High Threshold Error Correction for the Surface Code

James R. Wootton and Daniel Loss

High threshold error correction for the surface code

James R. Wootton and Daniel Loss

Department of Physics, University of Basel, Klingelbergstrasse 82, CH-4056 Basel, Switzerland

An algorithm is presented for error correction in the surface code quantum memory. This is shown to correct depolarizing noise up to a threshold error rate of 18.5%, exceeding previous results and coming close to the upper bound of 18.9%. The time complexity of the algorithm is found to be polynomial with error suppression, allowing efficient error correction for codes of realistic sizes.

Introduction: Topological error correcting codes, and the topological quantum computation that they may be used for, have attracted wide attention in recent years [1–5]. As such, it is important to determine the threshold error rates for realistic error models and find efficient error correction algorithms to achieve them. The most studied, and most realistic topological error correcting codes are the surface codes [1, 2], and the most realistic error model that is well-studied is that of depolarizing noise. The application of this noise model to a surface code induces correlations between different kinds of topological defects. Thus far, error correction algorithms have only been found that correct up to an error threshold of 16.5%, the upper bound achievable when the correlations are ignored [6–8]. Here we present an efficient algorithm that can correct beyond this bound. A threshold of 18.5% is found, falling only a little short of the recently established 18.9% limit [9, 10].

The planar code: The algorithm presented below is designed to correct errors in the the planar code, the planar variant of Kitaev's surface codes [1, 2]. The code is defined on the spin lattice of Fig. 1, where a spin-1/2 particle is placed on each vertex. The following Hermitian operators are then defined around each plaquette of the lattice,

$$A_s = \prod_{i \in s} \sigma_i^x, \ \ B_p = \prod_{i \in p} \sigma_i^z. \tag{1}$$

These operators determine the anyonic occupation of their corresponding plaquettes, with so-called flux anyons on the *p*-plaquettes and charge anyons on the *s*-plaquettes. Since the operators mutually commute, they also form the stabilizers of a stabilizer code. The anyonic vacuum is the corresponding stabilizer space and the anyon configuration is the syndrome. The code can store a single qubit, whose state is determined by the anyonic occupations of the edges. The X (Z) basis of the stored qubit may be chosen such that the $|+\rangle$ ($|0\rangle$) state corresponds to the vacuum on the top (left) edge and $|-\rangle$ ($|1\rangle$) corresponds to a flux (charge) anyon. The effect of errors on the spins is to create and move anyons, causing logical errors when they are moved off the edges.

Depolarizing noise: The error model considered in this study is that of single qubit depolarizing noise. This is
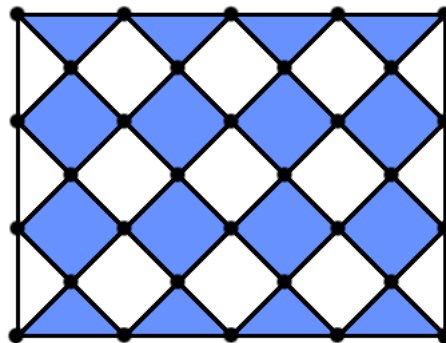

FIG. 1. The spin lattice on which an $L \times L$ planar code is defined, with *s*-plaquettes shown in blue and *p*-plaquettes in white. A spin-1/2 particle resides on each vertex. The linear size L is characterized by the number of *s*-plaquettes along each side, with $L = 4$ in this case.

characterized by an error rate, p, which is taken to be equal for all spins. The probability that no error occurs on a spin is $1 - p$. Otherwise, a σ^x, σ^y or σ^z error is applied, each with probability $p/3$. Such noise therefore takes an arbitrary single qubit state ρ and transforms it to,

$$D_p(\rho) = (1 - p)\rho + \sum_{\alpha = x,y,z} (p/3)\sigma^\alpha \rho \, \sigma^\alpha. \tag{2}$$

In the planar code, such noise results in correlations between the configurations of charge and flux anyons. Should these be ignored, error correction can be achieved so long as the probability that either a σ^x or a σ^y error occurs (or equivalently a σ^z or a σ^y error) is less than around 11% [2]. This gives a threshold of $p_c \approx 16.5\%$. If the correlations are taken into account, the threshold increases to $p_c \approx 18.9\%$ [9, 10].

Error correction: Suppose a planar code, initially prepared in a state of the stabilizer space, is subject to depolarizing noise with a known rate p. Given the resulting anyon configuration (measurement of which we assume to be perfect), the job of error correction is to determine which of the four possible logical errors was caused by the physical errors.

Let us use e to denote a configuration of errors, which

records whether $\mathbb{1}$, σ_x, σ_y or σ_z has occurred on each physical spin. Let us also use A to denote a configuration of anyons and E to denote the logical error ($\mathbb{1}$, X, Y or Z) that has occurred on the encoded qubit. Each e is consistent with a unique A and a unique E, so let us also use A and E to denote the set of e consistent with the anyon configuration A and logical error E, respectively. Given an anyon configuration A after measurement of the stabilizers, the probability for each logical error is,

$$P(E|A) = \sum_{e \in A \cap E} P(e|A). \qquad (3)$$

Here $P(e|A)$ is the probability that the error configuration e occurred given that the anyon configuration is A, etc. It can then be assumed that whichever E is most likely is that which occurred, and error correction can be performed accordingly. For any $p < p_c$, this error correction procedure succeeds with a probability that tends to unity as $L \to \infty$. For $p > p_c$ the success probability tends to $1/2$ in this limit, making error correction no better than guessing.

Note that $P(e|A)$ can be related to the unconditioned probabilities of e and A by $P(e|A) = P(e)/P(A)$. Since $P(A)$ is a common factor for all E, it does not need to be calculated in order to determine which of the $P(E|A)$ is greater, and hence which E is most likely. The $P(e)$ may be calculated easily. For depolarizing noise $P(e) = (1 - p)^{1-n_e}(p/3)^{n_e}$, where n_e is the number of spins on which a σ_x, σ_y or σ_z has occurred on the error configuration e. The number of error configurations consistent with any anyon configuration is 2^N, where $N = 2L^2 - 1$ is the total number of plaquettes in the code. Calculating the $P(E|A)$ using a brute force approach will therefore take a time that is exponential with the system size. In fact, the scaling of this is so bad that no existent computer could correct an $L = 11$ planar code in less than the age of the universe. As such, approximate methods are used to determine the most likely logical error for any anyon configuration. These achieve thresholds that are lower than the ideal case, but run for realistic time-scales [6–8].

The algorithm presented here uses a Markov chain Monte Carlo method to sample error configurations from the distribution $P(e|A)$. By taking many such samples, the probabilities $P(E|A)$ may then be approximated and hence the most likely logical error found. The most straightforward way to carry out this procedure, given an anyon configuration A, is using the Metropolis method as follows [11]. First a pattern of errors $e_0 \in A$ is generated randomly. This can be done in $O(L^2)$ time by first placing errors such that all anyons are connected, and then randomly applying each of the stabilizer. The first step ensures that e_0 is within A. The second ensures that it is random, since application of stabilizers deforms the error configuration without changing the anyon configuration. Once e_0 is generated, it can be used to generate a second

configuration, $e_1 \in A$. To do this, a random change is made to e_0 to create a configuration $e_0' \in A$. The ratio,

$$r(e_0, e_0') = \frac{P(e_0'|A)}{P(e_0|A)} = \left(\frac{p/3}{1-p}\right)^{n_{e_0'} - n_{e_0}}, \qquad (4)$$

is then determined. If $r(e, e') > 1$, we set $e_1 = e_0'$. Otherwise we set $e_1 = e_0'$ with probability $r(e, e')$ and $e_1 = e_0$ with probability $1 - r(e, e')$. This process then continues until the sequence of e_n converges, at which point they will be generated according to the distribution $P(e|A)$ [11].

The most intuitive method that could be used to generate each e_n' from each e_n is to randomly pick a stabilizer and apply it. This will cause an $O(1)$ change in the number of errors and hence yield an $r(e_n, e_n')$ of $O(1)$. However, only making such changes means that only error configurations corresponding to the same E as e_0 will be generated. Additional changes in which logical operators spanning the code can be randomly applied must therefore also be made, such that configurations from all E are sampled from. However, these will add $O(L)$ errors to any configuration on which they are applied, resulting in $r(e_n, e_n') = O(\exp{-L})$. Since the acceptance of such changes is exponentially small, the time taken to convergence will be at least $O(\exp L)$. Some additional methods are therefore required to avoid this source of inefficiency.

A solution to the problem is to use parallel tempering [12]. For this, many Markov chains such as that described above are run in parallel. Let us use N_c to denote the number of such chains, and restrict it to being odd. The first chain (which we will refer to as the bottom chain) works exactly as described above. Each e_n' is generated from e_n by application of a random stabilizer. No logical operators are applied to change the value of E. The second chain works in the same way, except for a difference in the calculation of the $r(e_n, e_n')$. Instead of using the error rate p when calculating the $P(e)$, a slightly higher error rate $p_2 = p + \Delta$ is used, where $\Delta = (0.75 - p)/(N_c - 1)$. Similarly the mth chain will use an error rate of $p_m = p + (m-1)\Delta$. Using this prescription, the N_cth chain (which we will refer to as the top chain) has $p_{N_c} = 0.75$, and so $r(e_n, e_n') = 1$ in all cases. As such, we need not restrict each e_n' for this chain to be only an $O(1)$ change away from e_n. Accordingly, the e_n' are generated randomly and independently from the e_n by randomly applying all stabilizers and logical operators. It is therefore in the top chain, and only the top chain, where the value of E changes.

The randomness in E generated in the top chain is introduced to the rest of them as follows. After running each chain for a certain number of iterations, swaps between neighbouring chains are attempted. For a swap between chains m and $m+1$, the ratio

$$r(e^m, e^{m+1}) = \left(\frac{p_m}{p_{m+1}} \frac{1 - p_{m+1}}{1 - p_m}\right)^{n_{e^{m+1}} - n_{e^m}}, \qquad (5)$$
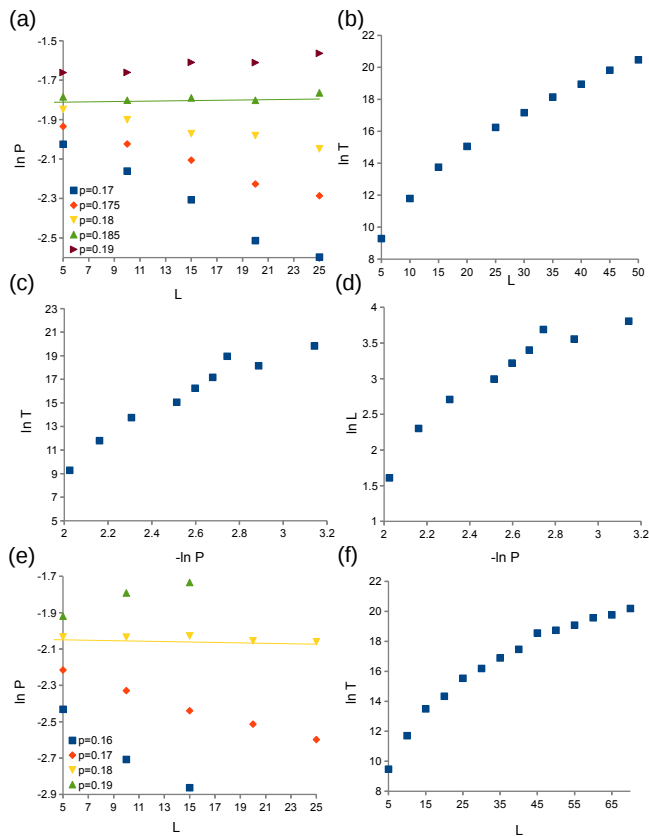
FIG. 2. Results for the first variant of the algorithm are presented in (a), (b), (c) and (d). Those for the second are in (e) and (f). Both were run using the nearest odd integer to L for N_c, ten metropolis iterations per step, $\epsilon = 0.1$ and TOPS = 10. The first used SEQ = 2 and the second used SEQ = 10. (a) and (e) show plots of the logical bit flip error rate after correction, P, against linear system size, L. Each point was averaged over 10^4 samples. Fit lines for the threshold values are shown as a guide to the eye. (b) and (f) show plots of T, the number of steps required by the algorithm before convergence, against L. Since only the order of magnitude of the run-time is important, the number of samples averaged over was reduced to between 10 and 100, allowing higher system sizes to be probed. The results in (c) and (d) are both for the first variant, and show what T and L are required to achieve a logical error rate P.

system size. This uses the number of steps required by the algorithm before convergence, T, with data obtained for $p = 0.17$. It is found that $\log T$ scales sublinearly with L, and hence T scales subexponentially with L. Each step requires $O(\log L)$ actions on $N_c = O(L)$ chains. The former is required to generate the random numbers that choose which of the $O(L^2)$ stabilizers to apply during the Metropolis procedure. The the total time complexity of the process is therefore also subexponential in L. Note that the second variant has a more efficient scaling with system size than the first, but achieves lower logical error rates. This variant therefore balances an increase in

efficiency with a decrease in effectiveness. This makes it more useful than the first variant in theoretical studies, since it allows higher system sizes to be probed, but less useful when actually performing error correction.

In Fig. 2 (c) and (d) the time and system size complexity of the first variant are considered as a function of the acheived logical error rate, P. As well as a combination of the data in Fig. 2 (a) and (b), logical error rates were also obtained for system sizes from $L = 30$ to $L = 45$ (also at $p = 0.17$). Due to the prohibitive runtime for such sizes, this data was not averaged over a fixed number of samples, but instead run until a fixed number of logical errors occurred. The number used for this was 10 (though the number of samples is much greater than this). The plots show that $\log T$ and $\log L$ grow no faster than linearly with $-\ln P$, meaning T and L grow no faster than polynomially with P^{-1}. In fact, the logarithms seems to scale sublinearly with $-\ln P$, meaning a similarly sublinear scaling of T and L with P^{-1}.

Conclusions: We have presented an algorithm for error correction in the planar code. It is demonstrated that this achieves thresholds higher than existing algorithms, approaching the theoretical bounds. The efficiency is shown to be polynomial with error suppression. This allows effective error correction for applications where the classical post-processing may be left until readout, such as a planar code quantum memory. The method on which this algorithm is based, Markov chain Monte Carlo, is not limited to the code and error model presented here. Our work therefore forms a foundation on which error correction algorithms for other topological codes and error models may be built. Future work will be dedicated to further development of the the algorithm, to increase both the threshold and the efficiency. This will allow it to be applied to cases for which classical post-processing cannot be postponed, such as quantum computation. Also, application of the algorithm to the case of noisy stabilizer measurements is sure to yield important results for this physically realistic case.

[1] A. Kitaev, Annals Phys. **303**, 2 (2003).
[2] E. Dennis, A. Kitaev, A. Landahl, J. Preskill, J. Math. Phys. **43**, 4452 (2002).
[3] M. H. Freedman, A. Kitaev, M. J. Larsen, and Z. Wang, Bull. Amer. Math. Soc., **40**, 31 (2003).
[4] R. Raussendorf and J. Harrington, Phys. Rev. Lett. 98, 190504 (2007).
[5] G. Brennen and J. K. Pachos, Proc. R. Soc. London, A **464**, 2089 (2008).

[6] J. Edmonds, Can. J. Math., **17**, 449 (1965).

[7] D. S. Wang, A. G. Fowler, A. M. Stephens, L. C. L. Hollenberg, Quant. Inf. and Comp. **10**, 456 (2010).

[8] G. Duclos-Cianci and D. Poulin, Phys. Rev. Lett. **104**, 050504 (2010).

[9] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, M. A. Martin-Delgado, arXiv:1202.1852 (2012).

[10] M. Ohzeki, arXiv:1202.2593 (2012).

[11] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, (1953). J. Chem. Phys., **21**, 1087 (1953).

[12] D. J. Earl and M. W. Deem, Phys. Chem. Chem. Phys., **7**, 3910 (2005).

[13] M. K. Cowles and B. P. Carlin, J. Amer. Stat. Assoc., **91**, 883 (1996).

[14] A. G. Fowler, A. C. Whiteside, L. C. L. Hollenberg, arXiv:1202.5602 (2012).