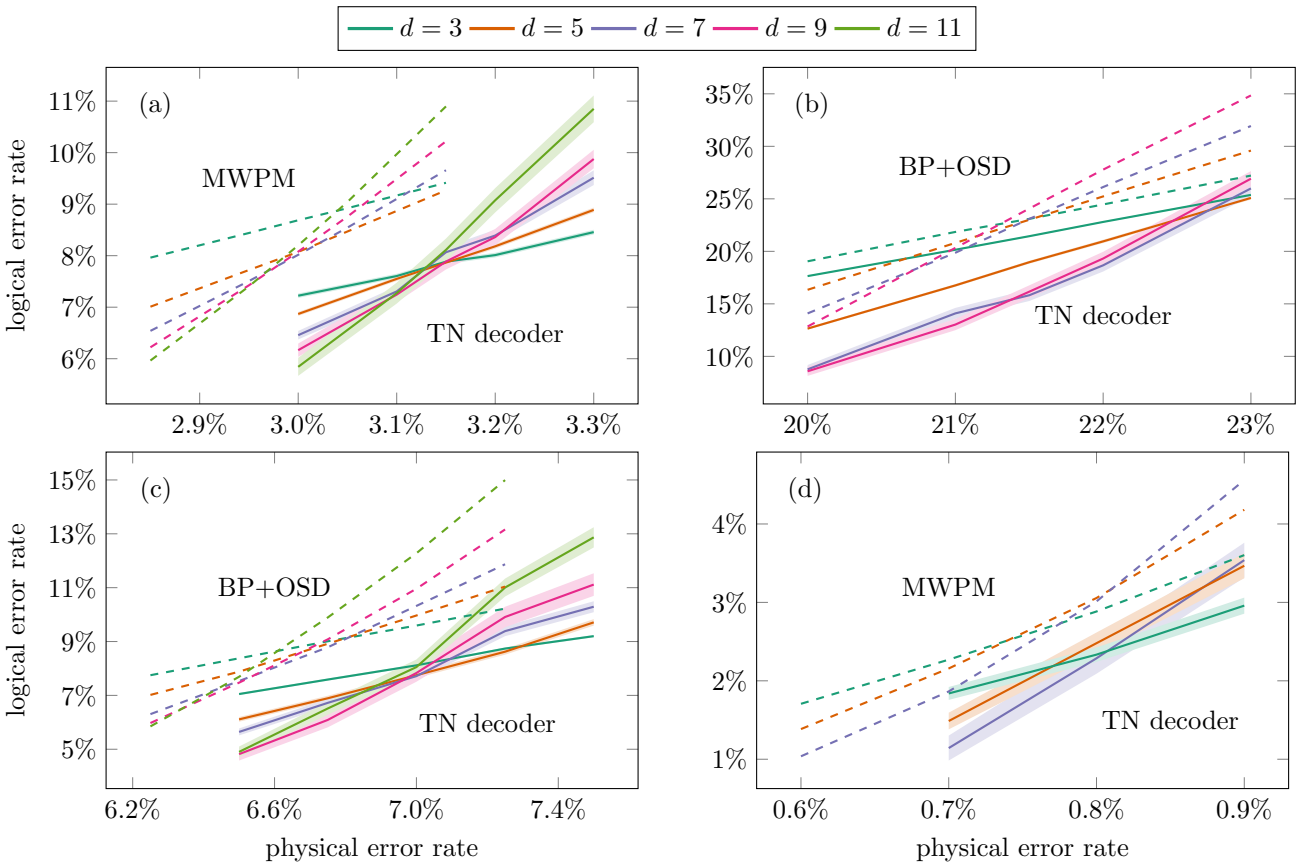


Figure 1: Threshold plots for (a) point sector, (b) loop sector, (c) depolarizing noise for the 3D unrotated surface code. For all three cases, we compare the tensor network decoder with the current state of the art. (d) depicts how the tensor network decoder compares to the matching decoder for circuit-level noise on the rotated surface code. The thresholds are depicted in Table 1.

	TN decoder (ours)	**Other decoders**		**Optimal threshold**
Point sector	$3.136^{+0.012}_{-0.014}\%$	Matching [19]	$2.93 \pm .02\%$	$\approx 3.3\%$ [26]
		RG [27]	$1.9 \pm 0.4\%$	
Loop sector	$22.788^{+0.123}_{-0.107}\%$	Erasure mapping [28]	$\approx 12.2\%$	$23.180 \pm 0.004\%$ [29, 30]
		Toom's rule [31]	$\approx 14.5\%$	
		Sweep [32]	$\approx 15.5\%$	
		RG [33]	$\approx 17.2\%$	
		Neural network [34]	$\approx 17.5\%$	
		BP+OSD [20]	$21.55 \pm 0.01\%$	
Depolarising	$7.067^{+0.034}_{-0.033}\%$	BP+OSD [21]	$5.95 \pm 0.03\%$	unknown
		BP+OSD (ours)[1]	$6.715 \pm 0.012\%$	
Circuit-level depolarising	$\approx 0.8\%$	Matching (ours)	$\approx 0.78\%$	unknown

Table 1: Comparison of thresholds between TN decoders and prior art. In some cases, the optimal threshold can be estimated using a statistical mechanical mapping. Loop sector references adapted from [20].

[1]There might be some technical differences between the two BP+OSD implementations. Furthermore we do not consider periodic boundary conditions and the small distances might lead to finite size effects playing a significant role.

2 Preliminaries

2.1 The Pauli Group and its Symplectic Structure

The n-qubit Pauli group $\mathcal{P}_n$ is defined as the set of n-fold tensor products of single-qubit Pauli operators with an additional phase ± 1 or $\pm i$:

$$\mathcal{P}_n := \left\{ i^c \cdot Q^{(1)} \otimes \cdots \otimes Q^{(n)} \,\middle|\, c \in \{0,1,2,3\}, Q^{(i)} \in \{\mathbb{1}, \sigma_X, \sigma_Y, \sigma_Z\} \right\} \tag{1}$$

where σ_X, σ_Y and σ_Z denote the three Pauli matrices. For $Q \in \mathcal{P}_n$ we denote its i-th component by $Q^{(i)} \in \{\mathbb{1}, \sigma_X, \sigma_Y, \sigma_Z\}$. Any Pauli operator can be represented (up to phase) by a pair of length-n binary bitstrings $u, v \in \mathbb{F}_2^n$ defined as follows:

$$u_i := \begin{cases} 1 & \text{if } Q^{(i)} \in \{\sigma_X, \sigma_Y\} \\ 0 & \text{else} \end{cases} \qquad v_i := \begin{cases} 1 & \text{if } Q^{(i)} \in \{\sigma_Z, \sigma_Y\} \\ 0 & \text{else} \end{cases} \tag{2}$$

The map $w : \mathcal{P}_n \to \mathbb{F}_2^{2n}$ taking Q to (u, v) is a group homomorphism in that $w(Q'Q) = w(Q') + w(Q)$.

The fact that any two Pauli operators either commute or anticommute allows us to define a symplectic form on $\mathcal{P}_n$, namely $\langle \cdot, \cdot \rangle : \mathcal{P}_n \times \mathcal{P}_n \to \mathbb{F}_2$ with

$$\langle Q, Q' \rangle := \begin{cases} 0 & \text{if } [Q, Q'] = 0, \\ 1 & \text{if } \{Q, Q'\} = 0. \end{cases} \tag{3}$$

Observe that $\langle Q, Q' \rangle = w(Q)^T J w(Q')$ for $J = \left(\begin{smallmatrix} 0 & \mathbb{1} \\ \mathbb{1} & 0 \end{smallmatrix}\right)$, when interpreting $w(Q)$ as a column vector, with $\mathbb{1}$ the n-dimensional identity matrix.

A symplectic basis for $\mathcal{P}_n$ is a set of $2n$ Pauli operators $X_1, \ldots, X_n, Z_1, \ldots, Z_n \in \mathcal{P}_n$ which fulfills the (anti-)commutation relations

$$\langle X_i, X_j \rangle = 0\,, \qquad \langle Z_i, Z_j \rangle = 0\,, \qquad \text{and} \qquad \langle X_i, Z_j \rangle = \delta_{ij}\,. \tag{4}$$

for all $i, j \in \{1, \ldots, n\}$. Given a symplectic basis, any Pauli operator $Q \in \mathcal{P}_n$ can be written as a product of the basis elements

$$Q = i^c \cdot X_1^{\lambda_1} \cdots X_n^{\lambda_n} \cdot Z_1^{\mu_1} \cdots Z_n^{\mu_n} \tag{5}$$

for some set of coefficients $c \in \{0,1,2,3\}, \lambda, \mu \in \mathbb{F}_2^{2n}$.

In many instances, the phase of a Pauli operator is irrelevant and to reflect this we define the *projective Pauli group* as

$$\mathcal{P}_n^* := \mathcal{P}_n / \{\mathbb{1}, -\mathbb{1}, i\mathbb{1}, -i\mathbb{1}\} \tag{6}$$

which represents the Pauli operators modulo phase. We denote the projection from $\mathcal{P}_n \to \mathcal{P}_n^*$ by π. For convenience, we will generally denote elements of $\mathcal{P}_n$ with upper-case letters and elements of $\mathcal{P}_n^*$ with lower-case letters. Any element $q \in \mathcal{P}_n^*$ can be written as $q = \pi(Q)$ for some $Q \in \mathcal{P}_n$, so by Equation (5) one can write

$$q = x_1^{\lambda_1} \cdots x_n^{\lambda_n} \cdot z_1^{\mu_1} \cdots z_n^{\mu_n} \tag{7}$$

where $x_i := \pi(X_i)$ and $z_i := \pi(Z_i)$. For a given symplectic basis, there is a one-to-one relation between the 4^n elements of $\mathcal{P}_n^*$ and the 4^n possible coefficients $(\lambda, \mu) \in \mathbb{F}_2^{2n}$.

2.2 Stabilizer Codes

Here we sketch the relevant aspects of stabilizer codes; for more details see [35]. An n-qubit stabilizer code is characterized by its *stabilizer group* $\mathcal{S}$, an Abelian subgroup of $\mathcal{P}_n$. The codespace is defined as the simultaneous $+1$ eigenspace of all $S \in \mathcal{S}$. For a stabilizer group generated by $n-k$ independent Pauli operators $S_1, \ldots, S_{n-k}$, the dimension of the codespace is 2^k, meaning it encodes k qubits.

When considering the action of a Pauli on a codeword, the phase of the operator is clearly physically irrelevant, so we will consider such a Pauli e to be $\in \mathcal{P}_n^*$. By the definition of the stabilizer group, any $e \in \mathcal{S}^*$, where $\mathcal{S}^* := \pi(\mathcal{S})$, will act trivially on any codeword. The information stored in the codespace is acted on by the logical operators of the code, the subgroup $\mathcal{L}^* := N(\mathcal{S}^*)/\mathcal{S}^* \subset \mathcal{P}_n^*$ defined as the normalizer of the stabilizer group in the Pauli group,

modulo the stabilizers themselves. Indeed, $\mathcal{L}^*$ consists of all Pauli operators not in $\mathcal{S}^*$ which commute with all its elements.

Another important related group are the *destabilizers* $\mathcal{D}^* := N(\mathcal{L}^*)/\mathcal{S}^* \subset \mathcal{P}_n^*$, those Pauli operators which commute with the logicals but which are not in the stabilizer group $\mathcal{S}^*$ [36]. Using a symplectic Gram-Schmidt procedure, any set of independent stabilizer generators can be extended to a symplectic basis in which $z_1, \ldots, z_k$ and $x_1, \ldots, x_k$ generate $\mathcal{L}^*$, $z_{k+1}, \ldots, z_n$ generate $\mathcal{S}^*$, and $x_{k+1}, \ldots, x_n$ generate $\mathcal{D}^*$ [37]. This symplectic basis extension of the stabilizer generators is sometimes called a *tableau* of the code. It should be noted that a tableau of a stabilizer code is not unique.

As a consequence, given some tableau of a stabilizer code, any Pauli operator $q \in \mathcal{P}_n^*$ can be uniquely decomposed in a stabilizer, destabilizer and logical part, i.e. $q = s \cdot d \cdot l$ where $s \in \mathcal{S}^*$, $d \in \mathcal{D}^*$, and $l \in \mathcal{L}^*$. More concretely, we can write $q = x_1^{\lambda_1} \cdot \cdots \cdot x_n^{\lambda_n} \cdot z_1^{\mu_1} \cdot \ldots z_n^{\mu_n}$ where $\lambda_i = \langle q, z_i \rangle$ and $\mu_i = \langle q, x_i \rangle$.

The general error correction procedure for a stabilizer code can be described as follows. First, the stabilizer generators $S_1, \ldots, S_{n-k}$ are measured, each one yielding a result $m_j \in \mathbb{F}_2$. Because the stabilizers commute, the order of the measurement does not matter. The entire collection of outcomes, $m \in \mathbb{F}_2^{n-k}$ is called the *syndrome*. The next step of the correction procedure is to decide on a suitable Pauli correction operation given the observed syndrome. This requires only classical computation, which we will call the *decoder*. Finally, the selected Pauli correction operation is applied to the data qubits.

2.3 Optimal Pauli Noise Decoding

The correction procedure succeeds if the product of the actual error $e \in \mathcal{P}_n^*$ and the correction operation $e' \in \mathcal{P}_n^*$ is a stabilizer: $e'e \in \mathcal{S}^*$. The fact that successful correction need not determine the actual error is due to the degeneracy of quantum stabilizer codes.

Decomposing $e = s \cdot d \cdot l$ and $e' = s' \cdot d' \cdot l'$, it is apparent that the stabilizer contribution is irrelevant. That is, $e'' = se'$ is just as good a correction operation as e' for any $s \in \mathcal{S}^*$. Meanwhile, the syndrome m specifies the destabilizer contribution as $d = x_{k+1}^{m_1} \cdot \cdots \cdot x_n^{m_{n-k}}$ since the s and l contributions commute with $\mathcal{S}^*$. Therefore to return to the codespace (with trivial syndrome), it is necessary that $d' = d$.

The remaining task of the decoder is to select an appropriate l' given the syndrome m. Suppose that the Pauli errors occur according to some distribution P_E over $\mathcal{P}_n^*$, so that the probability of an error e is $P_E(e)$. In view of the decomposition into stabilizer, destabilizer, and logical operator contributions, the distribution over errors may be thought of as a distribution over the random variables S, D, and L: $P_{S,D,L}(s, d, l) := P_E(s \cdot d \cdot l)$. The optimal decoding strategy then consists of selecting l' to be the most likely logical operator given the observed syndrome m. This may be written as

$$l' = \operatorname*{argmax}_{l \in \mathcal{L}^*} P_{L|D=d(m)}(l)\,, \tag{8}$$

where we have written the destabilizer d as a function of m since the latter determines the former. By Bayes' rule we have $P_{L|D=d(m)}(l) = P_{L,D}(l, d(m))/P_D(d(m))$, and therefore

$$l' = \operatorname*{argmax}_{l \in \mathcal{L}^*} \sum_{s \in \mathcal{S}^*} P_{S,D,L}(s, d(m), l) = \operatorname*{argmax}_{l \in \mathcal{L}^*} \sum_{s \in \mathcal{S}^*} P_E(s \cdot d(m) \cdot l)\,. \tag{9}$$

The optimal decoder is only concerned with the most-likely error class, namely Pauli operators $s \cdot d \cdot l'$ for any $s \in \mathcal{S}^*$. It is not concerned with the most-likely error, given by $s' \cdot d \cdot l'$ for $(s', l') = \operatorname{argmax}_{s \in \mathcal{S}^*, l \in \mathcal{L}^*} P_E(s \cdot m \cdot l)$.

2.4 Circuit-level Noise

When realizing error correction experiments in practice, one encounters the additional difficulty that the syndrome measurement itself is also affected by noise. This significantly complicates the decoding process on one hand because the syndrome measurement outcome cannot be trusted (and thus the destabilizer part of the error cannot be determined with certainty) and on the other hand because the syndrome readout circuit itself might generate and spread noise throughout the data qubits. The most common solution to address this issue for 2D topological codes is to repeat the syndrome measurement multiple times, which essentially increases the dimensionality of the decoding problem from 2D to 3D [18].

While the stabilizer formalism provides a means to describe the relationship between error events in the circuit and observed measurement outcomes, it can be computationally advantageous to make use of the slightly different formalism of *detector error models* (DEMs)[22]. In a DEM, noise in the circuit is modelled as a collection of *error*

mechanisms, where each "mechanism" is a a single Pauli operator occurring somewhere in the circuit. Furthermore, these errors are modelled as occurring independently of each other. For example, depolarizing noise can be represented in the framework of DEMs, as it can be seen as the concatenation of independent X, Y and Z error mechanisms, each occurring with identical probability:

$$\mathcal{D}_p = ((1-r_q)\cdot[\mathbb{1}] + r_q\cdot[\sigma_X]) \circ ((1-r_q)\cdot[\mathbb{1}] + r_q\cdot[\sigma_Y]) \circ ((1-r_q)\cdot[\mathbb{1}] + r_q\cdot[\sigma_Z]]) \tag{10}$$

where $\mathcal{D}_p(\rho) := (1-p)\rho + p\frac{\mathbb{1}}{2}$ is the depolarizing channel of rate p, $r_q := \frac{1-\sqrt{1-p}}{2}$, and $[U](\rho) := U\rho U^\dagger$ denotes the channel induced by the unitary U. Note that not all single-qubit Pauli channels can be represented as three independent X, Y and Z error mechanisms. Compared to the stabilizer formalism in which individual qubit Pauli errors are specified by an element of $\mathbb{F}_2^2$, e.g. X^uZ^v for $(u,v)\in\mathbb{F}_2$, the DEM framework uses an overcomplete representation $X^uY^vZ^w$ for $(u,v,w)\in\mathbb{F}_2^3$. Thus (u,v,w) and $(1-u,1-v,1-w)$ specify the same Pauli error.

Every error mechanism in a DEM causes certain *detectors* to flip, where detectors are parity checks of measurement outcomes that would be zero should no noise occurs in the circuit. Often the detectors are given by the parity of two-subsequent measurements of the same stabilizer. Typically, detectors should be chosen in such a way that an error mechanism only causes detectors to flip that are nearby in space and time. Similarly to detectors, an error mechanism can also cause a *logical observable* to flip. Analogously to detectors, these are parity checks that are supposed to take a deterministic value if no noise is present in the circuit. The main difference is that logical observables are not given as input to the decoding algorithm—instead the task of the decoder is to estimate their value.

Mathematically speaking, a DEM with n error mechanisms and m detectors is characterized by the triple (H,p,ℓ), with $H\in\mathbb{F}_2^{m\times n}$ a parity check matrix, $p\in\mathbb{R}^n$ a probability vector, and $\ell\in\mathbb{F}_2^n$ a logical error indicator vector. Here we have assumed a DEM with only a single logical observable, but the formalism can be extended to multiple logical operators. The entry $H_{i,j}\in\mathbb{F}_2$ is 1 if and only if the error mechanism j causes the detector i to flip and zero otherwise, while the ith component p_i of p is the probability that the ith error mechanism occurs, and the ith component ℓ_i of ℓ is 1 when the ith error mechanism causes the logical observable to flip and 0 otherwise.

It should be noted that multiple error mechanisms which cause the identical detectors to flip and have the same logical effect can be combined into one single error mechanism with combined probability. This optimization is always done in practice, as it can considerably reduce the number of error mechanisms. Indeed, this is one of the main advantages of the DEM formulation.

The optimal decoder for a detector error model is formulated as follows: Assume that the measurement outcome of the detectors is given by $m\in\mathbb{F}_2^m$. The joint probability of the detector outcome m occurring and the logical error being $L\in\mathbb{F}_2$ is given by

$$p_{m,L} = \sum_{\substack{x\in\mathbb{F}_2^n \\ Hx=m \text{ and} \\ \ell\cdot x=L}} P_X[x] \tag{11}$$

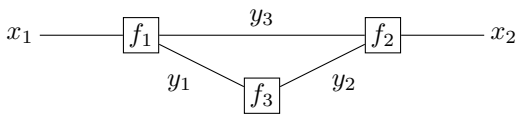
where

$$P_X[x] := (1-p_1)^{1-x_1}p_1^{x_1}\cdots(1-p_n)^{1-x_n}p_n^{x_n}\,. \tag{12}$$

The optimal logical correction L^* is then given by $L^*(m) = \operatorname*{argmax}_{L\in\mathbb{F}_2} p_{m,L}$ due to a similar reasoning as in Section 2.3.

2.5 Tensor Networks

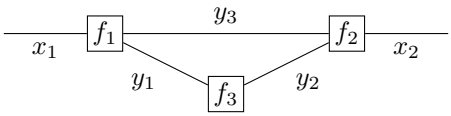
Here we give a brief overview; for more details see e.g. [38]. Fundamentally, tensor networks are a graphical representation of algebraic expressions which are sums of products. This graphical representation can be useful in designing algorithms for exact or approximate evaluation of the intended expression. Consider an expression of the form $g(x_1,x_2) = \sum_{y_1,y_2,y_3} f_1(x_1,y_1,y_3)f_2(x_2,y_2,y_3)f_3(y_1,y_2)$. We associate a vertex or node to each factor and to each variable external to the summation. Edges connect the appropriate nodes; there is one edge for each of the internal and external variables. The result is the following.



(13)

Our convention is edges terminating on a variable name fix the value of that edge, so the tensor network represents $g(x_1,x_2)$ for given values of x_1 and x_2. On the other hand, a variable name next to an edge is just a label. In this

way we can use the tensor network to represent the entire function g by using half-edges for free variables:



(14)

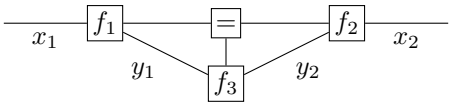
The example g is not the general case of a sum-of-products, as no variable appears in more than two factors. However, any general sum of products can be brought into this particular form by making use of additional internal variables and indicator functions to enforce equality. For instance, consider

$$g'(x_1, x_2) = \sum_{y_1, y_2, y_3} f_1(x_1, y_1, y_3) f_2(x_2, y_2, y_3) f_3(y_1, y_2, y_3), \tag{15}$$

where y_3 now appears in each factor. But this is just

$$g'(x_1, x_2) = \sum_{y_1, y_2, y_3, z_1, z_2} f_1(x_1, y_1, z_1) f_2(x_2, y_2, z_2)\, f_3(y_1, y_2, y_3) \delta(y_3, z_1, z_2) \tag{16}$$

for δ the function which is 1 when all its arguments are equal and zero otherwise. Now we apply the above construction. Denoting the δ factor by $=$, the following tensor network represents the function g':



(17)

3 Optimal Decoding as Tensor Network Contraction

In this section we describe two formulations of the quantity $\sum_{s \in \mathcal{S}^*} P_E(s \cdot d(m) \cdot l)$ as tensor network contraction. By Equation (9), this is sufficient to realize the optimal decoder; the error class probability can be calculated for each value of l and fixed m via contraction. We call the two formulations the *detector picture* and the *generator picture*.

Specifically, we will consider the case of independent single-qubit noise of the form

$$P_E(q) = P_1(q^{(1)}) \cdot P_2(q^{(2)}) \cdots P_n(q^{(n)}), \tag{18}$$

where the P_i for $i \in \{1, \ldots n\}$ are distributions over $\{\mathbb{1}, \sigma_X, \sigma_Y, \sigma_Z\}$. This will ensure that the tensor network inherits the topology of the stabilizer code, i.e. a 2D topological code results in a tensor network that is local in 2 dimensions and a 3D topological code results in a tensor network that is local in 3 dimensions. More generally, it would be sufficient to assume that the noise model is expressible as a Markov random field [4], but we will not pursue this issue further here.

Exact contraction of the resulting tensor networks will generally be infeasible, and therefore approximate contraction procedures are required in practice. These are further discussed in Section 4.

Furthermore, we also discuss how the generator and detector picture simplify for the special case of CSS codes under purely X-type and Z-type Pauli noise. Finally, we also briefly discuss how to extend the formalism to circuit-level noise, which is more relevant for experimental implementations of quantum error correction.

Both tensor network formulations begin with the expression

$$\sum_{s \in \mathcal{S}^*} P_E(s \cdot d(m) \cdot l) = \sum_{q \in \mathcal{P}_n^*} P_E(q)\, \Pi_{m,l}(q) \tag{19}$$

where $\Pi_{m,l}(q)$ is an indicator function enforcing that q should have syndrome m and logical component l. This is the basic sum-of-products expression that leads to a tensor network representation. It will be more convenient to represent each Pauli operator as an element of $\mathbb{F}_2^{2n}$ using the map w instead of working over $\mathcal{P}_n^*$. Abusing notation somewhat to write $P_E(y)$ with $y \in \mathbb{F}_2^{2n}$ for $P_E(q)$ with $y = w(q)$ and similarly for $\Pi_{m,l}$, we have

$$\sum_{s \in \mathcal{S}^*} P_E(s \cdot d(m) \cdot l) = \sum_{y \in \mathbb{F}_2^{2n}} P_E(y)\, \Pi_{m,l}(y)\,. \tag{20}$$

Two different ways of expressing the indicator function give rise to the two tensor network formulations. Both make use of the parity function $\mathbb{P}$, which is zero unless the sum (modulo 2) of its arguments is itself zero.

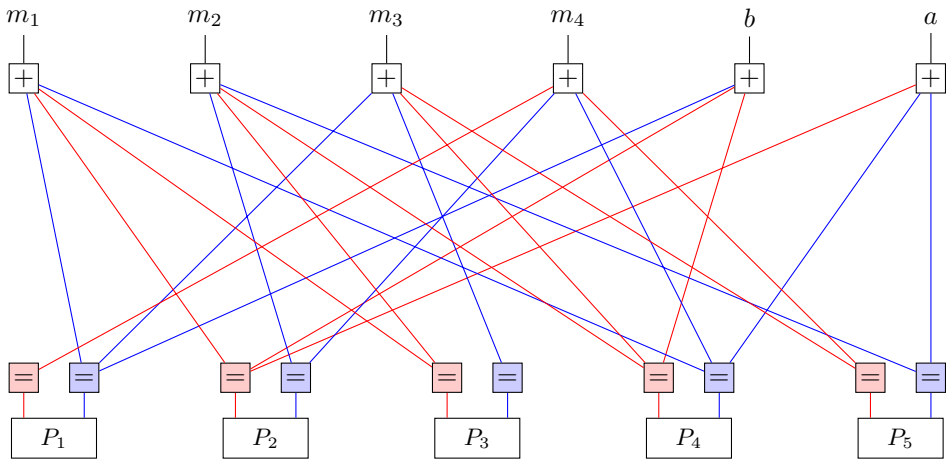


Figure 2: Detector picture tensor network of the 5-qubit code. In order to improve readability, the lines in the network are coloured red and blue to indicate connections to the X and Z tensors.

3.1 Detector Picture

3.1.1 Initial formulation

In the detector picture we start by using the parity function to enforce the particular syndrome and logical contribution. Working directly in the $\mathbb{F}_2^{2n}$ representation, the syndrome constraint is simply $y \cdot \bar{w}(s_i) = m_i$ for $i = 1, \ldots, n-k$, where we have defined $\bar{w}(q) = Jw(q)$ for any Pauli operator q and $s_1, \ldots, s_{n-k}$ denote the stabilizer generators. These constraints are captured by the indicator function $\prod_{i=1}^{n-k} \mathbb{P}(y \cdot \bar{w}(s_i), m_i)$. The constraint for the logical operator is similar. Writing l in terms of the generators x_j and z_j as $l = x_1^{a_1} \cdot x_2^{a_2} \cdots x_n^{a_n} \cdot z_1^{b_1} \cdot z_2^{b_2} \cdots z_n^{b_n}$ with $a, b \in \mathbb{F}_2^k$, we then have $\prod_{j=1}^{k} \mathbb{P}(y \cdot \bar{w}(x_j), b_j) \mathbb{P}(y \cdot \bar{w}(z_j), a_j)$. Note that commutation or anticommutation with x_j determines the z_j contribution to the Pauli operator, and vice versa. Thus we have

$$\Pi_{m,l}(q) = \prod_{i=1}^{n-k} \mathbb{P}(y \cdot \bar{w}(s_i), m_i) \prod_{j=1}^{k} \mathbb{P}(y \cdot \bar{w}(x_j), b_j)\, \mathbb{P}(y \cdot \bar{w}(z_j), a_j)\,. \tag{21}$$

Using this expression in Equation (20) gives a summation over product terms. For convenience we write the summed variable $y = (u, v)$ where $u \in \mathbb{F}_2^n$ is the X component and $v \in \mathbb{F}_2^n$ is the Z component of the error. Then for each qubit the TN includes a tensor node for the probability of error on that qubit. We label the node for the ith qubit P_i and it has two edges, corresponding to u_i and v_i, specifying the X and Z components of the Pauli error on that qubit, respectively.

For the parity factors, first observe that since $y \cdot \bar{w}(s_i)$ is just the sum of the components of $y = (u, v)$ for which $\bar{w}(s_i)$ is not zero, we may simply take these components as individual arguments to $\mathbb{P}$. Thus, each parity function is associated to a tensor node with edges given by the appropriate components of u and v, as well as a bit coming from the constraints, be it m_i, a_i, or b_i. Tensor nodes associated to parity functions are called check nodes and labelled by '+'.

Since any given u_i or v_i may participate in multiple parity constraints, to connect the output of a probability tensor P_j we make use of an equality tensor to copy the values of u_i and v_i. An equality node, labelled by '=', simply represents the function which is 1 if all of its arguments agree, and zero otherwise.

Figure 2 depicts the detector tensor network associated to the decoding problem of the 5-qubit code. The stabilizer generators are taken to be $s_1 = XZZXI$, $s_2 = IXZZX$, $s_3 = XIXZZ$, $s_4 = ZXIXZ$ and logical generators $x_1 = XZIZI$, $z_1 = IZIXX$. Edges terminating in variables are understood to take only the value of the indicated variable, i.e. an edge terminating in 'a' takes only the value a. The equality nodes are colored red and blue to distinguish u_i, or X contributions, in red from v_i, Z contributions, in blue.

To summarize, the detector picture tensor network is obtained as follows.

The tensor network is specified by the qubit probability distributions $\{P_j\}_{j=1}^{n}$, the syndrome $m \in \mathbb{F}_2^{n-k}$, the associated stabilizer generators $\{s_i \in \mathcal{S}^*\}_{i=1}^{n-k}$, the logical generators $\{x_j, z_j \in \mathcal{L}^*\}_{j=1}^{k}$, and a logical operator $l \in \mathcal{L}^*$ with $l = x_1^{a_1} \cdot x_2^{a_2} \cdots x_n^{a_n} \cdot z_1^{b_1} \cdot z_2^{b_2} \cdots z_n^{b_n}$ with $a, b \in \mathbb{F}_2^k$.

1. The ith qubit is represented by  with the red and blue equality nodes corresponding to the X and Z components respectively.
2. The jth stabilizer generator is represented by a check node with one edge connected to the syndrome value m_j.
3. The jth logical generator x_j (z_j) is represented by a check node with one edge connected to b_j (a_j).
4. A check node is connected to an equality node (of X or Z type) if and only if the corresponding qubit is involved in the corresponding stabilizer or logical generator. For instance, the jth stabilizer check is connected to u_i if $\bar{w}(s_j)_i = 1$ and v_i if $\bar{w}(s_j)_{i+n} = 1$. (Note the use of $\bar{w}$ here.)

3.1.2 Removing high-weight logical parity nodes

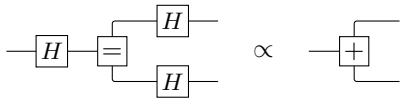
In the setting of topological codes, the check nodes corresponding to stabilizer generators are local and of constant degree. However, the check nodes associated to the logical generators, i.e. the factors $\mathbb{P}(y \cdot \bar{w}(x_j), b_j)$ and $\mathbb{P}(y \cdot \bar{w}(z_j), a_j)$ typically have neither of these properties. Their connectivity can cause difficulties in the approximate tensor contraction methods described in Section 4. These difficulties can however be circumvented by using the Walsh-Hadamard transform.

Recall that the decoder will need to evaluate 2^{2k} tensor networks, one for each value of $a, b \in \mathbb{F}_2^k$. That is to say, we are ultimately interested in the tensor network like that of Figure 2 but where the logical check nodes have open edges instead of these edges terminating on fixed values a and b. The network with open edges represents the collection of all the networks with fixed edges. Using the *Hadamard tensor* we can instead determine all the requisite error class probabilities by evaluating 2^{2k} simpler tensor networks and subsequently performing a Walsh-Hadamard transform of the resulting data.

The components of the degree-two Hadamard tensor on binary-valued edges are determined by the 2×2 Hadamard matrix. Observe that contracting a separate Hadamard tensor to all the edges of an equality tensor results in a tensor proportional to the parity tensor. Algebraically, say for the degree-three case, this is the statement that

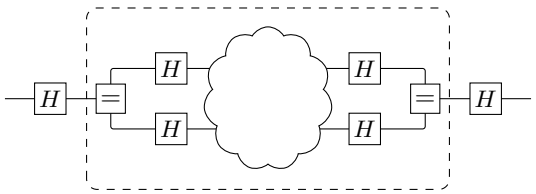
$$\sum_{x',y',z' \in \mathbb{F}_2} H_{xx'} H_{yy'} H_{zz'} \, \delta_{x'y'} \, \delta_{y'z'} \;\propto\; \mathbb{P}(x, y, z)\,. \tag{22}$$

In tensor network notation this is just



(23)

This equivalence allows us to replace the $2k$ parity check nodes associated with the logical generators with equality nodes and degree-two Hadamard nodes. For instance, if there are two logical parity check nodes, the general situation is given by



(24)

where the cloud represents the remainder of the network. Instead of determining the values of the total tensor network, one may instead evaluate network inside the dashed box. In this example, there are four such networks to be evaluated, and in general for $2k$ parity nodes there will be 2^{2k} relevant networks. Regarding this contraction

data as a vector of 2^{2k} entries, the final tensor network contraction with the "outbound" Hadamard nodes is just the Walsh-Hadamard transform of the data.

This approach fixes the problem of the non-local nodes, as the remaining high-weight equality node in the dashed network can be replaced with equality nodes on each involved site separately, given that the value of the external leg is fixed.

3.2 Generator Picture

The generator picture takes a different approach to handling the dependence on the logical operator l. Returning to Equation (20), the required terms in the sum correspond to summing over all the Pauli operators in the set $F_{m,l} = \{s \cdot x_{k+1}^{m_1} \cdots x_n^{m_{n-k}} \cdot l : s \in \mathcal{S}^*\}$ (a coset of the stabilizer group). Equivalently, for any $r_{m,l}$ in $F_{m,l}$, i.e. any Pauli operator with syndrome m and logical component l, $F_{m,l}$ can equally-well be specified by $\{sr_{m,l} : s \in \mathcal{S}^*\}$. In terms of the $\mathbb{F}_2^{2n}$ representation, $F_{m,l}$ is specified by $\{w(r_{m,l}) + \sum_{i=1}^{n-k} \lambda_i w(s_i) : \lambda \in \mathbb{F}_2^{n-k}\}$ where $s_1, \ldots, s_{n-k}$ denote the stabilizer generators. The indicator function $\Pi_{m,l}(y)$ enforces that y is an element of this set. Therefore, in terms of the parity function $\mathbb{P}$, it holds that

$$\Pi_{m,l}(y) = \sum_{\lambda \in \mathbb{F}_2^{n-k}} \mathbb{P}(y, w(r_{m,l}), \lambda_1 w(s_1), \ldots, \lambda_{n-k} w(s_{n-k}))\,. \tag{25}$$

Therefore, we now have the expression

$$\sum_{y \in \mathbb{F}_2^{2n}} P_E(y)\, \Pi_{m,l}(y) = \sum_{y \in \mathbb{F}_2^{2n}} \sum_{\lambda \in \mathbb{F}_2^{n-k}} P_E(y)\, \mathbb{P}(y, w(r_{m,l}), \lambda_1 w(s_1), \ldots, \lambda_{n-k} w(s_{n-k}))\,, \tag{26}$$

which is also in sum-of-products form, and hence amenable to tensor network representation. Here we are using a slightly different parity function than in the previous section; in Equation (26) the arguments are elements of $\mathbb{F}_2^{2n}$, as opposed to single bits. However, $\mathbb{P}(y, y')$ for $y = (u, v)$ and $y' = (u', v')$ both in $\mathbb{F}_2^{2n}$ can be decomposed into $\prod_{j=1}^{n} \mathbb{P}(u_j, u'_j)\mathbb{P}(v_j, v'_j)$.

The resulting algebraic expression for $\sum_{y \in \mathbb{F}_2^{2n}} P_E(y)\, \Pi_{m,l}(y)$ is rather unwieldy, so let us proceed directly to the tensor network description. In contrast to the detector picture, now there is a parity check for each u_j and for each v_j with edges heading to three different places. The u_j and v_j checks are each connected to the associated probability tensor P_j (again assuming the distribution P_E is independent). Another set of edges connects the u_j and v_j checks to the appropriate components of $w(r_{m,l})$; defining $w(r_{m,l}) = (r^x, r^z)$ with r^x and r^z elements of $\mathbb{F}_2^n$, the u_j node is connected to the value r_j^x and the v_j node to the value r_j^z. Finally, since the λ_j participate in many parity checks, there is an equality node for each λ_j, connected to those u_i and v_i for which $w(s_j)_i = 1$ and $w(s_j)_{i+n} = 1$, respectively.

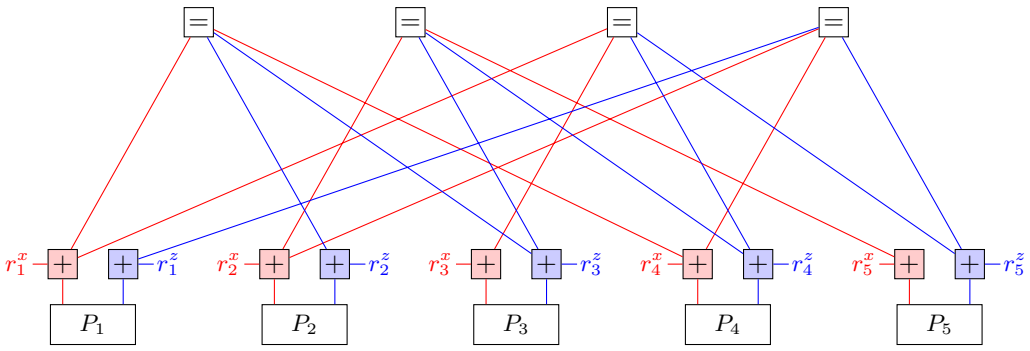


Figure 3: Generator picture tensor network of the 5-qubit code. As before, the lines in the network are coloured to improve readability.

Figure 3 depicts the resulting tensor network for the 5-qubit code, where we have chosen $r_{m,l}$ using only the logical operators and the destabilizer as suggested above. Observe that the values of $r_{m,l}$ could be locally incorporated into the check node tensors.

To summarize, the generator picture tensor network is obtained as follows.

The tensor network is specified by the qubit probability distributions $\{P_j\}_{j=1}^n$, a representative Pauli operator r satisfying the syndrome $m \in \mathbb{F}_2^{n-k}$ and logical contribution l, and the stabilizer generators $\{s_i \in \mathcal{S}^*\}_{i=1}^{n-k}$.

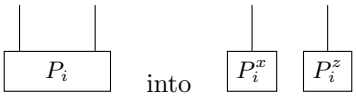
1. The ith qubit is represented by  with the red and blue parity nodes corresponding to the X and Z components respectively.
2. Parity node u_i is connected to the value $w(r)_i$ and parity node v_j to $w(r)_{i+n}$.
3. Each stabilizer generator s_j is represented by an equality node.
4. Equality node s_j is connected to parity node u_i when $w(s_j)_i = 1$ and to parity node v_j when $w(s_j)_{i+n} = 1$.

3.3 Bit- and phase-flip Noise on CSS Codes

Consider the special case where the considered code is CSS, i.e. all stabilizer generators are either purely X-type or Z-type. We will assume that the logical generators are chosen such that they are also all X-type or Z-type. Furthermore, assume that the noise model is given by independent X and Z errors on every qubit

$$P_i(\mathbb{1}) = (1 - p_i^x)(1 - p_i^z) \tag{27a}$$
$$P_i(\sigma_X) = p_i^x(1 - p_i^z) \tag{27b}$$
$$P_i(\sigma_Z) = (1 - p_i^x)p_i^z \tag{27c}$$
$$P_i(\sigma_Y) = p_i^x p_i^z \tag{27d}$$

for some $p_i^x, p_i^z \in [0, 1]$. In terms of the tensor representation of the distribution of noise on a single qubit, this structure means that we can separate



where P_i^x and P_i^z represent the vectors $(1 - p_i^x, p_i^x)$ and $(1 - p_i^z, p_i^z)$.

Under these assumptions the tensor network (both in the detector and generator pictures) separates into two disjoint sub-networks. One network is responsible for the decoding decision for Z-type logical operators (i.e. the problem of decoding bit-flip errors) and the other is responsible for the decoding decision for X-type logicals. To see this, notice that in the detector picture, the parity checks only act on one of the two sub-networks. Similarly, the logical component of the representative chosen in the generator picture can be separated into X-type and Z-type logicals, which again only impact the respective sub-network.

In the case when either all p_i^x are zero or all p_i^z are zero, one of the two sub-networks trivially contracts to 1 and can be omitted, meaning that only the other sub-network remains. More concretely, only one of the two stabilizer types (either X or Z type) remains represented as nodes in the tensor network. Interestingly, depending on whether one is in the detector or generator picture, the remaining type is different. For instance, when one considers pure bit-flip noise on a CSS code, then the generator picture involves the Z-type stabilizers, whereas the generator picture involves the X-type stabilizers. If we incorporate the remaining probability tensors into the neighboring = or + nodes, the remaining graph is a bipartite graph with one subgraph containing = nodes and the other + nodes. In the detector picture under bit-flip noise, the connectivity of the bipartite graph is described by the Z-type parity check matrix matrix H_Z of the CSS codes, whereas in the generator picture under bit-flip noise the connectivity is described by a dual matrix G_Z which fulfills $H_Z \cdot G_Z^T = 0$. This formalizes the notion of the generator picture being the dual of the detector picture.

As an illustrative example, Figures 4c and 4d depict the detector and generator tensor networks for the $d = 3$ unrotated 2D surface code under bit-flip noise. Only half of the plaquettes are involved in the diagram, in contrary to the detector and generator networks for general single-qubit i.i.d. noise, which are depicted in Figures 4a and 4b.

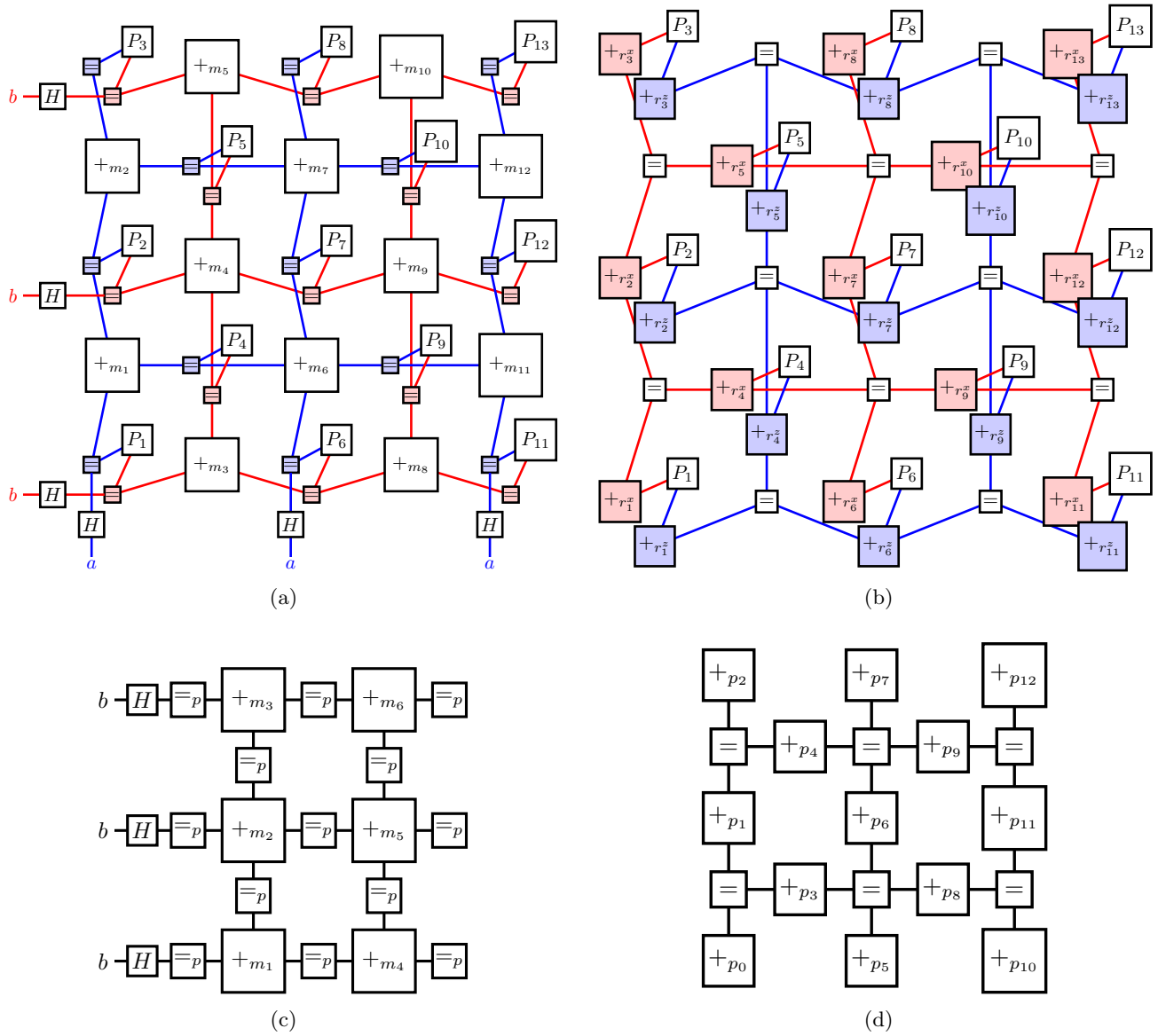


Figure 4: Detector (a) and generator (b) tensor networks for d=3 unrotated surface code for an arbitrary i.i.d. single-qubit Pauli noise model with marginals P_i. Detector (c) and generator (d) tensor networks for d=3 unrotated surface code under bit-flip noise of strength p. Here $=_q$ is the tensor that takes value $(1-q)$ if all incoming legs have value 1, value q if all incoming legs have value 0, and value 0 otherwise. Also, $+_q$ is the tensor that takes value $(1-q)$ if the incoming legs sum up to 0 (mod 2) and q otherwise. In (d) the p_i are defined as p if the i-th entry r_i^x of the representative bit-flip error is 0, and $1-p$ otherwise. The detector tensor networks in (a) and (c) implement the Walsh-Hadamard transform discussed in Section 3.1.2.

3.4 Detector Error Models for Circuit-level Noise

Mathematically speaking, the optimal decoding problem for a DEM is equivalent to the optimal decoding of one sector of a CSS code, with the corresponding parity check matrix given by the DEM. The error mechanism play the role of the qubits and the detectors play the role of the stabilizers generators. Thus, the detector and generator tensor network constructions are directly applicable. The resulting tensor network will be local under the assumption that all error mechanisms in the model are local, i.e. only trigger nearby detectors. In circuit-level noise, this is usually achieved by defining detectors to be the parity of two subsequent measurements of the same stabilizer generator.

4 Decoding by Approximate Contraction

The previous section laid out in detail how the optimal decoding process can be expressed in terms of a sequence of contractions of tensor networks that exhibit a topology respecting the locality of the underlying code. The exact contraction of these tensor networks for larger codes is generally not feasible, so approximate contraction schemes must be used for a practical decoder. While generally NP-hard, the approximate contraction of planar two-dimensional tensor network is theoretically well understood and tends to behave very well when used in practice. In the context of decoding of 2D Pauli codes, previous work by one of us demonstrated a powerful approximate contraction scheme that proved extremely successful in decoding a wide variety of two-dimensional stabilizer codes and effectively achieved the optimal threshold in all cases [8].

Generally, three-dimensional approximate contraction is significantly less well-behaved than its two-dimensional counterpart. In essence, the problem lies in the impossibility to define a canonical gauge for a tensor network that is not a tree [11, §5.2]. Still, there are some techniques that go beyond naïve time-evolving block decimation (TEBD) to deal with three-dimensional networks, with a certain amount of success [13–15, 17, 39–42]. These techniques are typically studied in the context of real or imaginary time evolution of two-dimensional condensed-matter systems, but they can also be applied in the setting of quantum error correction. They incur different trade-offs between accuracy and speed.

The technique we present works as follows: We assume that the 3D tensor network can be written as a stack of layers where each layer has the identical topology and connections between layers may only occur between equivalent sites. While the detector/generator tensor networks of most codes do not directly fulfill this assumption, we can slightly adapt the networks to make it hold (see Section 5 for details). We sweep a 2D tensor network from bottom to top in a TEBD-style fashion, contracting in one layer after the other while continually truncating the 2D tensor network bonds. For example, if the 3D tensor network is a cubic lattice, then we would sweep a PEPS from one side of the cube to the opposing side. The remaining two-dimension network is then contracted by sweeping an MPS across it (in case it is not a square lattice, the sweep-line-based contraction algorithm from [8] can be used).

A layer is contracted into the 2D tensor network by decomposing it into a sequence of two-qubit gates which can then be contracted one at a time. For some tensor networks (when the involved tensors are weighted delta or check nodes for instance) this can be done in a natural way—this is the case for the 3D surface code under bit-flip or phase-flip noise, as will be discussed in the next section. If no structure is known for doing the decomposition, one can instead use the singular value decomposition to separate a two-qubit gate from the layer, as depicted in figure Figure 5.

For the purpose of truncating the bond dimensions of the 2D tensor network we make use of the *simple update* technique [13–15]. The idea here is to keep track of a rank-1 approximation of the environment for each tensor in order to allow for a more accurate truncation. This means that on each bond, we keep track of a diagonal matrix. This method was recently shown to be sufficient for fast and highly accurate simulations of 2D quantum systems with more than a thousand qubits [16]. Its theoretical justification lies in the fact that it makes the loopy network approach the *Vidal gauge* when the applied gates are close enough to the identity [17].

5 Results

5.1 Point Sector of 3D Surface Code

Following the construction described in Section 3.3, one can straightforwardly produce a detector and generator tensor network. Consider the detector tensor network depicted in Figure 6a: it is almost of a cubic tensor network form, except for some additional tensors on the bonds between the cubic lattice sites. In contrary, the detector picture is less favorable and cannot be embedded as easily in a cubic lattice of the same size. For this reason, we make use of the detector picture here.

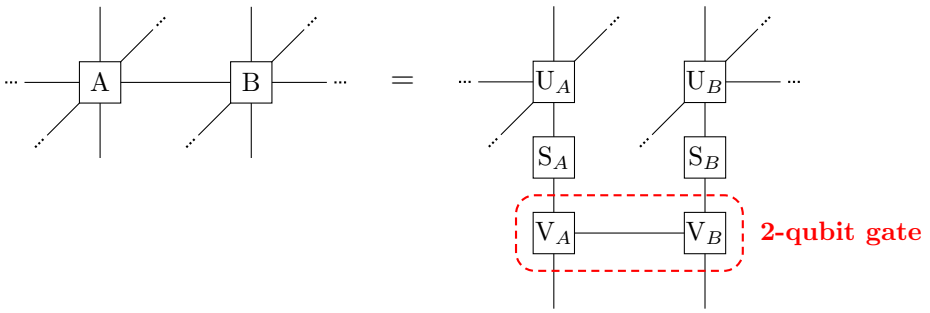
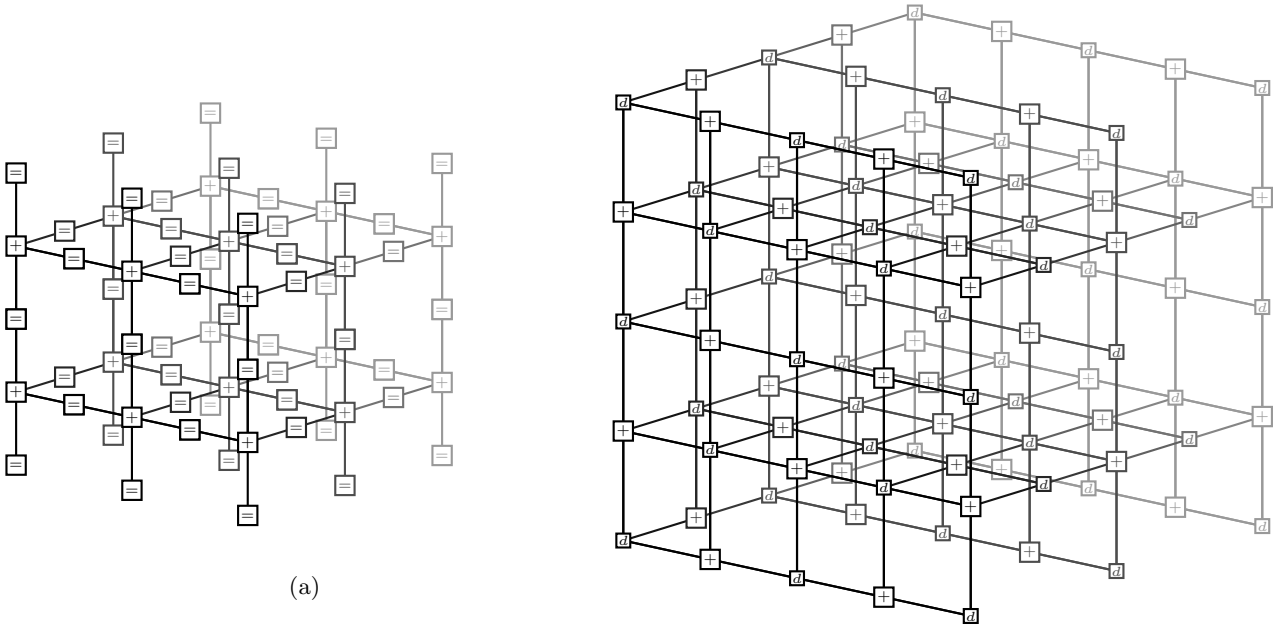


Figure 5: By iteratively repeating the depicted procedure, a layer from the 3D tensor network can be split into a sequence of two-qubit gates. The decomposition of A into U_A, S_A and V_A (respectively B into U_B, S_B and V_B) is done via the singular value decomposition. The contraction of V_A and V_B form the 2-qubit gate, while the contraction of U_A and S_A, respectively U_B and S_B replace A and B in the layer. For practical applications, the number of nonzero eigenvalues in S_A and S_B are truncated to a maximal value to avoid encountering tensors that are too large.



(a)

(b)

Figure 6: (a) Detector tensor network for the point sector of the 3D surface code of distance $d = 3$. The $=$ and $+$ tensors are weighted (not displayed in figure), since the probability tensors, Hadamard tensors and syndrome values are incorporated in them. (b) Detector picture tensor network for the distance-3 3D surface code subject to depolarizing noise. The d tensors represent the contraction of the single-qubit marginals P_i, the two connected $=$ tensors as well as the Hadamard gate for the Hadamard trick (where applicable). The syndrome values are incorporated in the $+$ nodes.

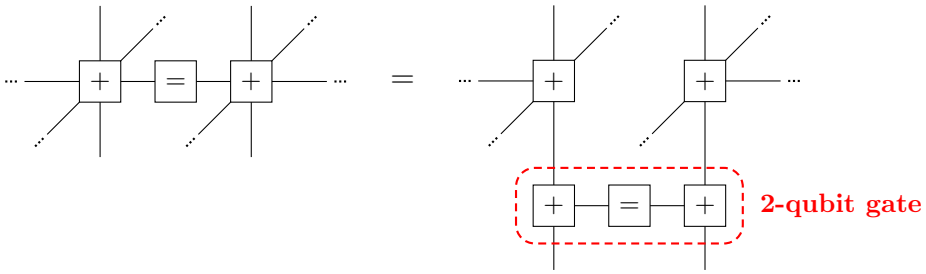


Figure 7: Splitting off of two-qubit gates from a layer of the 3D tensor network for the point sector 3D surface code contraction.

The tensors on the bonds between two cubic lattice sites require a small modification in the contraction algorithm. The tensors on the horizontal bonds can be integrated into the two-qubit gates of the simple update contraction procedure, whereas the tensors on vertical bonds can be exactly contracted in between layers. Since the tensors on the sites are either + or = nodes, the splitting up of the layer into two-qubit gates can be done very naturally as depicted in Figure 7.

For our numerical experiments, we use a maximum bond dimension of 24 for the simple update contraction procedure and a maximum bond dimension of 32 for the MPS contraction of the final 2D tensor network. The results are depicted in Figure 1a. Also depicted are results for the minimum-weight perfect matching decoder, which finds the most likely error and is thus the optimal decoder up to degeneracy. The matching decoder is the previous state-of-the-art decoder for this problem [19], and the optimal threshold for the point sector is known to be around 3.3% [26]. We used PyMatching2 [24] for the matching implementation.

5.2 Loop Sector of 3D Surface Code

By the considerations in Section 3.3, the detector network for the point sector (see Figure 6a) and the generator network for the loop sector of the 3D surface code are essentially the same, except the + and = nodes are interchanged. So due to the same reasoning as in the point sector, we choose to use the generator tensor network for the loop sector as it fits more naturally in a cubic form. Since the tensor network is topologically the same as the one of the point sector, we use the analogous contraction technique: The gate splitting is identical as in Figure 7, except that the = and + nodes are interchanged.

For our numerical experiments, we use a maximum bond dimension of 24 for the simple update contraction procedure and a maximum bond dimension of 48 for the MPS contraction of the final 2D tensor network. To our knowledge, the state-of-the-art decoding algorithm on the loop sector of the 3D surface code is BP+OSD as reported by Huang *et al.* [20], though they consider periodic boundary conditions. They report a threshold of around 21.55%, while the optimal threshold is known [29, 30] to be roughly 23.180%. Our numerical results for the TN decoder and BP+OSD (both on non-periodic boundary conditions) are depicted in Figure 1b. We employ Roffe *et al.*'s BP+OSD implementation [43, 44] using the min-sum algorithm and combination-sweep OSD with an order of 60.

5.3 Depolarizing Noise on 3D Surface Code

For depolarizing noise we make use of the detector tensor network. This means that now both the point-like and loop-like stabilizers are represented by nodes in the network. This incurs an additional difficulty: In the standard description of the 3D surface code, the loop-like stabilizer generators are redundant. Therefore, one must remove a subset of the loop stabilizers to obtain a linearly independent generator set. For this purpose we remove all the stabilizers on the 'top' of a cell, other than those on one (smooth) boundary. The tensor network for $d = 3$ is depicted in Figure 6b. Notice that compared to the point or loop sector, the lattice size of the resulting cubic tensor network is doubled, or respectively the volume is increased eight-fold. Furthermore, not all vertices and edges of the cubic lattice are occupied. In order to bring the tensor network to a truly cubic form amenable to our contracition algorithm, we fill up these empty slots with trivial tensors (having value 1) and trivial bonds of dimension 1.

We use a maximum bond dimension of 20 for the simple update contraction procedure and a maximum bond dimension of 64 for the MPS contraction of the final 2D tensor network. The maximum bond dimension of the

singular value decomposition of the splitting technique (see Figure 5) is chosen to be 4.

In Figure 1c we depict the numerical results comparing our TN decoder with the current state-of-the-art decoder, which to our knowledge is BP+OSD which has been reported to have a threshold of $5.95 \pm 0.03\%$[21]. Note that our BP+OSD results seem to perform significantly better than the threshold indicated by these authors—this could however just be boundary effects that disappear at larger distances. We use Roffe *et al.*'s BP+OSD implementation [43, 44] using the min-sum algorithm and combination-sweep OSD with an order of 60. As in [21], phase-flips are decoded assuming i.i.d. phase-flip noise first (with the noise rate given by the marginal of depolarizing noise), and subsequently bit-flips are decoded, using the conditional distribution of bit-flips given the phase-flip pattern from the first step.

We also attempted to realize a TN decoder using the generator tensor network, but it performed significantly worse than the detector TN decoder.

5.4 Circuit-level Noise

We study the rotated surface code under circuit level depolarizing noise. More concretely, for a distance d code we study the protocol consisting of d repeated stabilizer measurement rounds. The first X-stabilizer measurement round serves as an initialization of the logical qubit in the X basis, and at the very end of the protocol all data qubits are measured in the X basis to realize a logical X measurement. The goal of the decoder is to error-correct the outcome of said logical X measurement. After each gate, after each reset and before every measurement, we assume that the involved qubits undergo depolarizing noise (either 1-qubit or 2-qubit depolarizing noise accordingly) of some strength p. We use the Stim package [22] to generate the DEM model, so we refer the readers to there for more information about the precise details about the considered circuit.

We use the detector picture to represent the decoding problem. Therefore, following the discussion in Section 3.4, the ith error mechanism of the DEM is represented by a weighted equality node in the tensor network, i.e. a tensor that takes value $(1-p_i)$ if all incoming legs are 0, value p_i if all incoming legs are 1 and value 0 elsewise. The ith detector is represented by a check tensor that takes the value $(1-m_i)$ if the parity of the incoming legs is 0 and value m_i elsewise. An equality and check node are connected if and only if the involved error mechanism causes said detector to flip. We use the Walsh-Hadamard transform discussed in Section 3.1.2 to realize the logical parity check. For example, Figure 8 depicts the circuit-level tensor network for the $d = 3$ rotated surface code. A depiction of the $d = 5$ tensor network can be found at the end of the manuscript in Figure 10. Clearly, the resulting tensor network is not even remotely close to a cubic structure: Some of the check nodes have a very high degree, which is caused by the high number of distinct error mechanisms that can cause said detector to flip. Furthermore, the size of the tensor network is much higher than the non-circuit-level noise counterparts: The $d = 3$, $d = 5$ and $d = 7$ tensor networks contain 245, 1799 and 6351 tensors respectively.

Figure 8: Circuit-level noise tensor network derived from detector error model for the $d = 3$ rotated surface code with 3 rounds of measurements. They weighting of the check and equality nodes is not depicted. The Hadamard nodes from the Walsh-Hadamard transform are also not depicted.

Clearly, the issues laid out above make it impossible to directly contract the circuit-level tensor network directly. In fact, just naïvely storing the tensors would require an infeasible amount of memory due to the high tensor degrees. In order to address these problems, we propose the following scheme, in which the complicated DEM-derived tensor network is compressed and brought into a cubic form more amenable to our contraction scheme.

First, only consider the check nodes (corresponding to the detectors) and arrange them in a cubic lattice according to their location in space-time. Then, add the equality nodes (corresponding to the error mechanisms) one-by-one into this tensor network. If this step was done without approximation, then the resulting tensor network would clearly be equivalent to the desired circuit-level tensor network. To bring the equality node into something that respects the cubic lattice structure, we "snake" it along the lattice as depicted in Figure 9. Clearly, this will cause the bond dimensions of the cubic tensor network to grow very quickly, so we use the simple update method to truncate the bonds to some maximum bond dimension.

During this compression step we keep an open edge on each site to represent the detector measurement outcome. When a sample is to be decoded, we contract the corresponding observed detector values on each site of the cubic tensor network. The resulting 3D tensor network can then be contracted with our technique described in Section 4. Thanks to this trick, the compression step needs only to be performed *once* offline and not every time a new sample is to be decoded. This largely alleviates the cost of this compression step and allows us to perform it with much larger bond dimensions.

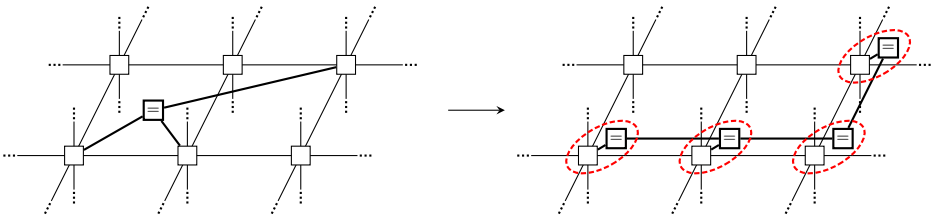


Figure 9: "Snaking" of a weighted equality node corresponding to an error mechanism along the cubic TN. The red dashed lines depict the updated tensors of the cubic TN. Clearly, the dimensions of the bonds involved in the snaking are doubled.

Figure 1d depicts the numerical results of our experiments comparing the tensor network decoder to the commonly used minimum-weight perfect matching decoder, implemented using PyMatching [23, 24]. For the offline pre-compression of the circuit-level tensor network, we use a maximum bond dimension of 16. Technically, if the physical error rate p is varied, one would have to generate a separate pre-compressed tensor network for every value of p. For convenience, we forgo this step and only generate one pre-compressed tensor network for a value of $p = 1\%$. Before being used for decoding (but after all error mechanisms have been compressed), we further truncate all bonds to a dimension of 8 in order to improve the speed of the decoder. To contract the resulting 3D cubic tensor network, we use a maximum bond dimension of 12 for the simple update, a maximum bond dimension of 64 for the MPS contraction and maximum bond dimension of 12 for the splitting process. For $d = 7$, we increase the simple update bond dimension to 20 and the splitting bond dimension to 14.

6 Discussion

Decoding 3D codes is a notoriously difficult problem, especially when the code does not allow for a matching-like decoding algorithm. We propose a new ML-approximating decoding scheme that outperforms the current state-of-the-art algorithm on the point sector, loop sector, and depolarizing noise for 3D surface codes of moderate size. Once the size of the code becomes too large, typically around linear size (code distance) $d = 11$, our 3D tensor network decoder starts facing numerical issues, which degrades the accuracy. This is in stark contrast to the 2D tensor network decoder (based on MPS truncation) which "out of the box" is both fast and reliable even for large codes.

This can be most likely attributed to the lack of a canonical form which cannot be defined for a PEPS in contrast to an MPS, making it difficult to realize bond truncations accurately. There are many techniques and heuristics for dealing with this issue. While we roughly explored several of them during the development of this paper, we settled for the simple update technique, since in our tests it provided the best accuracy-to-speed ratio. We experimented with bringing the PEPS into the Vidal gauge before truncation, either with belief propagation or with repeated trivial simple update steps, but the convergence of the gauging procedure was very slow and made up for the vast majority of the contraction time. We also investigated exactly contracting a complete layer at once before truncating the resulting bonds, but again the accuracy-time trade-off was not favorable.

The space of possibilities to realize approximate 3D tensor network contraction is much larger than in 2D, and is still an active field of research. Certainly the possibilities greatly exceed our limited exploration. We therefore expect that future work will be able to improve upon our results.

The 3D surface code can in some sense be considered one of the simplest 3D codes to decode, since the tensor network structure is naturally (almost) cubic. To show that our 3D contraction algorithm does not necessarily rely on a code that induces such a nicely structured tensor network, we also demonstrate our technique for decoding circuit-level noise, which arguably exhibits the most complicated structure encountered in practice. We show that our technique outperforms the matching decoder, though we did not compare it to the state-of-the-art algorithm belief-matching, which currently does not have a publicly available implementation. Our technique is not very optimized, but rather serves the purpose of demonstrating that the general TN decoding method works in principle. We fully expect that more careful and tedious tuning of the decoder parameters could significantly improve the performance and/or accuracy of the TN decoder on circuit-level noise. Similarly, we spent very little time with trying out different snaking and truncation procedures for the tensor network pre-compression step. Further research in optimizing the decoder and comparing it with other circuit-level decoders is necessary. The possible impact for

near-term quantum error correction experiments is very high, since the involved distances are rather small and the decoding can typically be performed offline. So a very accurate-but-slow tensor network decoder could facilitate the demonstration of break-even error correction.

Another point that requires further investigation is whether the detector or generator picture performs better in practice. While we did not test this systematically, we did notice that the detector tensor network seems much easier to contract for depolarizing noise compared to the generator tensor network. Also for the circuit-level noise one could make use of the generator picture tensor network, though this would require first finding a dual parity check matrix to the detector error model.

Acknowledgements

This work was supported by the ETH Quantum Center, the National Centres for Competence in Research in Quantum Science and Technology (QSIT) and The Mathematics of Physics (SwissMAP), and the Swiss National Science Foundation Sinergia grant CRSII5_186364. The authors are grateful to Michael Vasmer for useful discussions. We extensively used the panqec library[45] and thank Eric Huang and Arthur Pesah for their insights.

References

[1] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory", Physical Review A **52**, R2493 (1995) (page 1).

[2] A. M. Steane, "Error Correcting Codes in Quantum Theory", Physical Review Letters **77**, 793 (1996) (page 1).

[3] D. Gottesman, "Stabilizer Codes and Quantum Error Correction", (California Institute of Technology, May 28, 1997), 114 pp., arXiv:quant-ph/9705052 (page 1).

[4] C. T. Chubb and S. T. Flammia, "Statistical mechanical models for quantum codes with correlated noise", Annales de l'Institut Henri Poincaré D **8**, 269–321 (2021), arXiv:1809.10704 [quant-ph] (pages 1, 2, 7).

[5] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons", Annals of Physics **303**, 2–30 (2003), arXiv:quant-ph/9707021 (page 1).

[6] H. Bombin and M. A. Martin-Delgado, "Topological Quantum Distillation", Physical Review Letters **97**, 180501 (2006), arXiv:quant-ph/0605138 (page 1).

[7] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code", Physical Review A **90**, 032326 (2014), arXiv:1405.4883 [quant-ph] (pages 1, 2).

[8] C. T. Chubb, "General tensor network decoding of 2D Pauli codes", Oct. 13, 2021, arXiv:2101.04125 [quant-ph] (pages 1, 2, 13).

[9] A. J. Ferris and D. Poulin, "Tensor Networks and Quantum Error Correction", Physical Review Letters **113**, 030501 (2014), arXiv:1312.4578 [quant-ph] (page 1).

[10] A. S. Darmawan, Y. Nakata, S. Tamiya, and H. Yamasaki, "Low-depth random Clifford circuits for quantum coding against Pauli noise using a tensor-network decoder", Dec. 9, 2022, arXiv:2212.05071 [quant-ph] (page 1).

[11] R. Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states", Annals of Physics **349**, 117–158 (2014), arXiv:1306.2164 [cond-mat.str-el] (pages 1, 13).

[12] N. Delfosse, P. Iyer, and D. Poulin, "A linear-time benchmarking tool for generalized surface codes", Nov. 14, 2016, arXiv:1611.04256 [quant-ph] (page 2).

[13] S. S. Jahromi and R. Orus, "A universal tensor network algorithm for any infinite lattice", Physical Review B **99**, 195105 (2019), arXiv:1808.00680 [cond-mat.str-el] (pages 2, 13).

[14] P. Corboz, R. Orus, B. Bauer, and G. Vidal, "Simulation of strongly correlated fermions in two spatial dimensions with fermionic Projected Entangled-Pair States", Physical Review B **81**, 165104 (2010), arXiv:0912.0646 [cond-mat.str-el] (pages 2, 13).

[15] H. C. Jiang, Z. Y. Weng, and T. Xiang, "Accurate determination of tensor network state of quantum lattice models in two dimensions", Physical Review Letters **101**, 090603 (2008), arXiv:0806.3719 [cond-mat.str-el] (pages 2, 13).

[16] S. Patra, S. S. Jahromi, S. Singh, and R. Orus, *Efficient tensor network simulation of IBM's largest quantum processors*, 2023, arXiv:2309.15642 [quant-ph] (pages 2, 13).

[17] J. Tindall and M. Fishman, "Gauging tensor networks with belief propagation", Aug. 18, 2023, arXiv:2306.17837 [quant-ph] (pages 2, 13).

[18] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory", Journal of Mathematical Physics **43**, 4452–4505 (2002), arXiv:quant-ph/0110143 (pages 2, 5).

[19] C. Wang, J. Harrington, and J. Preskill, "Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory", Annals of Physics **303**, 31–58 (2003), arXiv:quant-ph/0207088 (pages 2, 3, 15).

[20] A. O. Quintavalle, M. Vasmer, J. Roffe, and E. T. Campbell, "Single-shot error correction of three-dimensional homological product codes", PRX Quantum **2**, 020340 (2021), arXiv:2009.11790 [quant-ph] (pages 2, 3, 15).

[21] E. Huang, A. Pesah, C. T. Chubb, M. Vasmer, and A. Dua, "Tailoring three-dimensional topological codes for biased noise", PRX Quantum **4**, 030338 (2022), arXiv:2211.02116 [quant-ph] (pages 2, 3, 16).

[22] C. Gidney, "Stim: a fast stabilizer circuit simulator", Quantum **5**, 497 (2021), arXiv:2103.02202 [quant-ph] (pages 2, 5, 16).

[23] O. Higgott, "PyMatching: A Python Package for Decoding Quantum Codes with Minimum-Weight Perfect Matching", ACM Transactions on Quantum Computing **3**, 16:1–16:16 (2022), arXiv:2105.13082 [quant-ph] (pages 2, 17).

[24] O. Higgott and C. Gidney, "Sparse Blossom: correcting a million errors per core second with minimum-weight matching", Mar. 28, 2023, arXiv:2303.15933 [quant-ph] (pages 2, 15, 17).

[25] C. Chubb, *Bootstrap threshold analysis script*, `https://gist.github.com/chubbc/59f7dcb8d5d6f7a1a8f5ccc237c8d61d`, 2023 (page 2).

[26] T. Ohno, G. Arakawa, I. Ichinose, and T. Matsui, "Phase Structure of the Random-Plaquette Z_2 Gauge Model: Accuracy Threshold for a Toric Quantum Memory", Nuclear Physics B **697**, 462–480 (2004), arXiv:quant-ph/0401101 (pages 3, 15).

[27] G. Duclos-Cianci and D. Poulin, "Fault-tolerant renormalization group decoder for abelian topological codes", Quantum Info. Comput. **14**, 721–740 (2014) (page 3).

[28] A. B. Aloshious and P. K. Sarvepalli, "Decoding toric codes on three dimensional simplical complexes", IEEE Transactions on Information Theory **67**, 931–945 (2021) (page 3).

[29] Y. Ozeki and N. Ito, "Multicritical dynamics for the ± J Ising Model", Journal of Physics A: Mathematical and General **31**, 5451 (1998) (pages 3, 15).

[30] M. Hasenbusch, F. P. Toldin, A. Pelissetto, and E. Vicari, "Magnetic-glassy multicritical behavior of the three-dimensional ± J Ising model", Physical Review B **76**, 184202 (2007), arXiv:0707.2866 [cond-mat.dis-nn] (pages 3, 15).

[31] A. M. Kubica, "The abcs of the color code: a study of topological quantum codes as toy models for fault-tolerant quantum computation and quantum phases of matter", PhD thesis (California Institute of Technology, 2017) (page 3).

[32] M. Vasmer, D. E. Browne, and A. Kubica, "Cellular automaton decoders for topological quantum codes with noisy measurements and beyond", Scientific Reports **11**, `10.1038/s41598-021-81138-2` (2021) `10.1038/s41598-021-81138-2` (page 3).

[33] K. Duivenvoorden, N. P. Breuckmann, and B. M. Terhal, "Renormalization group decoder for a four-dimensional toric code", IEEE Transactions on Information Theory **65**, 2545–2562 (2019) (page 3).

[34] N. P. Breuckmann and X. Ni, "Scalable Neural Network Decoders for Higher Dimensional Quantum Codes", Quantum **2**, 68 (2018) (page 3).

[35] D. Gottesman, "An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation", in *Quantum Information Science and Its Contributions to Mathematics*, Proceedings of Symposia in Applied Mathematics, 68 (American Mathematical Society, Washington, D.C., Oct. 2010), arXiv:0904.2557 [quant-ph] (page 4).

[36] S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits", Physical Review A **70**, 052328 (2004), arXiv:quant-ph/0406196 (page 5).

[37] M. M. Wilde, "Logical operators of quantum codes", Physical Review A **79**, 062322 (2009), arXiv:0903.5256 [quant-ph] (page 5).

[38] J. C. Bridgeman and C. T. Chubb, "Hand-waving and interpretive dance: an introductory course on tensor networks", Journal of Physics A: Mathematical and Theoretical **50**, 223001 (2017), arXiv:1603.03039 [quant-ph] (page 6).

[39] V. Murg, F. Verstraete, and J. I. Cirac, "Variational study of hard-core bosons in a 2-D optical lattice using Projected Entangled Pair States (PEPS)", Physical Review A **75**, 033605 (2007), arXiv:cond-mat/0611522 (page 13).

[40] M. Lubasch, J. I. Cirac, and M.-C. Bañuls, "Algorithms for finite Projected Entangled Pair States", Physical Review B **90**, 064425 (2014), arXiv:1405.3259 [quant-ph] (page 13).

[41] H. N. Phien, J. A. Bengua, H. D. Tuan, P. Corboz, and R. Orús, "Infinite projected entangled pair states algorithm improved: Fast full update and gauge fixing", Physical Review B **92**, 035142 (2015), arXiv:1503.05345 [cond-mat.str-el] (page 13).

[42] J. Jordan, R. Orus, G. Vidal, F. Verstraete, and J. I. Cirac, "Classical simulation of infinite-size quantum lattice systems in two spatial dimensions", Physical Review Letters **101**, 250602 (2008), arXiv:cond-mat/0703788 (page 13).

[43] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape", Physical Review Research **2**, 043423 (2020), arXiv:2005.07016 [quant-ph] (pages 15, 16).

[44] J. Roffe, *LDPC: Python tools for low density parity check codes*, 2022 (pages 15, 16).

[45] E. Huang and A. Pesah, *Panqec*, `https://github.com/panqec/panqec`, 2023 (page 18).

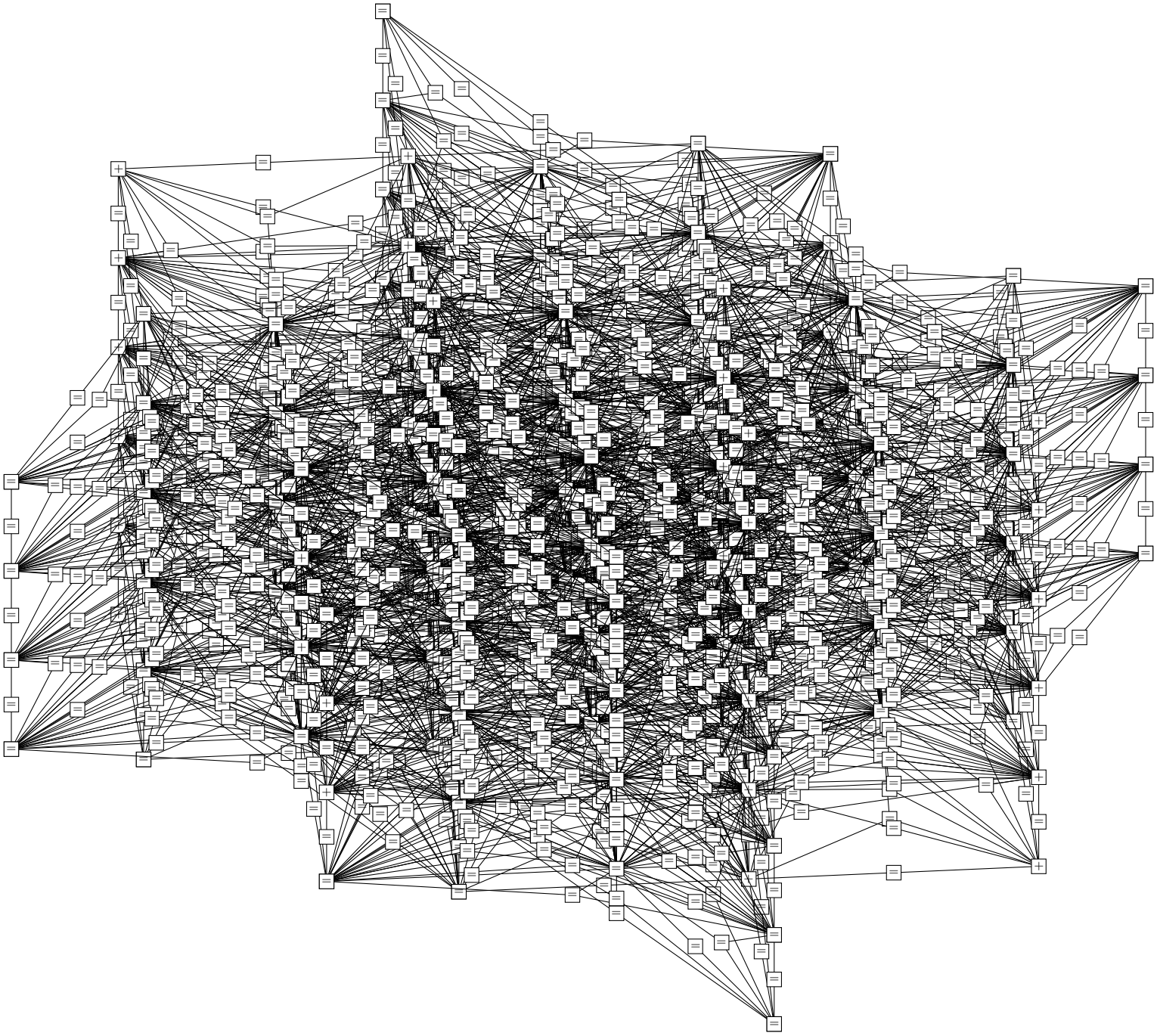


Figure 10: Circuit-level noise tensor network derived from detector error model for the $d = 5$ rotated surface code with 5 rounds of measurements.