PHYSICAL REVIEW A **90**, 032326 (2014)

Efficient algorithms for maximum likelihood decoding in the surface code

Sergey Bravyi, Martin Suchara, and Alexander Vargo
IBM Watson Research Center, Yorktown Heights, New York 10598, USA
(Received 23 June 2014; published 25 September 2014)

We describe two implementations of the optimal error correction algorithm known as the maximum likelihood decoder (MLD) for the two-dimensional surface code with a noiseless syndrome extraction. First, we show how to implement MLD exactly in time $O(n^2)$, where n is the number of code qubits. Our implementation uses a reduction from MLD to simulation of matchgate quantum circuits. This reduction however requires a special noise model with independent bit-flip and phase-flip errors. Secondly, we show how to implement MLD approximately for more general noise models using matrix product states (MPS). Our implementation has running time $O(n\chi^3)$, where χ is a parameter that controls the approximation precision. The key step of our algorithm, borrowed from the density matrix renormalization-group method, is a subroutine for contracting a tensor network on the two-dimensional grid. The subroutine uses MPS with a bond dimension χ to approximate the sequence of tensors arising in the course of contraction. We benchmark the MPS-based decoder against the standard minimum weight matching decoder observing a significant reduction of the logical error probability for $\chi \geqslant 4$.

DOI: 10.1103/PhysRevA.90.032326 PACS number(s): 03.67.Lx, 03.67.Pp

I. INTRODUCTION

The surface code [1,2] is one of the simplest and most studied quantum error correcting codes. It can be realized on a two-dimensional grid of qubits such that the codespace is defined by simple four-qubit parity check operators acting on nearest-neighbor qubits. Recent years have witnessed a surge of interest in the surface code as a promising architecture for a scalable quantum computing [3,4]. Experimental advances in manufacturing of multiqubit devices [5,6] give us hope that a small-scale quantum memory based on the surface code may become a reality soon. Given high operational costs of a quantum hardware compared with the classical one, it is crucial to put enough effort in optimizing algorithmic or software aspects of error correction. In the present paper we focus on optimizing the decoding algorithm that takes as input measured syndromes of the parity checks and computes a recovery operation returning a corrupted state of the memory back to the codespace.

As the name suggests, the maximum likelihood decoder (MLD) is an algorithm that finds a recovery operation maximizing the probability of a successful error correction conditioned on the observed error syndrome. By definition, MLD is the optimal error correction algorithm for a fixed quantum code and a fixed noise model. The first rigorous definition of MLD for the surface codes was proposed by Dennis *et al.* [2]. An important observation made in [2] was that the computational problem associated with MLD can be reduced to computing the partition function of a classical Ising-like Hamiltonian on the two-dimensional lattice. This observation has generated a vast body of work exploring connections between MLD and the statistical physics of disordered Ising-like Hamiltonians; see for instance [7–10]. The insights made in [2] have also guided the search for efficient implementations of MLD. Although an exact and efficient algorithm for MLD remains an elusive goal, several approximate polynomial-time algorithms have been discovered, most notably the renormalization-group decoder due to Duclos-Cianci and Poulin [11], and the Markov chain Monte Carlo method due to Hutter, Wootton, and Loss [12]. In the case of concatenated codes an efficient exact algorithm for MLD based on the message passing algorithm was proposed by Poulin [13]. By comparing MLD with the level-by-level decoder commonly used for concatenated codes, Ref. [13] found that MLD offers a significant advantage with almost twofold increase of the error threshold for the depolarizing noise and a significant reduction of the logical error probability.

Here we propose an alternative method of implementing MLD in the case of the surface code for two simple noise models known as the bit-flip noise and the depolarizing noise. Our method combines the ideas of Dennis *et al.* [2] and the standard classical-to-quantum mapping from classical two-dimensional (2D) spin systems in the thermal equilibrium to quantum 1D spin chains. It enables us to reduce the computational problem associated with MLD to simulating a particular type of quantum dynamics for a chain of qubits.

In the case of the bit-flip noise, MLD can be reduced to simulating a quantum circuit with a special type of two-qubit nearest-neighbor gates known as matchgates. It was shown by Valiant [14] that quantum circuits composed of matchgates can be efficiently simulated by classical means. Matchgate circuits and their generalizations give rise to efficient holographic algorithms for certain combinatorial problems [15] and efficient tensor network contraction methods [16,17]. Matchgate-based algorithms have been used to simulate quantum dynamics in systems of fermionic modes with quadratic interactions [18,19] and study statistics of dimer coverings in classical lattice models [20–22]. Here we demonstrate that matchgates also have applications for quantum error correction. Our simulation algorithm based on fermionic Gaussian states [23] provides an exact implementation of MLD with the running time $O(n^2)$, where n is the number of code qubits. The same algorithm can also be applied to a noise model with independent bit-flip and phase-flip errors. We note that a similar but technically different algorithm has been used by Merz and Chalker in the numerical study of the random-bond 2D Ising model [24].

In the case of the depolarizing noise, MLD can be reduced to simulating the dynamics generated by matrix product operators with a small bond dimension. To perform the simulation efficiently we conjecture that all intermediate states generated by this dynamics are weakly entangled. This enables us to employ a vast body of efficient classical algorithms for simulating weakly entangled quantum spin chains based on matrix product states (MPS); see [25–29]. Our approximate implementation of MLD for the depolarizing noise has running time $O(n\chi^3)$ where χ is a parameter that controls the approximation precision (the bond dimension of the MPS). Although we do not have any rigorous arguments in support of the weak entanglement conjecture, it reflects the physical intuition that the classical 2D spin system associated with MLD has a finite correlation length for error rates below the threshold [2]. Accordingly, one should expect that the classical-to-quantum mapping cannot generate highly entangled states since the latter require long-range correlations. Furthermore, we have justified the conjecture numerically by applying the MPS-based decoder to the bit-flip noise [30]. We observed that the logical error probabilities of the exact MLD and the MPS-based decoder with a relatively small bond dimension $\chi = 6$ are virtually indistinguishable. Likewise, in the case of the depolarizing noise we observed that the logical error probability exhibits a fast convergence as a function of χ suggesting that the MPS-based decoder with $\chi = 6$ implements nearly exact MLD.

Finally, we benchmark the exact and the approximate implementations of MLD against the commonly studied minimum weight matching (MWM) decoder [2,31]. The benchmarking was performed for a fixed code distance $d = 25$ and a wide range of error rates. In the case of the bit-flip noise we observed that the MWM decoder approximates the logical error probability of MLD within a factor of 2. The observed difference between MLD and the MWM decoder can be attributed to the fact that the latter ignores the error degeneracy [32]. Since the observed difference is relatively small, we conclude that ignoring the error degeneracy does not have a significant impact on the decoder's performance for the studied noise model. In the case of the depolarizing noise we observed that the MPS-based decoder is far superior than the MWM decoder offering more than two orders of magnitude reduction of the logical error probability even for small values of χ. This can be attributed to the fact that the MWM decoder often fails to find the minimum weight error consistent with the syndrome since it ignores correlations between X and Z errors [33].

We emphasize that the studied noise models assume perfect syndrome measurements. Extending our decoding algorithms to the more realistic case of noisy syndrome extraction is an interesting open problem that we leave for future work.

The rest of the paper is organized as follows. We formally define the maximum likelihood decoder, the studied noise models, and the surface code in Secs. II, III, and IV, respectively. Our exact implementation of MLD for the bit-flip noise is described in Sec. V. The approximate implementation of MLD based on matrix product states is presented in Sec. VI. A comparison between the exact MLD, the approximate MLD with various bond dimensions χ, and the minimum weight matching decoder is presented in Sec. VII that describes our numerical results.

II. MAXIMUM LIKELIHOOD DECODER

In this section we formally define MLD. We consider a quantum memory composed of n physical qubits. Let $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$ be the full n-qubit Hilbert space and $\mathcal{P}$ be the group of n-qubit Pauli operators. By definition, any element of $\mathcal{P}$ has a form $f = cf_1 \otimes f_2 \otimes \cdots \otimes f_n$, where $f_j \in \{I,X,Y,Z\}$ are single-qubit Pauli operators and $c \in \{\pm 1, \pm i\}$ is an overall phase factor. A quantum code of stabilizer type is defined by an Abelian *stabilizer group* $\mathcal{G} \subset \mathcal{P}$ such that $-I \notin \mathcal{G}$. Quantum codewords are n-qubit states invariant under the action of any element of $\mathcal{G}$. Such states define a *codespace*

$$\mathcal{H}_0 = \{\psi \in \mathcal{H} : g\psi = \psi \text{ for all } g \in \mathcal{G}\}.$$

The encoding step amounts to initializing the memory in some (unknown) state ρ supported on the codespace $\mathcal{H}_0$.

We shall consider a stochastic Pauli noise described by a linear map $\rho \to \mathcal{N}(\rho)$, where

$$\mathcal{N}(\rho) = \sum_{f\in\mathcal{P}} \pi(f) f\rho f^\dagger \tag{1}$$

and π is some normalized probability distribution on the Pauli group. Since the initial state ρ is supported on the codespace $\mathcal{H}_0$, one has $f\rho f^\dagger = \rho$ for any $f \in \mathcal{G}$. By the same token, $f\rho f^\dagger = h\rho h^\dagger$ whenever $f\mathcal{G} = h\mathcal{G}$. Given a Pauli operator $f \in \mathcal{P}$, a subset $f\mathcal{G} \equiv \{fg : g \in \mathcal{G}\}$ is called a *coset* of $\mathcal{G}$. Clearly, $\mathcal{P}$ is a disjoint union of cosets $\mathcal{C}^\alpha = f^\alpha\mathcal{G}$, where f^α is some fixed representative of $\mathcal{C}^\alpha$. The above shows that errors in the same coset have the same action on the codespace. Thus

$$\mathcal{N}(\rho) = \sum_{\alpha} \pi(f^\alpha\mathcal{G}) f^\alpha \rho f^\alpha, \tag{2}$$

where the sum ranges over all cosets of $\mathcal{G}$ and

$$\pi(f\mathcal{G}) \equiv \sum_{g\in\mathcal{G}} \pi(fg).$$

For simplicity, here we assumed that all coset representatives f^α are Hermitian operators. We shall refer to the quantity $\pi(f\mathcal{G})$ as a *coset probability*.

At the decoding step one attempts to guess the coset of the stabilizer group that contains the actual error based on partial information about the error known as a syndrome. More precisely, let $g^1,\ldots,g^m \in \mathcal{G}$ be some fixed set of generators of $\mathcal{G}$. Since the generators g^i pairwise commute, they can be diagonalized simultaneously. A configuration of eigenvalues $g^i = \pm 1$ can be described by a *syndromes* $\in \{0,1\}^m$ such that $g^i = (-1)^{s_i}$ for all $i = 1,\ldots,m$, where $s_i \in \{0,1\}$. Assuming that the generators g^i are independent, there are 2^m possible syndromes. The full Hilbert space can be decomposed into a direct sum of syndrome subspaces

$$\mathcal{H} = \bigoplus_{s\in\{0,1\}^m} \mathcal{H}_s,$$

where $\mathcal{H}_s = \{\psi \in \mathcal{H} : g^i\psi = (-1)^{s_i}\psi \quad \text{for all i}\}$. Note that the codespace $\mathcal{H}_0$ corresponds to the zero syndrome. A Pauli

operator $f \in \mathcal{P}$ is said to have a syndrome s iff $fg^i = (-1)^{s_i} g^i f$ for all $i = 1, \ldots, m$. Equivalently, f has a syndrome s iff $f\mathcal{H}_0 = \mathcal{H}_s$. For each syndrome s let us choose some fixed Pauli operator $f(s)$ with the syndrome s. One can easily check that the set of all Pauli operators with a syndrome s coincides with the coset $f(s)\mathcal{C}(\mathcal{G})$, where

$$\mathcal{C}(\mathcal{G}) = \{f \in \mathcal{P} : fg = gf \text{ for all } g \in \mathcal{G}\}$$

is a group known as the centralizer of $\mathcal{G}$. Note that $\mathcal{G} \subseteq \mathcal{C}(\mathcal{G})$. Thus each coset of $\mathcal{C}(\mathcal{G})$ can be partitioned into a disjoint union of several cosets of $\mathcal{G}$. In the present paper we only consider stabilizer codes with a single logical qubit. Let $\overline{X}, \overline{Y}, \overline{Z} \in \mathcal{C}(\mathcal{G}) \backslash \mathcal{G}$ be the logical Pauli operators on the encoded qubit. Then each coset of $\mathcal{C}(\mathcal{G})$ consists of four disjoint cosets of $\mathcal{G}$, namely,

$$f(s)\mathcal{C}(\mathcal{G}) = \mathcal{C}_I^s \cup \mathcal{C}_X^s \cup \mathcal{C}_Y^s \cup \mathcal{C}_Z^s, \tag{3}$$

where

$$\mathcal{C}_I^s = f(s)\mathcal{G}, \quad \mathcal{C}_X^s = f(s)\overline{X}\mathcal{G}, \tag{4}$$

$$\mathcal{C}_Y^s = f(s)\overline{Y}\mathcal{G}, \quad \mathcal{C}_Z^s = f(s)\overline{Z}\mathcal{G}. \tag{5}$$

The decoding step starts by a *syndrome measurement* that projects the corrupted state $\mathcal{N}(\rho)$ onto one of the syndrome subspaces $\mathcal{H}_s$. The above arguments show that $f\rho f^\dagger$ has support on $\mathcal{H}_s$ iff $f \in f(s)\mathcal{C}(\mathcal{G})$. Thus the syndrome measurement reveals the coset of $\mathcal{C}(\mathcal{G})$ that contains the error f, whereas our goal is to determine which coset of $\mathcal{G}$ contains f. Using Eqs. (2), and (3)–(5), one can write the postmeasurement (unnormalized) state as

$$\begin{aligned}\rho(s) = {} & \pi\left(\mathcal{C}_I^s\right) f(s)\rho f(s) + \pi\left(\mathcal{C}_X^s\right) f(s)\overline{X}\rho\overline{X} f(s) \\ & + \pi\left(\mathcal{C}_Y^s\right) f(s)\overline{Y}\rho\overline{Y} f(s) + \pi\left(\mathcal{C}_Z^s\right) f(s)\overline{Z}\rho\overline{Z} f(s),\end{aligned} \tag{6}$$

where s is the observed syndrome. Here we assumed for simplicity that $f(s)$ and the logical operators $\bar{X}, \bar{Y}, \bar{Z}$ are Hermitian. This shows that the effective noise model conditioned on the syndrome can be described by applying one of the four Pauli errors $f(s), f(s)\bar{X}, f(s)\bar{Y}$, and $f(s)\bar{Z}$ with probabilities $\pi(\mathcal{C}_I^s), \pi(\mathcal{C}_X^s), \pi(\mathcal{C}_Y^s)$, and $\pi(\mathcal{C}_Z^s)$, respectively. Clearly, the best possible error correction algorithm for this effective noise model is to choose a recovery operator as the most likely of the four errors. Equivalently, we should choose a recovery operator as any Pauli operator that belongs to the most likely of the four cosets $\mathcal{C}_I^s, \mathcal{C}_X^s, \mathcal{C}_Y^s, \mathcal{C}_Z^s$, which we denote $\mathcal{C}_{\mathrm{ML}}^s$. These steps can be summarized as follows:

ML Decoder
Input: syndrome $s \in \{0,1\}^m$
Output: recovery operator $g \in \mathcal{P}$

$f(s) \leftarrow$ any Pauli operator with a syndrome s
$\mathcal{C}_{\mathrm{ML}}^s \leftarrow \arg\max_{\mathcal{C}} \pi(\mathcal{C})$, where $\mathcal{C} \in \{\mathcal{C}_I^s, \mathcal{C}_X^s, \mathcal{C}_Y^s, \mathcal{C}_Z^s\}$
return any $g \in \mathcal{C}_{\mathrm{ML}}^s$

The final step of the decoding is to apply the optimal recovery operator g. It results in a state $g\rho(s)g^\dagger$. We conclude that MLD correctly identifies the coset of $\mathcal{G}$ that contains the actual error and maps the corrupted state $\mathcal{N}(\rho)$ back to the encoded state ρ with a probability

$$P_{\mathrm{success}} = \sum_{s \in \{0,1\}^m} \pi\left(\mathcal{C}_{\mathrm{ML}}^s\right).$$

The logical error probability $1 - P_{\mathrm{success}}$ for a particular noise model can be computed numerically using the Monte Carlo method; see Sec. VII for details.

In what follows we shall always ignore overall phase factors of Pauli operators. Such phase factors are irrelevant for our purposes since they do not change the outcome of error correction.

III. NOISE MODELS

We shall consider a stochastic i.i.d. (independent and identically distributed) Pauli noise

$$\mathcal{N} = \bigotimes_{j=1}^{n} \mathcal{N}_j,$$

where

$$\mathcal{N}_j(\rho) = (1-\epsilon)\rho + \epsilon_X X\rho X + \epsilon_Y Y\rho Y + \epsilon_Z Z\rho Z$$

and $\epsilon \equiv \epsilon_X + \epsilon_Y + \epsilon_Z$ is called an *error rate*. Two commonly studied noise models are the classical bit-flip noise where only X-type errors are allowed (the X noise) and the depolarizing noise where all types of errors are equally likely. The formal definitions are given below:

$$X \text{ noise:} \quad \epsilon_X = \epsilon, \quad \epsilon_Y = \epsilon_Z = 0,$$
$$\text{Depolarizing noise:} \quad \epsilon_X = \epsilon_Y = \epsilon_Z = \epsilon/3.$$

The corresponding probability distributions on the Pauli group are

$$\pi(f) = (1-\epsilon)^{n-|f|}(\epsilon/3)^{|f|}$$

for the depolarizing noise and

$$\pi(f) = \begin{cases} (1-\epsilon)^{n-|f|}\epsilon^{|f|} & \text{if} \quad f \in \mathcal{P}^X, \\ 0 & \text{otherwise} \end{cases}$$

for the X noise. Here $|f|$ denotes the Hamming weight of f, that is, the number of qubits on which f acts nontrivially, while $\mathcal{P}^X \subset \mathcal{P}$ denotes the subgroup generated by single-qubit Pauli X operators.

One may also consider a noise model with independent bit-flip and phase-flip errors, that is, $\epsilon_X = \epsilon_Z$ and $\epsilon_Y = (\epsilon_X)^2$. Since there are no correlations between the two types of errors, one can perform error correction independently for bit-flip and phase-flip errors. Furthermore, since correcting phase-flip errors is equivalent to correcting bit-flip errors on the surface code lattice rotated by 90°, it suffices to consider the X-noise model only.

IV. SURFACE CODES

We consider the surface code on a square lattice of size $d \times d$ with open boundary conditions. The boundaries parallel to the horizontal (vertical) axis are smooth (rough). The surface code lattice with $d = 3$ is shown on Fig. 1. For the chosen geometry the surface code encodes one logical qubit into $n = d^2 + (d-1)^2$ physical qubits with the minimum distance

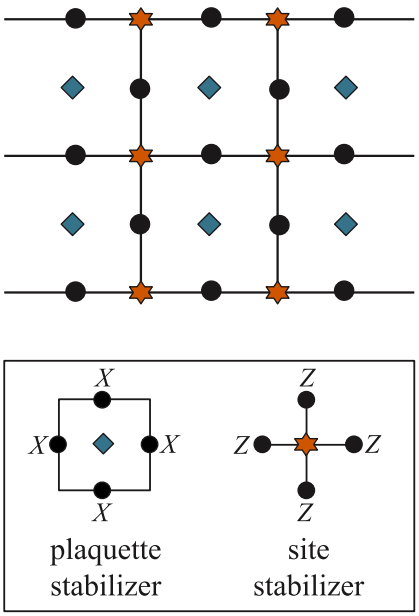


FIG. 1. (Color online) Distance-3 surface code. Solid dots, stars, and diamonds indicate locations of qubits, site stabilizers, and plaquette stabilizers, respectively. Stabilizers located near the boundary act only on three qubits. The distance-d surface code has d^2 qubits on horizontal edges, $(d-1)^2$ qubits on vertical edges, and $d(d-1)$ stabilizers of each type.

d. We shall always consider odd values of d such that the code corrects any combination of $(d-1)/2$ single-qubit errors. Let A_u and B_p be the stabilizers of the surface code associated with a site u and a plaquette p, respectively. We have $B_p = \prod_{e\in p} X_e$, where the product runs over all edges e making up the boundary of p. Likewise, $A_u = \prod_{e\ni u} Z_e$, where the product runs over all edges e incident to u. Let $\mathcal{G}^Z = \langle A_u\rangle$ and $\mathcal{G}^X = \langle B_p\rangle$ be the subgroups of the Pauli group $\mathcal{P}$ generated by all site stabilizers and all plaquette stabilizers, respectively. Finally, let $\mathcal{G} = \langle A_u, B_p\rangle$ be the full stabilizer group. Logical Pauli operators $\overline{X}, \overline{Z}$ are shown on Fig. 2, while $\overline{Y} = i\overline{X}\overline{Z}$.

By a slight abuse of notations, below we shall often identify a Pauli operator f with the subset of edges in the lattice on which f acts nontrivially.

V. EXACT ALGORITHM

In this section we consider the X noise and describe an exact implementation of MLD. We begin by specializing MLD to the X noise (Sec. V A) and describing our algorithm (Sec. V B). A reader interested only in the question of *how* the algorithm works can skip the remaining sections explaining *why* it works and proving its correctness. Specifically, Sec. V C shows how to express the coset probability as a matrix element of a matchgate quantum circuit. Our derivation partially follows the one of Refs. [2,24]. An efficient method of simulating matchgate circuits based on fermionic Gaussian states is described in Sec. V D. The material of this section mostly follows Ref. [23].

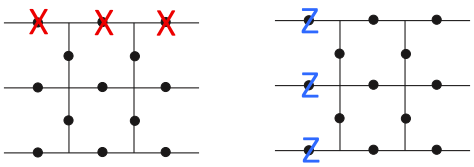


FIG. 2. (Color online) Logical Pauli operators $\overline{X}$ (left) and $\overline{Z}$ (right).

A. Specializing the ML decoder to X noise

Let s be the input syndrome and $f(s) \in \mathcal{P}$ be some fixed Pauli error consistent with s. We can always choose $f(s) \in \mathcal{P}^X$, that is, such that $f(s)$ acts on any qubit by I or X. Indeed, since only X-type errors can appear with a nonzero probability, the syndromes of all plaquette stabilizers must be zero. Let s_u be the syndrome of a site stabilizer A_u. We choose the desired error $f(s)$ by connecting each site u with a nonzero syndrome s_u to the left boundary by a horizontal string of X errors and adding all such strings modulo 2. Note that $f(s)$ can be constructed in time $O(n)$.

Let π be the probability distribution on the Pauli group describing the X noise; see Sec. III. To implement the ML decoder it suffices to compute the four coset probabilities $\pi(\mathcal{C}_I^s)$, $\pi(\mathcal{C}_X^s)$, $\pi(\mathcal{C}_Y^s)$, and $\pi(\mathcal{C}_Z^s)$ as defined in Sec. II. Note that $\pi(\mathcal{C}_Y^s) = \pi(\mathcal{C}_Z^s) = 0$ since any element of these two cosets acts by Pauli Z on at least d qubits. Choose any logical operator $\overline{L} \in \{\overline{I}, \overline{X}\}$ and let $f \equiv f(s)\overline{L}$. From now on we shall assume that f is fixed. Since Z-type errors are not allowed, one has $\pi(f\mathcal{G}) = \pi(f\mathcal{G}^X)$. Thus it suffices to compute the coset probability $\pi(f\mathcal{G}^X)$.

B. Algorithm for computing the coset probability

In this section we describe an algorithm that takes as input an X-type Pauli operator f and outputs the coset probability $\pi(f\mathcal{G}^X)$. The algorithm has running time $O(n^2)$.

Let us begin by introducing some notations. The sets of all horizontal and vertical edges of the surface code lattice will be denoted H and V, respectively. For the code of distance d one has $|H| = d^2$ and $|V| = (d-1)^2$. We partition the set H into columns of edges such that

$$H = H^1 \cup H^2 \cup \ldots \cup H^d,$$

where H^j denotes the jth leftmost column of horizontal edges; see Fig. 3. Edges of every column H^j will be labeled by integers $1, \ldots, d$ starting from the top edge. Likewise,

$$V = V^1 \cup V^2 \cup \ldots \cup V^{d-1},$$

where V^j denotes the jth leftmost column of vertical edges; see Fig. 3. Edges of every column V^j will be labeled by integers $1, \ldots, d-1$ starting from the top edge. We shall refer to H^j and V^j as horizontal and vertical columns, respectively.

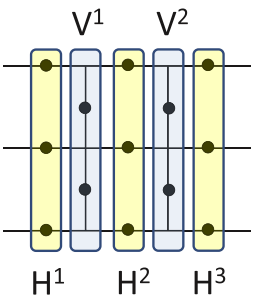


FIG. 3. (Color online) Partition of edges into "horizontal" columns $H^1,\ldots,H^d$ and vertical columns $V^1,\ldots,V^{d-1}$. Every edge is identified with the respective code qubit (solid dot).

For each edge e of the surface code lattice define a weight

$$w_e = \begin{cases} \epsilon(1-\epsilon)^{-1} & \text{if} \quad e \notin f, \\ \epsilon^{-1}(1-\epsilon) & \text{if} \quad e \in f. \end{cases} \tag{7}$$

Recall that ϵ is the error rate.

For any integer $m \geqslant 1$ and a vector $\lambda \in \mathbb{R}^m$ let $\mathcal{A}(\lambda)$ be the antisymmetric matrix of size $(m+1)\times(m+1)$ that contains λ above the main diagonal and $-\lambda$ below the main diagonal. For example, if $\lambda = (\lambda_1,\lambda_2,\lambda_3)$, then

$$\mathcal{A}(\lambda) = \begin{bmatrix} 0 & \lambda_1 & 0 & 0 \\ -\lambda_1 & 0 & \lambda_2 & 0 \\ 0 & -\lambda_2 & 0 & \lambda_3 \\ 0 & 0 & -\lambda_3 & 0 \end{bmatrix}.$$

Let $\mathcal{D}(\lambda)$ be the diagonal matrix of size $m\times m$ that contains λ on the main diagonal. Define also a standard antisymmetric matrix

$$M_0 = \begin{bmatrix} 0 & & & & & 1 \\ & \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} & & & & \\ & & \ddots & & & \\ & & & \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} & & \\ -1 & & & & & 0 \end{bmatrix}, \tag{8}$$

such that M_0 has size $2d\times 2d$. The matrix M_0 contains $d-1$ blocks of size 2×2 on the main diagonal and two nonzero elements $M_{1,2d} = 1 = -M_{2d,1}$. All remaining elements of M_0 are zero. Let I be the identity matrix of size $2d\times 2d$.

The first step of our algorithm is to compute the probability of the input error $\pi(f) = (1-\epsilon)^{n-|f|}\epsilon^{|f|}$ and the coefficients w_e defined in Eq. (7). This step takes time $O(n)$. At each subsequent step of the algorithm we maintain a pair (M,Γ), where M is an antisymmetric real matrix of size $2d\times 2d$ and $\Gamma \geqslant 0$ is a real number. The algorithm calls two functions SimulateHorizontal(j,M,Γ) and SimulateVertical(j,M,Γ) that update the pair (M,Γ) by applying a simple combination of matrix inversions and matrix multiplications:

Algorithm 1

Input: X-type Pauli operator f
Output: Coset probability $\pi(f\mathcal{G}^X)$
Compute the coefficients w_e defined in Eq. (7)
$\pi(f) \leftarrow (1-\epsilon)^{n-|f|}\epsilon^{|f|}$
$M \leftarrow M_0$
$\Gamma \leftarrow 2^{d-1}$
for $j = 1$ to $d-1$ do
  SimulateHorizontal(j,M,Γ)
  SimulateVertical(j,M,Γ)
end for
SimulateHorizontal(d,M,Γ)
return $\pi(f)\sqrt{\Gamma/2}\,\det(M+M_0)^{1/4}$

function SimulateHorizontal(j,M,Γ)
  for $i = 1$ to d do
    $e \leftarrow i$-th edge of the column H^j
    $\Gamma \leftarrow \Gamma\,(1+w_e^2)/2$
    $t_i \leftarrow (1-w_e^2)/(1+w_e^2)$
    $s_i \leftarrow 2w_e/(1+w_e^2)$
  end for
  $A \leftarrow \mathcal{A}(t_1 0 t_2 0 \ldots t_{d-1} 0 t_d)$
  $B \leftarrow \mathcal{D}(s_1 s_1 s_2 s_2 \ldots s_d s_d)$
  $\Gamma \leftarrow \Gamma\sqrt{\det(M+A)}$
  $M \leftarrow A - B(M+A)^{-1}B$
end function

function SimulateVertical(j,M,Γ)
  for $i = 1$ to $d-1$ do
    $e \leftarrow i$-th edge of the column V^j
    $\Gamma \leftarrow \Gamma\,(1+w_e^2)$
    $t_i \leftarrow 2w_e/(w_e^2+1)$
    $s_i \leftarrow (1-w_e^2)/(1+w_e^2)$
  end for
  $A \leftarrow \mathcal{A}(0 t_1 0 t_2 \ldots 0 t_{d-1} 0)$
  $B \leftarrow \mathcal{D}(1 s_1 s_1 s_2 s_2 \ldots s_{d-1} s_{d-1} 1)$
  $\Gamma \leftarrow \Gamma\sqrt{\det(M+A)}$
  $M \leftarrow A - B(M+A)^{-1}B$
end function

If implemented naively, each matrix inversion and each matrix multiplication takes time $O(d^3)$. Likewise, computing each determinant takes time $O(d^3)$. Simple counting then shows that the overall running time of the algorithm is $O(d^4) = O(n^2)$. Suggestions on improving stability of the algorithm against rounding errors can be found in Sec. VII.

C. Reduction to a matchgate quantum circuit

Consider any stabilizer $g \in \mathcal{G}^X$. A simple algebra shows that

$$\pi(fg) = \pi(f)\prod_{e\in g} w_e,$$

where w_e are the weights defined in Eq. (7). Thus

$$\pi(f\mathcal{G}^X) = \pi(f)\mathcal{Z}(w),$$

where $w = \{w_e\}$ is the list of coefficients w_e and

$$\mathcal{Z}(w) = \sum_{g\in\mathcal{G}^X}\prod_{e\in g} w_e. \tag{9}$$

Since the factor $\pi(f)$ is easy to compute, below we concentrate on computing $\mathcal{Z}(w)$. We shall express $\mathcal{Z}(w)$ as a matrix element of a certain quantum circuit acting on d qubits. The circuit will be composed of single-qubit and two-qubit gates

$$G(w) \equiv \begin{bmatrix} 1 & 0 \\ 0 & w \end{bmatrix}, \quad G'(w) \equiv \begin{bmatrix} 1 & 0 & 0 & w \\ 0 & 1 & w & 0 \\ 0 & w & 1 & 0 \\ w & 0 & 0 & 1 \end{bmatrix}, \quad (10)$$

where w is a real parameter. We note that $G(w)$ and $G'(w)$ are not unitary gates. Let $\mathcal{H}_d = (\mathbb{C}^2)^{\otimes d}$ be the Hilbert space of d qubits. For each horizontal column H^j and each vertical column V^j defined at Fig. 3 define linear operators $\hat{H}^j, \hat{V}^j$ acting on $\mathcal{H}_d$ such that

$$\hat{H}^j = G(w_{e_1}) \otimes \cdots \otimes G(w_{e_d}) \quad (11)$$

and

$$\hat{V}^j = G'_{12}(w_{e_1}) G'_{23}(w_{e_2}) \cdots G'_{d-1,d}(w_{e_{d-1}}). \quad (12)$$

Here the subscripts indicate the qubits acted upon by each gate and e_i denotes the ith edge of the respective columns H^j and V^j counting from the top to the bottom. Finally, define a state

$$|\psi_e\rangle = \sum_{x \in \{0,1\}^d_{\text{even}}} |x\rangle, \quad (13)$$

where $\{0,1\}^d_{\text{even}}$ is the set of all d-bit binary strings with the even Hamming weight.

Lemma 1. One has

$$\mathcal{Z}(w) = \langle \psi_e | \hat{U} | \psi_e \rangle, \quad (14)$$

where

$$\hat{U} = \hat{H}^d \hat{V}^{d-1} \cdots \hat{H}^2 \hat{V}^1 \hat{H}^1 \quad (15)$$

is a quantum circuit on d qubits shown at Fig. 4.

The gates $G(w)$ and $G'(w)$ defined in Eq. (10) are examples of the so-called matchgates discovered by Valiant [14]. It was

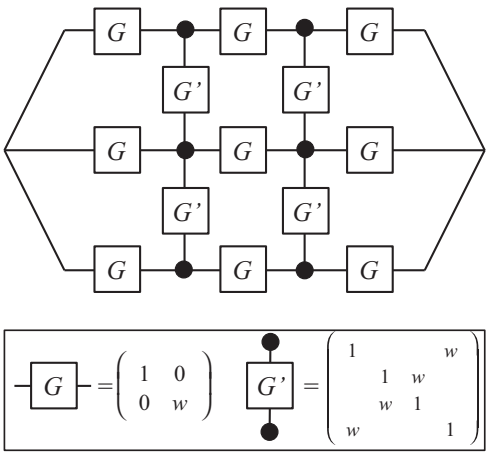


FIG. 4. Computing the coset probabilities for the X-noise model is equivalent to computing the matrix element $\langle \psi_e | \hat{U} | \psi_e \rangle$, where $\hat{U}$ is a quantum circuit on d qubits shown above and ψ_e is the superposition of all even-weight d-bit strings. The above example is for $d = 3$. Each gate depends on a parameter w_e defined in Eq. (7).

shown in [14] that quantum circuits composed of matchgates can be efficiently simulated by classical means. In the next section we describe an alternative algorithm for computing the quantity $\langle \psi_e | \hat{U} | \psi_e \rangle$ based on fermionic Gaussian states with a running time $O(n^2)$. [For comparison, the original algorithm of Ref. [14] would have running time $O(n^3)$ since it requires computing the Pfaffian of a matrix of size $O(n)$.]

In the rest of this section we prove Lemma 1.

Proof.

Proposition 1. For any subset $T \subseteq H$ such that $|T \cap H^j|$ is even for all $j = 1, \ldots, d$, there exists a unique $g \in \mathcal{G}^X$ such that $g \cap H = T$.

Proof. Recall that a subset of edges g is called a cycle iff any site has even number of incident edges from g. Let us first show that for any $T \subseteq H$ there exists exactly one cycle g such that $g \cap H = T$. Indeed, consider any vertical column V^j. It comprises a set of sites $u_1, \ldots, u_d$ and a set of edges $e_1, \ldots, e_{d-1}$ (listed in the order from the top to the bottom). Since $g \cap H^1 = T \cap H^1$ and $g \cap H^2 = T \cap H^2$, the cycle condition at u_1 uniquely determines g_{e_1}. Once g_{e_1} is determined, the cycle condition u_2 uniquely determines g_{e_2}. Continuing in this fashion uniquely determines $g \cap V^j$. Since V^j can be any vertical column, we conclude that g is uniquely determined by T. It remains to note that $\mathcal{G}^X$ coincides with the set of cycles that have even intersection with any column H^j. ■

Let $g(T) \in \mathcal{G}^X$ be the Pauli operator constructed in Proposition 1. Then

$$\mathcal{Z}(w) = \sum_{T \subseteq H} \prod_{e \in g(T)} w_e, \quad (16)$$

where the sum ranges over all subsets T such that $|T \cap H^j|$ is even for all j. Let $T^j \equiv T \cap H^j$. We can regard T^j as a binary d-bit string such that $T^j_i = 1$ iff the ith edge of H^j belongs to T. Let $|T^j\rangle \in \mathcal{H}_d$ be the basis vector corresponding to T^j. Since $g(T) \cap H^j = T^j$, we have

$$\prod_{e \in g(T) \cap H^j} w_e = \langle T^j | G(w_{e_1}) \otimes \cdots \otimes G(w_{e_d}) | T^j \rangle, \quad (17)$$

where $e_1, \ldots, e_d$ are the edges comprising the column H^j listed in the order from the top to the bottom and $G(w)$ is the single-qubit gate defined in Eq. (10).

Consider now some vertical column V^j. Let $e_1, \ldots, e_{d-1}$ be the edges comprising V^j listed in the order from the top to the bottom. We claim that

$$\prod_{e \in g(T) \cap V^j} w_e = \langle T^j | G'_{12}(w_{e_1}) G'_{23}(w_{e_2}) \cdots \cdots G'_{d-1,d}(w_{e_{d-1}}) | T^{j+1} \rangle, \quad (18)$$

where $G'(w)$ is the two-qubit gate defined in Eq. (10) and $G'_{i,i+1}(w) : \mathcal{H}_d \to \mathcal{H}_d$ denotes the gate $G'(w)$ applied to the pair of qubits $i, i+1$. One can easily check Eq. (18) by noting that $G'(w) = I \otimes I + wX \otimes X$ and following the arguments given in proof of Propostion 1 to reconstruct $g(T) \cap V^j$ from T^j and T^{j+1}.

Let $\{0,1\}^d_{\text{even}}$ be the set of all d-bit strings with even Hamming weight. Combining Eqs. (16), (17), and (18), one

arrives at

$$\mathcal{Z}(w) = \sum_{T^1,\ldots,T^d \in \{0,1\}^d_{\text{even}}} \langle T^d|\hat{H}^d|T^d\rangle\langle T^d|\hat{V}^{d-1}|T^{d-1}\rangle \cdots \cdots \langle T^2|\hat{V}^1|T^1\rangle\langle T^1|\hat{H}^1|T^1\rangle, \quad (19)$$

where $\hat{H}^j$ and $\hat{V}^j$ are the linear operators on $\mathcal{H}_d$ defined in Eqs. (11) and (12).

Let $\mathcal{H}_d^{\text{even}} \subseteq \mathcal{H}_d$ be the subspace spanned by vectors $|x\rangle$ with $x \in \{0,1\}^d_{\text{even}}$. Note that the operators $\hat{H}^j$ and $\hat{V}^j$ preserve $\mathcal{H}_d^{\text{even}}$ since the gates $G(w)$ and $G'(w)$ preserve the Hamming weight modulo 2. The above observations imply that $\mathcal{Z}(w) = \langle\psi_e|\hat{U}|\psi_e\rangle$, which completes the proof of Lemma 1. ■

D. Fermionic Gaussian states

Let $\mathcal{H}_d$ be the Hilbert space of d qubits. For each $p = 1,\ldots,2d$ define a *Majorana operator* $\hat{c}_p$ acting on $\mathcal{H}_d$ such that

$$\hat{c}_{2j-1} = Z_1 \cdots Z_{j-1} X_j, \quad \hat{c}_{2j} = Z_1 \cdots Z_{j-1} Y_j. \quad (20)$$

The Majorana operators obey the well-known commutation rules

$$\hat{c}_p\hat{c}_q + \hat{c}_q\hat{c}_p = 2I\delta_{p,q}, \quad \hat{c}_p^2 = I, \quad \hat{c}_p^\dagger = \hat{c}_p. \quad (21)$$

We shall often use a formula

$$Z_j = (-i)\hat{c}_{2j-1}\hat{c}_{2j}, \quad X_jX_{j+1} = (-i)\hat{c}_{2j}\hat{c}_{2j+1}. \quad (22)$$

A *covariance matrix* of a pure (unnormalized) state $\psi \in \mathcal{H}_d$ is a $2d\times 2d$ matrix M with matrix elements

$$M_{p,q} = \frac{(-i)}{2\langle\psi|\psi\rangle}\langle\psi|\hat{c}_p\hat{c}_q - \hat{c}_q\hat{c}_p|\psi\rangle. \quad (23)$$

From Eq. (21) one can easily check that M is a real antisymmetric matrix.

Consider as an example the state ψ_e defined in Eq. (13). Let us compute its covariance matrix M. One can easily check that ψ_e is a stabilizer state with the stabilizer group

$$\mathcal{G}(\psi_e) = \langle X_1X_2, X_2X_3, \ldots, X_{d-1}X_d, Z_1Z_2\cdots Z_d\rangle.$$

Applying Eq. (22) one can get an alternative set of generators that are quadratic in Majorana operators,

$$\mathcal{G}(\psi_e) = \langle(-i)\hat{c}_2\hat{c}_3, \ldots, (-i)\hat{c}_{2d-2}\hat{c}_{2d-1}, (-i)\hat{c}_1\hat{c}_{2d}\rangle. \quad (24)$$

This shows that $M_{2j,2j+1} = 1$ for all $j = 1,\ldots,d-1$ and $M_{1,2d} = 1$. Furthermore, $\langle\psi_e|\hat{c}_p\hat{c}_q|\psi_e\rangle = 0$ whenever $\hat{c}_p\hat{c}_q$ anticommutes with at least one of the generators defined in Eq. (24). Combining the above observations one can easily check that ψ_e has the covariance matrix M_0 defined in Eq. (8).

A state $\psi \in \mathcal{H}_d$ is said to obey the Wick's theorem iff the expectation value of any even tuple of Majorana operators on ψ can be computed from its covariance matrix M using the formula

$$\langle\psi|(-i)^m\hat{c}_{p_1}\hat{c}_{p_2}\cdots\hat{c}_{p_{2m}}|\psi\rangle = \Gamma\cdot\text{Pf}\big(M|_{p_1,p_2,\ldots,p_{2m}}\big), \quad (25)$$

where $\Gamma = \langle\psi|\psi\rangle$ is the norm of ψ, $M|_{p_1,p_2,\ldots,p_{2m}}$ is the $2m\times 2m$ submatrix of M formed by the rows and columns $p_1,p_2,\ldots,p_{2m}$, and Pf is the Pfaffian [14]. Recall that the Pfaffian of an antisymmetric matrix K of size $2m\times 2m$ is defined as

$$\text{Pf}(K) = \frac{1}{2^m m!}\mathcal{A}(K_{1,2}K_{3,4}\cdots K_{2m-1,2m}),$$

where $\mathcal{A}$ stands for the antisymmetrization over all $(2m)!$ permutations of indexes. For example, $\text{Pf}(K) = K_{1,2}$ for $m = 1$ and

$$\text{Pf}(K) = K_{1,2}K_{3,4} - K_{1,3}K_{2,4} + K_{1,2}K_{3,4}$$

for $m = 2$.

A state $\psi \in \mathcal{H}_d$ is called a (fermionic) *Gaussian state* iff it obeys the Wick's theorem and, in addition, all odd tuples of Majorana operators have zero expectation value on ψ. By definition, a Gaussian state ψ is fully specified by the pair (M,Γ), where M is the covariance matrix of ψ and $\Gamma = \langle\psi|\psi\rangle$ is the norm. Below we shall identify a Gaussian state and the corresponding pair (M,Γ).

We shall need the following well-known facts; see, for instance, Ref. [23].

Fact 1. A state ψ is Gaussian iff its covariance matrix obeys $MM^T = I$.

One can easily check that standard antisymmetric matrix M_0 defined in Eq. (8) satisfies $M_0M_0^T = I$. This shows that ψ_e is a Gaussian state with the covariance matrix M_0 and the norm $\Gamma = 2^{d-1}$.

Fact 2. Let $\psi = (M,\Gamma)$ and $\phi = (M',\Gamma')$ be Gaussian states of d qubits. Then

$$|\langle\phi|\psi\rangle| = \frac{\sqrt{\Gamma\Gamma'}}{2^{d/2}}\det(M+M')^{1/4}. \quad (26)$$

Fact 3. Let G be a (complex) antisymmetric matrix of size $2d\times 2d$. Consider an operator

$$W = \exp(\hat{G}), \quad \hat{G} = \sum_{1\leqslant p<q\leqslant 2d} G_{p,q}\hat{c}_p\hat{c}_q. \quad (27)$$

Then W maps Gaussian states to Gaussian states.

The last fact will be very important for us since the operators $\hat{H}^j$ and $\hat{V}^j$ constructed in Sec. V C have the form Eq. (27). Indeed, consider the gates $G(w)_a$ and $G'(w)_{a,a+1}$ where $G(w), G'(w)$ are defined in Eq. (10) and the subscripts indicate which qubits are acted upon by the gate. One can easily check that $G(w)_a = \sqrt{w}e^{\beta Z_a}$, where β is defined through $e^{-2\beta} = w$. Likewise, $G'(w)_{a,a+1} = \sqrt{w}e^{\beta X_aX_{a+1}}$. From Eq. (22) and Eq. (11) one gets

$$\hat{H}^j = \sqrt{w_{e_1}\cdots w_{e_d}}\exp\left(\sum_{a=1}^d \beta_a(-i)\hat{c}_{2a-1}\hat{c}_{2a}\right), \quad (28)$$

where β_a is defined through

$$e^{-2\beta_a} = w_{e_a}, \quad a = 1,\ldots,d.$$

The relation between the parameter β_a and the error rate ϵ defined by the above equation and Eq. (7) is known as the Nishimori line condition [2]. Likewise, Eq. (12) implies

$$\hat{V}^j = \sqrt{w_{e_1}\cdots w_{e_{d-1}}}\exp\left(\sum_{a=1}^{d-1} \beta_a(-i)\hat{c}_{2a}\hat{c}_{2a+1}\right). \quad (29)$$

Since ψ_e is a Gaussian state, Fact 3 implies that all intermediate states obtained from ψ_e by applying the operators $\hat{H}^j$ and $\hat{V}^j$

are Gaussian. Therefore, $\mathcal{Z}(w) = \langle\psi_e|\hat{H}^d\hat{V}^{d-1}\cdots\hat{V}^1\hat{H}^1|\psi_e\rangle$ can be efficiently computed if we have a rule describing how the covariance matrix and the norm of a Gaussian state change upon application of $\hat{H}^j$ and $\hat{V}^j$. The desired rule can be obtained using a fermionic version of the Jamiolkowski duality between states and linear maps introduced in [23]. Let us first define a fermionic version of the maximally entangled state for a bipartite system of $d+d$ qubits. Let $\hat{c}_1,\ldots,\hat{c}_{4d}$ be the Majorana operators defined for a system of $2d$ qubits according to Eq. (20). Define a $2d$-qubit state normalized ψ_I such that ψ_I has a stabilizer group

$$\mathcal{G}(\psi_I) = \langle(-i)\hat{c}_a\hat{c}_{a+2d}, a = 1,\ldots,2d\rangle.$$

One can easily check that ψ_I has a covariance matrix

$$M_I = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix},$$

where each block has dimensions $2d\times 2d$. Fact 1 implies that ψ_I is a Gaussian state. Let $W = \exp(\hat{G})$ be the operator defined in Eq. (27). Define a $2d$-qubit state

$$\psi_W = (W\otimes I)\psi_I \in \mathcal{H}_{2d}. \tag{30}$$

Note that ψ_W is a Gaussian state due to Fact 3. Let

$$M_W = \begin{bmatrix} A & B \\ -B^T & D \end{bmatrix}$$

be the covariance matrix of ψ_W. Here A,B,D are some matrices of size $2d\times 2d$. Let $\Gamma_W = \langle\psi_W|\psi_W\rangle$. We shall need the following fact proved in [23].

Fact 4. Let $\psi = (M,\Gamma)$ be a Gaussian state. Then $W\psi = (M',\Gamma')$, where

$$M' = A - B(M-D)^{-1}B^T \tag{31}$$

and

$$\Gamma' = \Gamma_W\Gamma\sqrt{\det(M-D)}. \tag{32}$$

It remains to compute (M_W,Γ_W) for the two special cases $W = \hat{H}^j$ and $W = \hat{V}^j$.

We shall perform the calculation for $W = \hat{H}^j$ since both cases are quite similar. First we note that W is a product of operators acting on disjoint pairs of Majorana modes $(\hat{c}_{2a-1},\hat{c}_{2a})$. Accordingly, ψ_W is a product of states involving disjoint 4-tuples of Majorana modes $(\hat{c}_{2a-1},\hat{c}_{2a},\hat{c}_{2a-1+2d},\hat{c}_{2a+2d})$. It suffices to compute the covariance matrix and the norm for each of those 4-tuples. Equivalently, it suffices to do the calculation for $d=1$. In this case $W = G(w)$ is the single-qubit operator defined in Eq. (10). By definition, ψ_I is a two-qubit state with stabilizers $(-i)\hat{c}_1\hat{c}_3 = -Y_1X_2$ and $(-i)\hat{c}_2\hat{c}_4 = X_1Y_2$. It can be written explicitly as

$$|\psi_I\rangle = \frac{1}{\sqrt{2}}(|10\rangle + i|01\rangle).$$

Hence

$$|\psi_W\rangle \equiv (W\otimes I)|\psi_I\rangle = \frac{1}{\sqrt{2}}(w|10\rangle + i|01\rangle).$$

This state has norm

$$\Gamma_W = \langle\psi_W|\psi_W\rangle = \tfrac{1}{2}(1+w^2).$$

To compute the covariance matrix M_W we shall use a shorthand notation

$$\langle\cdot\rangle \equiv \frac{\langle\psi_W|\cdot|\psi_W\rangle}{\langle\psi_W|\psi_W\rangle}.$$

By definition,

$$(M_W)_{p,q} = \langle(-i)\hat{c}_p\hat{c}_q\rangle \quad \text{for } 1\leqslant p<q\leqslant 4.$$

A straightforward calculation shows that the only nonzero elements (with $p<q$) of M_W are

$$(M_W)_{1,2} = \langle Z_1\rangle = \frac{1-w^2}{1+w^2} \equiv t,$$
$$(M_W)_{1,3} = \langle -Y_1X_2\rangle = \frac{2w}{1+w^2} \equiv s,$$
$$(M_W)_{2,4} = \langle X_1Y_2\rangle = s,$$

and

$$(M_W)_{3,4} = \langle Z_2\rangle = -t.$$

Thus

$$M_W = \begin{bmatrix} A & B \\ -B^T & D \end{bmatrix}, \tag{33}$$

where

$$A = -D = \begin{bmatrix} 0 & t \\ -t & 0 \end{bmatrix}, \quad B = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}.$$

For an arbitrary d we just need to take a direct sum of d matrices M_W as above and take the product of d normalizing coefficients Γ_W defined above. This yields

$$\Gamma_W = \prod_{a=1}^{d}\frac{1}{2}(1+w_{e_a}^2),$$

whereas M_W is given by Eq. (33) where $A = -D = \mathcal{A}(t_1,0,t_2,0,\ldots,0,t_d)$ and $B = \mathcal{D}(s_1,s_1,\ldots,s_d,s_d)$ with

$$t_a = \frac{1-w_{e_a}^2}{1+w_{e_a}^2}, \quad s_a = \frac{2w_{e_a}}{1+w_{e_a}^2}.$$

Combining the above analysis and Fact 4 we infer that the function SimulateHorizontal(j,M,Γ) defined in Sec. V B describes how the covariance matrix and the norm of a Gaussian state change under application of the operator $\hat{H}^j$. A similar calculation shows that the function SimulateVertical(j,M,Γ) describes how the covariance matrix and the norm of a Gaussian state change under application of the operator $\hat{V}^j$. The very last step of Algorithm 1 corresponds to computing the overlap between ψ_e and the final state $\hat{H}^d\cdots\hat{V}^1\hat{H}^1\psi_e$ using Eq. (26). This completes the proof of correctness of Algorithm 1.

VI. APPROXIMATE ALGORITHM

In this section we describe an approximate algorithm for computing the coset probabilities. It is applicable to a general stochastic i.i.d. Pauli noise including the depolarizing noise. We assume some level of familiarity with matrix product states and tensor networks; see [29] or [28] for a thorough review. For the sake of completeness we summarize some basic facts about matrix product states in Sec. VI C.

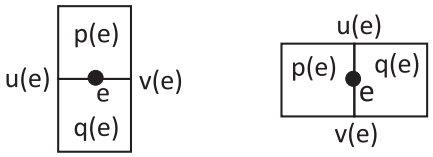


FIG. 5. Restriction of a stabilizer $g(\alpha;\beta)$ onto the edge e depends only on the variables $\alpha_{u(e)}$, $\alpha_{v(e)}$ and $\beta_{p(e)}$, $\beta_{q(e)}$.

A. Construction of the tensor network

Let $f\mathcal{G}$ be one of the cosets $\mathcal{C}_I^s,\mathcal{C}_X^s,\mathcal{C}_Y^s,\mathcal{C}_Z^s$ defined in Sec. II. Our goal is to compute the coset probability $\pi(f\mathcal{G})$. Let π_1 be any probability distribution on the single-qubit Pauli group. For example,

$$\pi_1(X)=\pi_1(Y)=\pi_1(Z)=\epsilon/3,\quad \pi_1(I)=1-\epsilon$$

for the depolarizing noise with a rate ϵ. By definition,

$$\pi(f\mathcal{G})=\sum_{g\in\mathcal{G}}\prod_e \pi_1(f_e g_e), \tag{34}$$

where the product ranges over all edges of the surface code lattice. Let us parametrize $g\in\mathcal{G}$ by binary variables $\alpha_u,\beta_p\in\{0,1\}$ associated with sites u and plaquettes p such that

$$g(\alpha;\beta)=\prod_u (A_u)^{\alpha_u}\cdot\prod_p (B_p)^{\beta_p}.$$

Here we used a convention $(B_p)^0\equiv I$ and $(A_u)^0\equiv I$. Let e be some edge of the surface code lattice with end points $u(e),v(e)$ and adjacent plaquettes $p(e),q(e)$; see Fig. 5. Let g_e be the restriction of g onto the qubit e. Clearly, g_e depends only on the bits $\alpha_{u(e)},\alpha_{v(e)}$ and $\beta_{p(e)},\beta_{q(e)}$. Thus we can write

$$g_e(\alpha;\beta)=g_e(\alpha_{u(e)},\alpha_{v(e)};\beta_{q(e)},\beta_{q(e)}),$$

where $g_e(i,j;k,l)$ is a function of just four binary variables $i,j,k,l\in\{0,1\}$. For horizontal edges located at the left or the right boundary of the lattice the variable $\alpha_{u(e)}$ or $\alpha_{v(e)}$ respectively is missing. Likewise, for horizontal edges located at the top or the bottom boundary the variable $\beta_{p(e)}$ or $\beta_{q(e)}$ respectively is missing. We arrive at

$$\pi(f\mathcal{G})=\sum_\alpha\sum_\beta T(\alpha;\beta), \tag{35}$$

where the sums range over binary strings $\alpha,\beta\in\{0,1\}^{d(d-1)}$ corresponding to all possible configurations of variables α_u,β_p and

$$T(\alpha;\beta)=\prod_e \pi_1(f_e g_e(\alpha_{u(e)},\alpha_{v(e)};\beta_{q(e)},\beta_{q(e)})). \tag{36}$$

The right-hand side of Eq. (35) coincides with the contraction value of a properly defined tensor network on a two-dimensional grid. To define this tensor network, consider the extended surface code lattice shown on Fig. 6. The extended lattice has three types of nodes which we call s nodes, h nodes, and v nodes. Each s node represents a location of a stabilizer (either a site stabilizer A_u or plaquette stabilizer B_p), while h nodes and v nodes represent code qubits located on horizontal and vertical edges of the original surface code lattice, respectively. We shall refer to edges of the extended lattice as *links* to distinguish them from edges of the original surface code lattice.

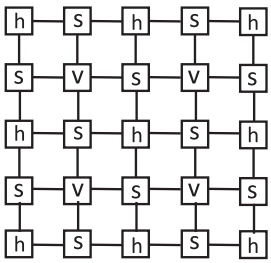


FIG. 6. Extended surface code lattice for $d=3$. Locations of stabilizers are represented by s nodes. Code qubits located on horizontal and vertical edges of the original lattice are represented by h nodes and v nodes, respectively. In general, the extended lattice has dimensions $(2d-1)\times(2d-1)$.

Consider any configuration of variables α,β and the corresponding term $T(\alpha;\beta)$ in Eq. (35). For each site stabilizer A_u let us copy the corresponding variable α_u to all links incident to the s node u. Likewise, for each plaquette stabilizer B_p let us copy the corresponding variable β_p to all links incident to the s node p. We obtain a labeling of the links by binary variables $\gamma(\alpha;\beta)$ with the property that all links incident to any s node have the same label. Let us call such a link labeling *valid*. By definition, $T(\alpha;\beta)$ is a product of terms

$$T_e(\alpha;\beta)\equiv\pi_1(f_e g_e(\alpha_{u(e)},\alpha_{v(e)};\beta_{q(e)},\beta_{q(e)}))$$

associated with h nodes and v nodes e of the extended lattice. Since α and β are uniquely determined by the link labeling $\gamma(\alpha;\beta)$, we can also write $T_e(\alpha;\beta)$ as a function of γ, that is, $T_e(\alpha;\beta)=T_e(\gamma)$. This shows that

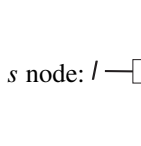
$$\pi(f\mathcal{G})=\sum_{\text{valid }\gamma}\prod_{e\in h,v} T_e(\gamma), \tag{37}$$

where the product is over all h nodes and v nodes and the sum ranges over all valid link labelings. We can now extend the sum in Eq. (37) to *all* link labelings γ by adding extra terms $T_e(\gamma)\in\{0,1\}$ associated with s nodes e such that $T_e(\gamma)=1$ iff all links incident to e have the same label and $T_e(\gamma)=0$ otherwise. We arrive at

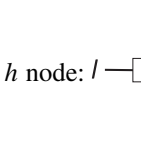
$$\pi(f\mathcal{G})=\sum_\gamma\prod_e T_e(\gamma), \tag{38}$$

Now the product ranges over all nodes of the extended lattice and the sum ranges over all link labelings. Furthermore, by construction, each term $T_e(\gamma)$ depends only on the labels of links incident to the node e. The expression in the right-hand side of Eq. (38) is known as a contraction value of the tensor network defined by the collection of tensors $T_e(\gamma)$. Tensor networks are usually represented by diagrams like the one shown on Fig. 6 such that each box on the diagram carries a tensor with several indexes. Indexes of a tensor are associated with the links emanating from the corresponding box. Diagrams representing the tensors $T_e(\gamma)$ are shown on Eqs. (39), (40), and (41). All tensor indexes i,j,k,l on these diagrams take values $0,1$. For tensors located at the boundary some of the indexes may be missing. Note that the order of arguments of g_e is interchanged in Eqs. (40) and (41). This

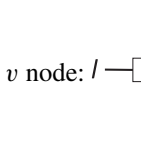
is simply because the qubits located on horizontal edges (h nodes) have site stabilizers on the left and on the right, whereas qubits located on vertical edges (v nodes) have site stabilizers on the top and on the bottom:



$$s \text{ node: } = \begin{cases} 1 & \text{if } \ i=j=k=l, \\ 0 & \text{otherwise,} \end{cases} \tag{39}$$



$$h \text{ node: } = \pi_1(f_e g_e(j,l;i,k)), \tag{40}$$



$$v \text{ node: } = \pi_1(f_e g_e(i,k;j,l)). \tag{41}$$

B. Approximate contraction algorithm

Let MPS(χ) and MPO(χ) be the set of matrix product states and matrix product operators defined on a chain of $2d-1$ qubits and having the bond dimension χ. In this section we shall identify a matrix product state (operator) with the corresponding tensor network. Consider a partition of the extended surface code lattice into columns shown on Fig. 7. Each column V^j and each internal column H^j defines a matrix product operator $\hat{V}^j \in$ MPO(2) and $\hat{H}^j \in$ MPO(2), respectively. The first and the last columns H^1, H^d define matrix product states $\hat{H}^1, \hat{H}^d \in$ MPS(2). Here we identify horizontal links of the lattice with physical indexes of MPO and MPS, while vertical links correspond to virtual indexes. By definition, contracting a consecutive pair of columns is equivalent to taking the product of the corresponding MPOs. Thus Eq. (38) can be rewritten as

$$\pi(f\mathcal{G}) = \langle \hat{H}^d | \hat{V}^{d-1} \cdots \hat{H}^2 \hat{V}^1 | \hat{H}^1 \rangle. \tag{42}$$

To approximate the right-hand side of Eq. (42) we shall employ the algorithm proposed by Murg, Verstraete, and Cirac [26,27]; see also [34,35]. The approximation accuracy of the algorithm is controlled by an integer parameter $\chi \geqslant 2$ such that

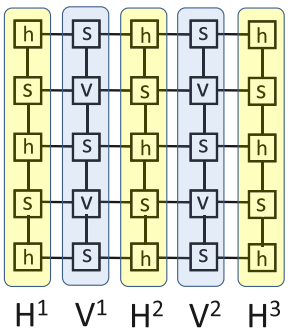


FIG. 7. (Color online) Partition of the extended lattice into "horizontal" columns $H^1, \ldots, H^d$ and "vertical" columns $V^1, \ldots, V^{d-1}$ (here $d = 3$).

the algorithm becomes exact if χ is exponentially large in d. At each step of the algorithm we maintain a state $\psi \in$ MPS(χ). Such a state can be described by a list of $2d-1$ tensors of dimension $2\times\chi\times\chi$ which requires $O(d\chi^2)$ real parameters. We begin by initializing $\psi = \hat{H}^1$. Note that $\hat{H}^1 \in$ MPS(2) $\subseteq$ MPS(χ). Each step of the algorithm updates ψ according to $\psi \to \hat{H}^j\psi$ (even steps) or $\psi \to \hat{V}^j\psi$ (odd steps). This update is realized simply by taking the product of tensors of ψ with the respective tensors of $\hat{H}^j$ or $\hat{V}^j$ which takes time $O(d\chi^2)$. Since $\hat{H}^j$ and $\hat{V}^j$ map MPS(χ) to MPS(2χ), extra measures have to be taken to reduce the bond dimension after each update. To this end we apply the truncation algorithm described in Sec. 4.5 of Ref. [29]. We shall use a function Truncate() that takes as input a state $\phi \in$ MPS(2χ) and returns a state $\psi \in$ MPS(χ) approximating ϕ. Such an approximation is obtained by computing the Schmidt decomposition of ϕ across each bipartite cut of the chain and retaining only the χ largest Schmidt coefficients. A detailed implementation of the function Truncate() is described in the next section. The last step of the algorithm is to compute the inner product between the final state $\psi \in$ MPS(χ) and $\hat{H}^d \in$ MPS(2). This can be done in time $O(d\chi^3)$ by applying the standard contraction method for MPS. As we explain in the next section, each call to the function Truncate() involves $2d-1$ QR decompositions and SVD decompositions on matrices of size $2\chi\times 2\chi$ and $2\chi\times\chi$, respectively, which takes time $O(d\chi^3)$. Since we need one truncation for each column of the lattice, the overall running time of the algorithm is $O(d^2\chi^3) = O(n\chi^3)$. The above steps can be summarized as follows:

Algorithm 2

Input: Pauli operator f
Output: Approximation to $\pi(f\mathcal{G})$

$\psi \leftarrow \hat{H}^1$
for $j = 1$ to $d-2$ do
  $\psi \leftarrow$ TRUNCATE$(\hat{V}^j\psi)$
  $\psi \leftarrow$ TRUNCATE$(\hat{H}^{j+1}\psi)$
end for
$\psi \leftarrow$ TRUNCATE$(\hat{V}^{d-1}\psi)$
return $\langle \hat{H}^d | \psi \rangle$

C. Truncation of a matrix product state

In this section we describe implementation of the function Truncate() in Algorithm 2. Our implementation closely follows Sec. 4.5 of Ref. [29]. For the sake of completeness, we begin by summarizing the necessary facts about matrix product states. Below we use a notation $L \equiv 2d-1$ for the number of qubits per column of the lattice.

A matrix product state $|\psi\rangle \in (\mathbb{C}^2)^{\otimes L}$ describing a chain of L qubits is defined by a list of $2L$ matrices $A_0(s), A_1(s)$, where $s = 1, \ldots, L$ is a qubit index (site of the chain). Any amplitude of ψ in the standard basis is expressed as a product of L matrices

$$\langle x|\psi\rangle = A_{x_1}(1)A_{x_2}(2)\cdots A_{x_L}(L), \quad x \in \{0,1\}^L. \tag{43}$$

We shall use a shorthand notation $A(s)$ for the pair of matrices $A_0(s), A_1(s)$ at some particular qubit s. Likewise A will stand

for the full matrix product state. The L-qubit state defined in Eq. (43) will be denoted $\psi(A)$. Let $r(s)$ and $c(s)$ be the number of rows and columns respectively in $A_{0,1}(s)$ [we shall always assume $A_0(s)$ and $A_1(s)$ have the same dimensions]. Since we want the product of matrices in Eq. (43) to be a 1×1 matrix (a complex number), dimensions of the matrices must satisfy

$$r(1)=1, \quad c(L)=1, \quad c(s)=r(s+1) \quad \text{for } 1\leqslant s<L.$$

A matrix product state is said to have a *bond dimension* χ iff $r(s)\leqslant\chi$ and $c(s)\leqslant\chi$ for all qubits s. Let $\mathrm{MPS}(\chi)$ be the set of all matrix product states A on L qubits with the bond dimension χ. We shall say that $A(s)$ has a *left canonical form* (LCF) or *right canonical form* (RCF) iff

$$A_0(s)^\dagger A_0(s)+A_1(s)^\dagger A_1(s)=I_{c(s)} \tag{44}$$

or

$$A_0(s)A_0(s)^\dagger+A_1(s)A_1(s)^\dagger=I_{r(s)}, \tag{45}$$

respectively. Here I_n denotes the identity matrix of size $n\times n$. The importance of LCF and RCF comes from the following lemma. Here and below we use a notation e^i for the column vector $[0,\ldots,0,1,0,\ldots,0]^\mathsf{T}$ with "1" at the ith coordinate.

Lemma 2. Suppose $A(s)$ has LCF for $s=1,\ldots,m$. For each $\alpha=1,\ldots,c(m)$ define a state $\phi_\alpha\in(\mathbb{C}^2)^{\otimes m}$ with amplitudes

$$\langle x|\phi_\alpha\rangle=A_{x_1}(1)A_{x_2}(2)\cdots A_{x_m}(m)e^\alpha, \quad x\in\{0,1\}^m. \tag{46}$$

Then ϕ_α form an orthonormal family of vectors, i.e., $\langle\phi^\beta|\phi^\alpha\rangle=\delta_{\alpha,\beta}$ for all $1\leqslant\alpha,\beta\leqslant c(m)$.

Proof. Indeed, using the definition of ϕ^α the inner product $\langle\phi^\beta|\phi^\alpha\rangle=\sum_x\langle\phi^\beta|x\rangle\langle x|\phi^\alpha\rangle$ can be written as

$$\sum_x (e^\beta)^\mathsf{T}A_{x_m}(m)^\dagger\cdots A_{x_1}(1)^\dagger A_{x_1}(1)\cdots A_{x_m}(m)e^\alpha,$$

where the sum runs over $x\in\{0,1\}^m$. The LCF at qubit 1 implies $\sum_{x_1}A_{x_1}(1)^\dagger A_{x_1}(1)=I_{c(1)}$. Hence $\langle\phi^\beta|\phi^\alpha\rangle$ is equal to

$$\sum_x (e^\beta)^\mathsf{T}A_{x_m}(m)^\dagger\cdots A_{x_2}(2)^\dagger A_{x_2}(2)\cdots A_{x_m}(m)e^\alpha,$$

where the sum runs over $x\in\{0,1\}^{m-1}$. Applying the same argument to the remaining qubits one arrives at $\langle\phi^\beta|\phi^\alpha\rangle=(e^\beta)^\mathsf{T}e^\alpha=\delta_{\alpha,\beta}$. ∎

Exactly the same arguments show that if $A(s)$ has RCF for all $s>m$ then states $\theta^\alpha\in(\mathbb{C}^2)^{\otimes(L-m)}$ with amplitudes

$$\langle y|\theta^\beta\rangle=(e^\beta)^\mathsf{T}A_{y_1}(m+1)\cdots A_{y_{L-m}}(L) \tag{47}$$

form an orthonormal family for $1\leqslant\beta\leqslant r(m+1)$.

The first step of the function Truncate is transforming all matrices $A(s)$ to LCF. We shall describe this step by a function LeftCanonical(A) that takes as input a matrix product state $A\in\mathrm{MPS}(\chi)$ and returns a pair (Γ,B), where $\Gamma\in\mathbb{C}$ is a scalar and $B\in\mathrm{MPS}(\chi)$ is a matrix product state such that $\psi(A)=\Gamma\cdot\psi(B)$ and B has LCF at every qubit. We shall define LeftCanonical(A) by the following algorithm:

```
function (Γ, B)=LeftCanonical(A)
  for s = 1 to L do
    (Q, R) ← QR-decomposition of A(s)
             as defined in Eqs (48,49)
    B_0(s) ← Q_0
    B_1(s) ← Q_1
    if s < L then
      A_0(s+1) ← R A_0(s+1)
      A_1(s+1) ← R A_1(s+1)
    else
      Γ ← R
    end if
  end for
end function
```

Let us explain the QR-decomposition step in the above algorithm and prove its correctness. Consider any qubit s and represent $A(s)$ as a block matrix

$$A(s)=\begin{bmatrix}A_0(s)\\A_1(s)\end{bmatrix}. \tag{48}$$

Note that $A(s)$ has $2r(s)$ rows and $c(s)$ columns. Let $m=\min\{c(s),2r(s)\}$. Applying the "economic" QR decomposition to $A(s)$ one gets

$$A(s)=QR, \tag{49}$$

where Q has dimensions $2r(s)\times m$, R has dimensions $m\times c(s)$, and columns of Q form an orthonormal family of vectors, that is, $Q^\dagger Q=I_m$. Finally, R is an upper triangular matrix (this property will not be important for us). Let us write

$$Q=\begin{bmatrix}Q_0\\Q_1\end{bmatrix},$$

where $Q_{0,1}$ have dimensions $r(s)\times m$. The property $Q^\dagger Q=I_m$ is equivalent to $Q_0^\dagger Q_0+Q_1^\dagger Q_1=I_m$. Hence $B(s)$ defined in the above algorithm has LCF. Note that dimensions of $B_{0,1}(s)$ may or may not be equal to the ones of $A_{0,1}(s)$. Let $A'_{0,1}(s+1)=RA_{0,1}(s+1)$ be the updated version of $A(s+1)$ defined in the algorithm. Obviously $A_x(s)A_y(s+1)=B_x(s)A'_y(s+1)$ for any $x,y=0,1$. Thus $A_{x_1}(1)\cdots A_{x_L}(L)$ is equal to

$$B_{x_1}(1)\cdots B_{x_s}(s)A'_{x_{s+1}}(s+1)A_{x_{s+2}}(s+2)\cdots A_{x_L}(L)$$

for all $x\in\{0,1\}^L$ and for all $s=1,\ldots,L-1$. The last step of the algorithm ($s=L$) applies a QR decomposition to a column vector $A(L)$, possibly updated by the previous step of the algorithm. Hence Q is a unit-norm column vector of size $2r(L)$, while R is a scalar which determines normalization of the overall state. This proves that $\psi(A)=\Gamma\cdot\psi(B)$ and B has LCF at every qubit.

Suppose the input matrix product state A has bond dimension χ. Then the computational cost of each QR decomposition is $O(\chi^3)$. Therefore, the function LeftCanonical(A) can be computed in time $O(L\chi^3)$. Since no step of the algorithm increases dimensions of the matrices, the final matrix product state B also has bond dimension χ.

We are now ready to describe the function Truncate. Choose any integer $1\leqslant m\leqslant L$ and partition the chain as $\mathcal{L}\cup m\cup\mathcal{R}$,

where

$$\mathcal{L} = \{1, \ldots, m-1\}, \quad \mathcal{R} = \{m+1, \ldots, L\}.$$

Consider a matrix product state A such that $A(s)$ has LCF for all $s \in \mathcal{L}$ and RCF for all $s \in \mathcal{R}$. Suppose also that the matrices $A_{0,1}(s)$ have dimensions at most χ for all $s \in \mathcal{R}$ and at most $\tilde{\chi}$ for all $s \in \mathcal{L}$. We assume that $\tilde{\chi} > \chi$ (we shall be interested in the case $\tilde{\chi} = 2\chi$). Using the orthonormal families of states ϕ^α and θ^α defined in Eqs. (46) and (47) one can write $\psi(A)$ as

$$\psi(A) = \sum_{\alpha=1}^{r(m)} \sum_{\beta=1}^{c(m)} \sum_{x=0,1} A_x(m)_{\alpha,\beta} |\phi^\alpha \otimes x \otimes \theta^\beta\rangle. \tag{50}$$

We shall compute the Schmidt decomposition of $\psi(A)$ with respect to the partition $\mathcal{L} \cup \{m, \mathcal{R}\}$ and truncate this decomposition by retaining only the χ largest Schmidt coefficients. To this end consider the singular value decomposition (SVD) of $A(m)$, namely,

$$A(m) \equiv [A_0(m) \,|\, A_1(m)] = USV^\dagger, \tag{51}$$

where the matrices U, S, V have dimensions

$$\dim U = r(m) \times n, \quad \dim S = n \times n,$$
$$\dim V = 2c(m) \times n,$$

with

$$n = \min\{r(m), 2c(m)\}.$$

The matrix S is diagonal such that $S_{i,i}$ is the ith largest singular value of $A(m)$. The matrices U and V are isometries, that is,

$$U^\dagger U = V^\dagger V = I_n.$$

Let us represent V as a block matrix

$$V = \begin{bmatrix} V_0 \\ V_1 \end{bmatrix}, \tag{52}$$

where V_0 and V_1 have dimensions $c(m) \times n$. Using the above SVD one can rewrite $\psi(A)$ as

$$\psi(A) = \sum_{i=1}^{n} S_{i,i} |\hat{\phi}^i\rangle \otimes |\hat{\theta}^i\rangle, \tag{53}$$

where $\hat{\phi}^i$ and $\hat{\theta}^i$ are orthonormal family of n states defined as

$$|\hat{\phi}^i\rangle = \sum_{\alpha=1}^{r(m)} U_{\alpha,i} \, |\phi^\alpha\rangle \tag{54}$$

and

$$|\hat{\theta}^i\rangle = \sum_{\beta=1}^{c(m)} (V_0^*)_{\beta,i} |0 \otimes \theta^\beta\rangle + (V_1^*)_{\beta,i} |1 \otimes \theta^\beta\rangle. \tag{55}$$

We conclude that Eq. (53) defines the Schmidt decomposition of $\psi(A)$ with respect to the partition $\mathcal{L} \cup \{m, \mathcal{R}\}$, while $S_{i,i}$ are the Schmidt coefficients. The best rank-χ approximation to $\psi(A)$ which we denote $\psi'(A)$ is obtained from Eq. (53) by retaining χ largest Schmidt coefficients, that is,

$$\psi'(A) = \sum_{i=1}^{\chi} S_{i,i} |\hat{\phi}^i\rangle \otimes |\hat{\theta}^i\rangle. \tag{56}$$

Decompose matrices U, S, V into blocks such that

$$U = [U' \,|\, U''], \quad S = \begin{bmatrix} S' & 0 \\ 0 & S'' \end{bmatrix}, \quad V' = [V' \,|\, V'']. \tag{57}$$

By definition, U', S', V' have dimensions

$$\dim U' = r(m) \times \chi, \quad \dim S' = \chi \times \chi,$$
$$\dim V' = 2c(m) \times \chi.$$

Furthermore, S' is a square diagonal matrix that contains χ largest singular values of $A(m)$, while U' and V' are isometries, that is, $(U')^\dagger U' = I_\chi$ and $(V')^\dagger V' = I_\chi$. We conclude that $\psi'(A) = \psi(A')$, where $A'(s) = A(s)$ for $s \in \mathcal{R}$ and for $s \in \mathcal{L} \setminus m$,

$$A'_{0,1}(m-1) = A_{0,1}(m-1)U'S', \quad A'(m) = (V')^\dagger.$$

The fact that V' is an isometry implies that $A'(m)$ has RCF, so we can apply the above procedure again with $\mathcal{L} = \mathcal{L} \setminus \{m-1\}$ and $\mathcal{R} = \mathcal{R} \cup \{m\}$. Starting from $m = L$ and moving towards the left boundary of the chain one can reduce the bond dimension from $\tilde{\chi}$ to χ. The above truncation algorithm can be summarized as follows:

```
function TRUNCATE(A)
    (Γ, A) ← LEFTCANONICAL(A)
    for m = L to 1 do
        (U, S, V) ← svd-decomposition of A(m)
                    defined in Eq. (51)
        U', S', V' ← submatrices of U, S, V
                     defined in Eq. (57)
        A_{0,1}(m − 1) ← A_{0,1}(m − 1)U'S'
        A_{0,1}(m) ← (V'_{0,1})†
    end for
    return Γ · A
end function
```

Here we decomposed V' into blocks V'_0 and V'_1 similar to Eq. (52).

VII. NUMERICAL RESULTS

We have studied the following combinations of noise models and decoders.

(1) X noise, ML decoder.
(2) X noise, MPS decoder.
(3) X noise, MWM decoder.
(4) Depolarizing noise, MPS decoder.
(5) Depolarizing noise, MWM decoder.

For each of the above combinations we estimated the probability of a logical error—the decoding outcome in which the recovery operator differs from the actual error by a logical Pauli operator (we do not differentiate between $\overline{X}, \overline{Y}$, or $\overline{Z}$ logical errors). The performance of each decoder was measured in terms of its error threshold and its *badness* parameter—the ratio between the logical error probabilities of a given decoder and the best available decoder for the considered noise model. Thus badness $\geqslant 1$ for any decoder with smaller values indicating better decoders. The exact ML decoder and MPS decoders were implemented as described in Sec. V and Sec. VI, respectively. The MWM decoder was implemented by a reduction from the minimum weight perfect

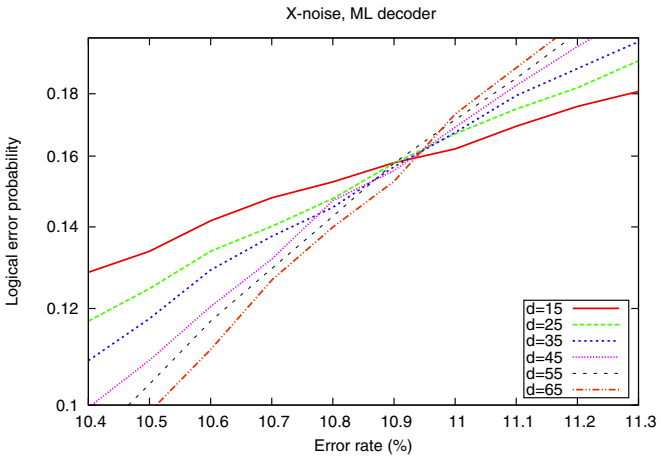


FIG. 8. (Color online) X noise: exact implementation of the ML decoder. The data suggest that the threshold error rate ϵ_0 is between 10.9% and 11%, which is in a good agreement with the estimate $\epsilon_0 = 10.93(2)$ of Ref. [24] which calculated the phase-transition point in the respective spin model. Each curve has data points at error rates $\epsilon = 10.4, 10.5, \ldots, 11.3\%$. To compute the logical error probability, at least 5000 failed error correction trials have been accumulated for each data point.

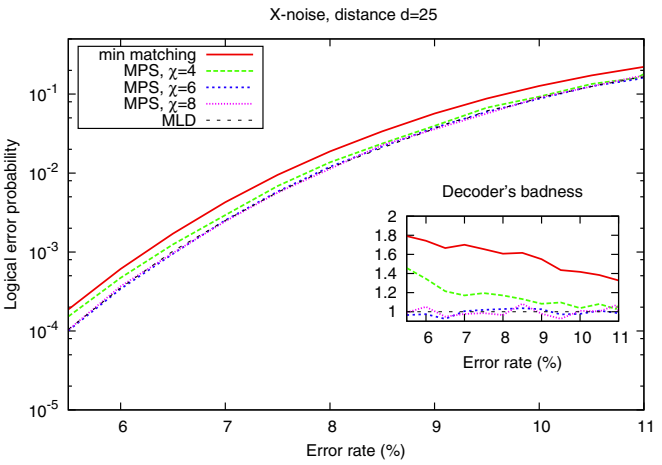


FIG. 9. (Color online) X noise: exact and approximate implementations of the ML decoder. Logical error probability as a function of the error rate ϵ is shown. The curves representing the exact MLD and MPS decoders with $\chi = 6,8$ are too close to be distinguishable on the main plot. The red curve represents the standard minimum weight matching decoder. The inset shows "badness" of various decoders as a function of the error rate. We define the badness as the ratio between logical error probabilities of a given decoder and the optimal decoder (MLD). Each curve has data points at error rates $\epsilon = 5, 5.5, 6, \ldots, 11\%$. To compute the logical error probability, at least 1000 failed error correction trials have been accumulated for each data point.

matching problem to the maximum weight matching problem as described in Ref. [36].

Let us first discuss our results for the X noise. The threshold error rate ϵ_0 of the ML decoder coincides with the critical density of antiferromagnetic bonds in the random-bond Ising model on the Nishimori line [2]. The latter has been estimated numerically by Mertz and Chalker [24] who found $\epsilon_0 = 10.93(2)\%$. Our data shown at Fig. 8 suggest that $10.9\% \leqslant \epsilon_0 \leqslant 11\%$, which is in a good agreement with the estimate of Ref. [24]. For comparison, the MWM decoder is known to have the threshold $\epsilon_0 \approx 10.31\%$; see [37].

The performance of different decoders for a fixed code distance $d = 25$ and a wide range of error rates is shown at Fig. 9. We observed that the MWM decoder remains nearly optimal for all simulated error rates with the badness parameter $\leqslant 2$, even though for these error rates the logical error probability changes by several orders of magnitude. The slight difference between MLD and the MWM decoder can be explained by the fact that the latter ignores the error degeneracy [32]. The data shown on Fig. 9 suggests that for X noise ignoring the error degeneracy does not have a significant impact on the performance, even for large error rates and large code distances.

Perhaps more surprisingly, Fig. 9 demonstrates that the MPS decoder with a relatively small bond dimension $\chi = 6,8$ is virtually indistinguishable from the optimal one in terms of the logical error probability. This serves as a numerical proof of correctness for the MPS decoder.

We observed numerically that the exact MLD algorithm described in Sec. V becomes very sensitive to rounding errors in the regime of large code distances and small error rates. One way to suppress rounding errors is to enforce an orthogonality condition $M^T M = I$ on the covariance matrix M in Algorithm 1. The orthogonality condition is satisfied automatically if all arithmetic operations are perfect (because M represents a covariance matrix of a pure Gaussian state; see Sec. V D for details). In practice, we observed that the orthogonality can be quickly lost if no special measures are taken. A simple and computationally cheap solution of the above problem is to compute the QR decomposition $M = QR$, where Q is an orthogonal matrix and R is an upper-triangular matrix. Note that $M^T M = I$ is possible only if R is a diagonal matrix with entries ± 1 on the diagonal. This form of R can be easily enforced by setting all off-diagonal entries of R to zero and replacing each diagonal entry $R_{i,i}$ by the sign of $R_{i,i}$. Let $\tilde{R}$ be the resulting diagonal matrix. We found that replacing M by $M' \equiv [Q\tilde{R} - (Q\tilde{R})^T]/2$ after each call to the functions SimulateHorizontal and SimulateVertical in Algorithm 1 makes the algorithm more stable against rounding errors.

Let us now discuss the depolarizing noise. In this case we only have an approximate implementation of MLD with no direct means of estimating the approximation precision. Hence the first natural question is whether the MPS decoder with a fixed bond dimension χ has a nonzero error threshold ϵ_0. Our data suggests (although not conclusively) that the answer is "yes." Most importantly, we observed an exponential decay of the logical error probability as a function of the code distance d for a fixed error rate, see Fig. 10, where we used $\chi = 6$. Assuming that the observed decay does not saturate for larger d, the data shown at Fig. 10 gives a lower bound $\epsilon_0 \geqslant 14\%$. The logical error probability as a function of the error rate for a fixed d is shown on Fig. 11 which also exhibits a typical thresholdlike behavior and suggests that $17\% \leqslant \epsilon_0 \leqslant 18.5\%$. Previously studied approximate versions

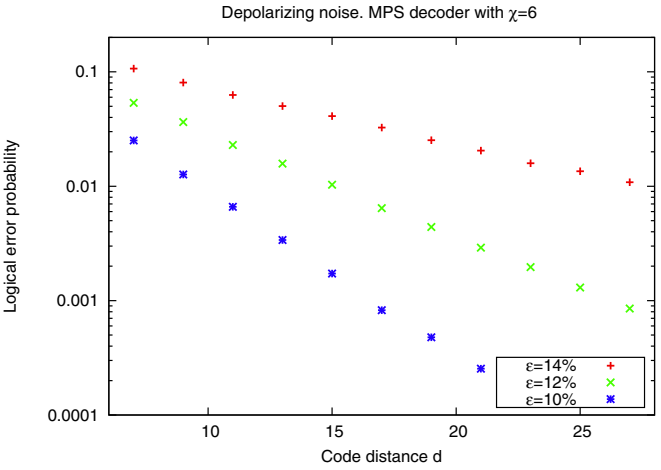


FIG. 10. (Color online) Depolarizing noise: logical error probability of the MPS decoder with $\chi = 6$ as a function of the code distance d for a fixed error rate $\epsilon = 10,12,14\%$. To compute the logical error probability, at least 1000 failed error correction trials have been accumulated for each data point.

of MLD such as the renormalization group decoder [11] and the Markov chain decoder [12], as well the MWM decoder [38] have error thresholds between 15% and 16%. The threshold of the exact ML decoder corresponding to the phase-transition point in the disordered eight-vertex Ising model is known to be $\epsilon_0 \approx 18.9(3)\%$; see Ref. [10]. Since the correlation length of the Ising model diverges at the phase-transition point, we expect that the MPS decoder can only achieve this optimal threshold if the bond dimension χ is a growing function of the code distance d.

The performance of different decoders for a fixed code distance $d = 25$ and a wide range of error rates is shown at Fig. 12. In a striking contrast with the analogous X-noise data, we observed that the MWM decoder becomes highly

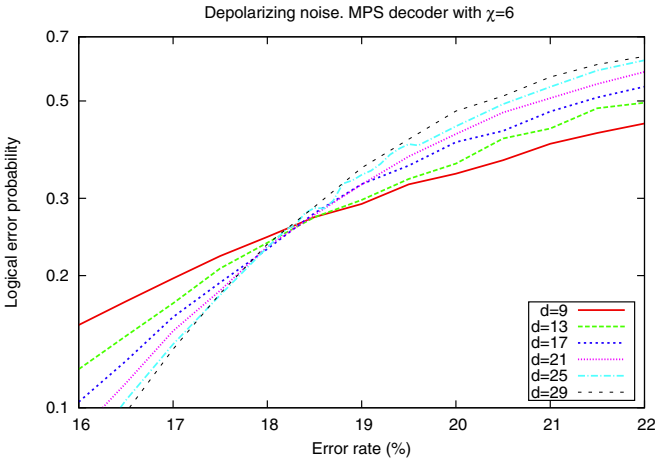


FIG. 11. (Color online) Depolarizing noise: logical error probability of the MPS decoder with $\chi = 6$ as a function of the error rate ϵ. Assuming a nonzero error threshold ϵ_0, the data suggest that $17\% \leqslant \epsilon_0 \leqslant 18.5\%$. To compute the logical error probability, at least 5000 failed error correction trials have been accumulated for each data point.

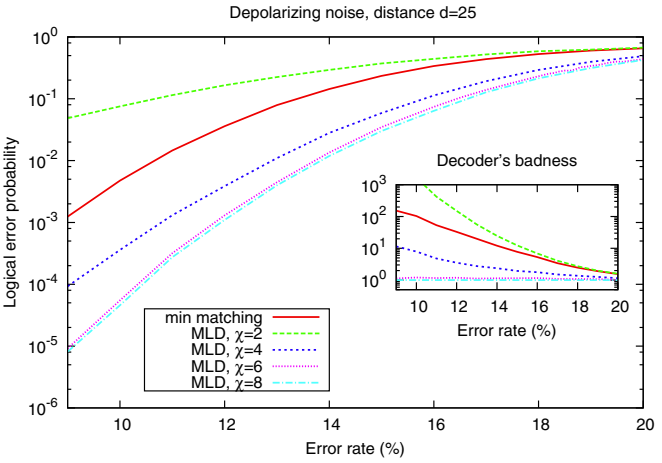


FIG. 12. (Color online) Depolarizing noise: approximate implementations of the ML decoder. Logical error probability as a function of the error rate ϵ is shown. The red curve represents the minimum weight matching decoder. The inset shows "badness" of various decoders as a function of the error rate. We define the badness as the ratio between logical error probabilities of a given decoder and the best decoder (MPS decoder with $\chi = 8$). Each curve has data points at error rates $\epsilon = 9,10,\ldots,20\%$. To compute the logical error probability, at least 1000 failed error correction trials have been accumulated for each data point.

nonoptimal in the regime of small error rates with the badness parameter above 100. This can be attributed to the fact that MWM decoder often fails to find the minimum weight error consistent with the syndrome since it ignores correlations between X and Z errors [33]. We also observed that the logical error probability of MPS decoders converges very quickly as one increases the bond dimension. The data shown on Fig. 12 indicates that the MPS decoder with $\chi = 6$ is nearly optimal for all error rates and all code distances $d \leqslant 25$.

While the logical error probability is the most natural figure of merit, one may also ask how well the MPS-based algorithm with a small bond dimension χ approximates the coset probabilities for some fixed syndrome. For simplicity, we considered the trivial syndrome, that is, the cosets $\mathcal{G}, \overline{X}\mathcal{G}, \overline{Y}\mathcal{G}$, and $\overline{Z}\mathcal{G}$. We observed a very fast convergence for the most likely coset and a poor convergence for the remaining cosets; see Tables I and II. Since the only goal of the decoder is to identify the most likely coset, the slower convergence for some of the unlikely cosets might not be a serious drawback.

TABLE I. X noise: probabilities of the two cosets computed by the MPS algorithm. The simulation parameters are $\epsilon = 5\%$ and $d = 25$. The exact values of the coset probabilities are $\pi(\mathcal{G}) = 1.78283\times10^{-27}$ and $\pi(\overline{X}\mathcal{G}) = 5.58438\times10^{-57}$.

χ	$\pi(\mathcal{G})\times10^{27}$	$\pi(\bar{X}\mathcal{G})\times10^{57}$
2	1.78275	4.72777
3	1.78277	5.52579
4	1.78283	5.80294
5	1.78283	6.03204

TABLE II. Depolarizing noise: probabilities of the four cosets computed by the MPS algorithm. The simulation parameters are $\epsilon = 10\%$ and $d = 25$.

χ	$\pi(\mathcal{G})\times 10^{55}$	$\pi(\bar{X}\mathcal{G})\times 10^{89}$	$\pi(\bar{Y}\mathcal{G})\times 10^{122}$	$\pi(\bar{Z}\mathcal{G})\times 10^{90}$
2	1.11782	2.81823	36.0410	1.64802
3	1.11781	2.81777	7.62958	1.70803
4	1.11781	2.81781	2.79984	1.78193
5	1.11781	2.81781	3.24487	2.94628

The MPS decoder offers a lot of possibilities for improvement. One rather obvious improvement (employed in the above simulations) is to use a single run of Algorithm 2 to compute two different coset probabilities. Indeed, suppose we choose the logical operator $\overline{Z}$ supported in the rightmost column of the lattice denoted H^d on Fig. 7. Then the tensor networks constructed for the cosets $\mathcal{C}_I^s$ and $\mathcal{C}_Z^s$ are exactly the same except for the column H^d. Since we contract the network column by column starting from the leftmost column H^1, the difference between the two cosets manifests itself only in the very last step of Algorithm 2 (computing the inner product $\langle \hat{H}^d|\psi\rangle$). Since this step takes a negligible time compared with the rest of the algorithm, it makes sense to compute both probabilities $\pi(\mathcal{C}_I^s)$ and $\pi(\mathcal{C}_Z^s)$ by performing a single network contraction. The same observation applies to the probabilities $\pi(\mathcal{C}_X^s)$ and $\pi(\mathcal{C}_Y^s)$. We also expect that a choice of the standard error $f(s)$ consistent with the syndrome s may affect the convergence of the algorithm. While we have chosen $f(s)$ by connecting each syndrome to the left-top boundary, it may be advantageous to choose $f(s)$ as a small-weight error, for example, using the MWM decoder. Finally, a challenging open problem is how to extend the MPS decoder to noisy syndrome extraction. A naive extension would require a contraction of a 3D tensor network. We anticipate that this problem can be attacked using recently developed algorithms for simulating 2D quantum systems based on projected entangled pairs States (PEPS); see [26,27].

ACKNOWLEDGMENTS

We would like to thank Graeme Smith and John Smolin for helpful comments. Computational resources for this work were provided by IBM Blue Gene Watson supercomputer center. S.B. acknowledges NSF Grant No. CCF-1110941.

[1] A. Kitaev, Ann. Phys. (NY) **303**, 2 (2003).
[2] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, J. Math. Phys. **43**, 4452 (2002).
[3] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).
[4] J. Ghosh, A. Fowler, and M. Geller, Phys. Rev. A **86**, 062318 (2012).
[5] J. Chow *et al.*, Nat. Commun. **5**, 4015 (2014).
[6] R. Barends *et al.*, Nature (London) **508**, 500 (2014).
[7] H. G. Katzgraber, H. Bombin, and M. A. Martin-Delgado, Phys. Rev. Lett. **103**, 090501 (2009).
[8] H. Bombin, Phys. Rev. A **81**, 032301 (2010).
[9] J. R. Wootton and D. Loss, Phys. Rev. Lett. **109**, 160503 (2012).
[10] H. Bombin, R. Andrist, M. Ohzeki, H. Katzgraber, and M. Martin-Delgado, Phys. Rev. X **2**, 021004 (2012).
[11] G. Duclos-Cianci and D. Poulin, Phys. Rev. Lett. **104**, 050504 (2010).
[12] A. Hutter, J. R. Wootton, and D. Loss, Phys. Rev. A **89**, 022326 (2014).
[13] D. Poulin, Phys. Rev. A **74**, 052333 (2006).
[14] L. Valiant, SIAM J. Comput. **31**, 1229 (2002).
[15] L. Valiant, SIAM J. Comput. **37**, 1565 (2008).
[16] J.-Y. Cai and V. Choudhary, in *Theory and Applications of Models of Computation* (Springer, New York, 2006), pp. 248–261.
[17] S. Bravyi, Contemp. Math. **482**, 179 (2009).
[18] E. Knill, arXiv:quant-ph/0108033.
[19] B. M. Terhal and D. P. DiVincenzo, Phys. Rev. A **65**, 032325 (2002).
[20] M. E. Fisher, Phys. Rev. **124**, 1664 (1961).
[21] P. W. Kasteleyn, Physica **27**, 1209 (1961).
[22] H. Temperley and M. Fisher, Philos. Mag. **6**, 1061 (1961).
[23] S. Bravyi, Quant. Inf. Comput. **5**, 216 (2005).
[24] F. Merz and J. T. Chalker, Phys. Rev. B **65**, 054425 (2002).
[25] G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003).
[26] F. Verstraete and J. I. Cirac, arXiv:cond-mat/0407066.
[27] V. Murg, F. Verstraete, and J. I. Cirac, Phys. Rev. A **75**, 033605 (2007).
[28] F. Verstraete, V. Murg, and J. I. Cirac, Adv. Phys. **57**, 143 (2008).
[29] U. Schollwöck, Ann. Phys. (NY) **326**, 96 (2011).
[30] It should be emphasized that the MPS-based decoder is applicable to any noise model that can be described by a stochastic i.i.d. Pauli noise. In contrast, the decoder based on matchgates is only applicable to noise models with independent bit-flip and phase-flip errors.
[31] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg, Phys. Rev. A **83**, 020302 (2011).
[32] T. Stace and S. Barrett, Phys. Rev. A **81**, 022317 (2010).
[33] A. Fowler, arXiv:1310.0863.
[34] S. Östlund and S. Rommer, Phys. Rev. Lett. **75**, 3537 (1995).
[35] S. R. White, Phys. Rev. Lett. **69**, 2863 (1992).
[36] S. Bravyi and A. Vargo, Phys. Rev. A **88**, 062308 (2013).
[37] C. Wang, J. Harrington, and J. Preskill, Ann. Phys. (NY) **303**, 31 (2003).
[38] D. Wang, A. Fowler, A. Stephens, and L. Hollenberg, Quant. Inf. Comput. **10**, 456 (2010).