Compare the pair: Rotated versus unrotated surface codes at equal logical error rates

Anthony Ryan O'Rourke *
Centre for Quantum Software and Information, University of Technology Sydney, Sydney, New South Wales 2007, Australia

Simon Devitt
Centre for Quantum Software and Information, University of Technology Sydney, Sydney, New South Wales 2007, Australia and InstituteQ, Aalto University, 02150 Espoo, Finland

(Received 14 October 2024; accepted 5 May 2025; published 18 July 2025)

Practical quantum computers will require resource-efficient error-correcting codes. The rotated surface code uses approximately half the number of qubits as the unrotated surface code to create a logical qubit with the same error-correcting distance. However, instead of distance, a more useful qubit-saving metric would be based on logical error rates. In this work we find the well-below-threshold scaling of logical to physical error rates under circuit-level noise for both codes at high odd and even distances and then compare the number of qubits used by each code to achieve equal logical error rates. We perform Monte Carlo sampling of memory experiment circuits with all valid CNOT orders using the stabilizer simulator Stim and the uncorrelated minimum-weight perfect matching decoder PyMatching 2. We find that the rotated code uses about 74% the number of qubits used by the unrotated code to achieve a logical error rate of $p_L = 10^{-12}$ at the operational physical error rate of $p = 10^{-3}$. The ratio remains ≈75% for p values within a factor of two of $p = 10^{-3}$ for all useful p_L. Our work finds the low-p_L scaling of the surface code and clarifies the qubit savings provided by the rotated surface code, providing numerical justification for its use in future implementations of the surface code.

DOI: 10.1103/PhysRevResearch.7.033074

I. INTRODUCTION

Certain algorithms can be run on quantum computers with far lower processing time and resource requirements than on classical computers [1,2]. Noise processes impeding this quantum advantage can come from unwanted environmental interactions and the analog nature of quantum operations. To reliably implement a quantum algorithm requires *fault-tolerant* [3,4] quantum operations and a quantum error-correcting code (QECC), which is a set of quantum states that enable the detection and correction of errors [5]. This involves encoding a qubit as a *logical qubit* of the QECC.

Minimizing the resources required by a QECC will help realize a fault-tolerant quantum computer, which is theoretically achievable if the physical qubits which make up its logical qubits experience only finitely correlated errors that occur below some probability threshold, p_{th}.

The surface code [6] is a stabilizer code [3] which has one of the highest thresholds of any QECC and which was used in an experiment demonstrating that increasing the size of a logical qubit decreases its logical error rate (p_L) [7]. This experiment used the *rotated* surface code [8,9], which uses about half the number of physical qubits as the *unrotated* surface code [10] to create a logical qubit of the same error-correcting distance, d. However, while the distance achieved is equal, the rate of logical error suppression is not. At the same d and under uniform depolarizing noise the unrotated code outperforms the rotated, achieving a lower p_L [11,12] and slightly higher threshold, likely because it has fewer minimum-weight paths that form logical errors (as Sec. II D) [13].

The aim of our work is to provide numerical justification for the use of either the rotated or unrotated surface code by comparing the scaling of their respective p_L values under two *circuit-level* noise models, which assume every quantum operation can be faulty, and a minimum-weight perfect matching decoder [14]. We quantify the qubit saving that the rotated code provides not in achieving the same d as the unrotated code but in achieving the same p_L. Due to its relevance to circuit-level noise, our work also investigates the effect of varying the CNOT orders in the stabilizer measurement circuits.

In Sec. II we provide a background to quantum error correction, the rotated and unrotated surface codes, the order of two-qubit gates in stabilizer measurement circuits, and a review of prior work leading to ours. Section III describes our methods. We present and discuss our results in Sec. IV before concluding in Sec. V.

II. BACKGROUND

A. The surface code

The surface code is a topological stabilizer QECC [3]. It was first introduced as the toric code [6], then with boundaries as the unrotated (planar) surface code [10], and finally reformulated as the rotated surface code [8,9], which

*Contact author: anthony.orourke@student.uts.edu.au

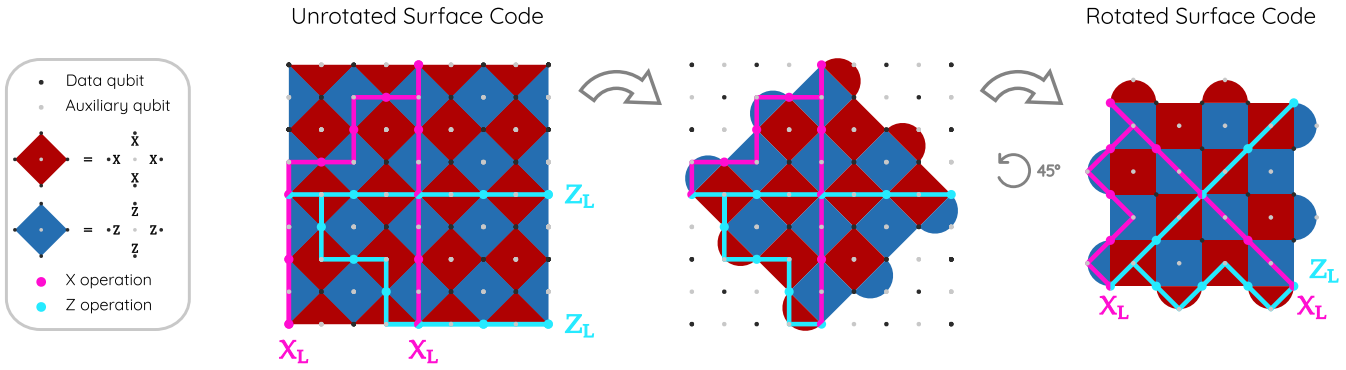


FIG. 1. The stabilizer generators of a $d = 5$, unrotated surface code logical qubit and its conceptual rotation to a $d = 5$ rotated code. $X(Z)$-type stabilizer generators are tensor products of up to four Pauli X (Z) operations on neighboring data qubits. Stabilizer measurements are performed using auxiliary qubits. A single qubit error is visualized as a chain linking the two auxiliary qubits of stabilizers it anticommutes with and hence cause to give -1 measurements results. Multiple errors form chains anticommute with stabilizers at their end points, unless the end point is at a boundary. A logical operator X_L (Z_L) is any chain that links boundaries of $X(Z)$-type boundaries. This requires at least d X (Z) operations.

requires about half the number of qubits as the unrotated code for the same d, as shown in Table I.

The basis states of the surface code's logical qubit are superpositions of entangled *data qubits*. The minimum number of single-qubit operations to transform from one logical basis state to the other is the *distance* d. This transformation is a logical operation, or error if unintended, and the code cannot detect it. Continuous noise on qubits is discretized in stabilizer codes by projective measurements of the code *stabilizers* [3]. These are usually tensor products of the Pauli operators, $\mathbb{1}$, X, Y, and Z, and are unitary and Hermitian operators whose ± 1 eigenvalues or measurement outcomes, the combination of which is called a *syndrome*, do not depend on the state itself but on which errors have occurred. Hence, measuring the stabilizers, also referred to as syndrome extraction, does not destroy information in the logical state but discretizes the noise and projects the state to a stabilizer eigenstate. The error-free logical qubit state is a superposition of the joint $+1$ stabilizer eigenstates.

Figure 1 depicts a $d = 5$ unrotated surface code logical qubit and its conceptual rotation to its rotated equivalent. $X(Z)$-type stabilizer generators are tensor products of up to four X (Z) operators on neighboring data qubits. Products of generators form the stabilizer group. Z (X) errors anticommute with $X(Z)$-type stabilizers to cause -1 measurement outcomes. *Auxiliary qubits* are used to perform and report the outcomes of stabilizer measurements. We visualize single-qubit errors as linking auxiliary qubits of the stabilizers they anticommute with. Multiple errors form chains that anticommute with stabilizers at their end points, unless an end point is at a boundary. Logical Pauli operations, X_L and Z_L, mutually anticommute but commute with the stabilizer group. X_L (Z_L) is a chain of at least d X (Z) operations joining opposite boundaries of $X(Z)$-type stabilizers.

TABLE I. A distance-d logical qubit's physical qubit count.

	Rotated	Unrotated
Data qubits	d^2	$d^2 + (d-1)^2$
Auxiliaries	$d^2 - 1$	$d^2 + (d-1)^2 - 1$
Total qubits	$2d^2 - 1$	$(2d-1)^2$

B. Errors and decoding

QECCs aim to protect the information stored in their logical qubits even as their physical qubits experience noise. In stabilizer codes, continuous noise acting on data qubits is projected to discrete Pauli noise by measuring the stabilizer generators. The circuits in Fig. 2(a) perform these measurements in the surface code. Including initialization and measurement of the auxiliary, the $X(Z)$-type stabilizer circuit is depth-eight(six).

Different noise models vary in how they model errors in the stabilizer measurement circuits. In a *code capacity* noise

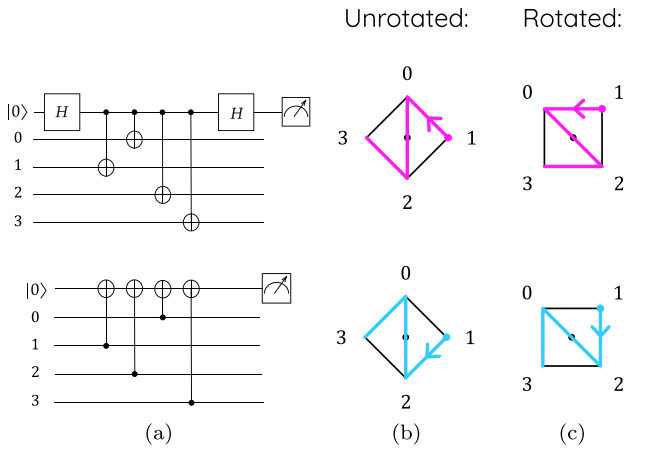


FIG. 2. Representations of X-type (upper diagrams) and Z-type (lower diagrams) stabilizer measurement circuits. The uppermost qubit in each circuit in (a) is the auxiliary. Data qubits are numbered from 0 to 3 depending on their position relative to the auxiliary in the surface code lattice, as depicted in (b) and (c) where the auxiliary is at the center of the stabilizer it measures. We refer to the depicted CNOT order as 10231203, quoting X then Z-type order.

model, it is assumed stabilizer measurement is perfect. Noise on data qubits is modelled as an error, chosen uniformly chosen from $\{\mathbb{1}, X, Y, Z\}$, occurring with probability p. It precedes perfect stabilizer measurements. *Circuit-level* noise, on the other hand, models all the operations as faulty. In addition, it features *idling* errors. These apply to any qubit not being operated on while another qubit is, that is in a particular *time step*. The measurement circuits can consequently cause errors to spread between data and auxiliary qubits, report an incorrect eigenvalue or both. Additionally, if idling errors are inserted in the Z-type circuit at the same time step and p value as the H gates in the X-type circuit, then the circuits become equivalent in terms of applied noise, as if they were both of depth eight.

Table II shows two circuit-level noise models. Standard (or uniform) depolarizing (SD) noise assumes that errors after a gate are uniformly distributed among Pauli operators and assumes that each operation is equally likely to be faulty. The superconducting-inspired (SI) noise model [15,16] also assumes uniform error distributions after each gate but assigns different relative failure probabilities to each operation, reflecting their varying durations and consequent error rates in superconductors.

Performing multiple repetitions, also referred to as rounds or cycles, of stabilizer measurements increases the reliability of the reported measurement outcomes, i.e., the syndrome. In the absence of noise, the syndrome should not change between rounds. When looking at a single stabilizer's measurement outcome, this implies that its parity from one round to the next should be consistent. To formalize this, a *detector* is a parity constraint on a set of measurement outcomes [17]. A detector's set could contain one stabilizer's measurement outcome in the previous and current round. If the parity is not as expected (given a noiseless circuit), then this is a *detection event* and indicates that an error has occurred. Note when calculating parity we refer to the $+1$ (-1) outcome as 0 (1).

Detectors are useful in solving the problem of *decoding*, which is identifying the most likely set of errors given a set of detection events. A decoder must solve this problem "fast" enough to avoid an exponentially growing backlog of output and ensure subsequent logical operations are performed correctly [18,19]. Some decoders prioritize accuracy [20–22], while others prioritize speed [14,23–28]. For a review of decoders used in the surface code, see Ref. [29].

To solve the decoding problem a *detector graph* is first created where detection events are nodes and edges are mechanisms that would trigger them, weighted by their probability of occurring. A *syndrome graph* could be constructed similarly, namely with stabilizer measurements (rather than detection events) as nodes and edges being errors that would cause them to report -1 eigenvalues. Figure 4 shows the X-error syndrome graphs for the rotated and unrotated surface codes. For the $X(Z)$-error syndrome graph the edges are X (Z) errors and nodes are the auxiliary qubits of $Z(X)$-type stabilizers. Syndrome graphs can be used for decoding under a code capacity noise model because only spatial errors occur and each round can be considered independently. However detector graphs, which track changes in the syndrome graph, are usually less dense and aid decoding when also considering temporal errors. This is especially necessary under circuit-level noise where an error in one round may not trigger a change in a stabilizer's eigenvalue until the following [30]. A detector graph is consequently the choice of graph given to the decoder and we subsequently refer to it as the *matching graph*.

In graph theory, a perfect matching is a set of edges where each node is connected to exactly one edge. If the edges are weighted, then a minimum-weight perfect matching (MWPM) is a subset with the minimum weight. When performing MWPM on a detector graph from a surface code, the MWPM is a set of errors that is most probable considering uncorrelated noise. In stabilizer codes, this matching can be equal to the set that actually occurred up to multiplication by stabilizers. This set of topologically equivalent errors is a *class*. Picking the most likely class is *maximum likelihood* decoding and is optimal [31]. However, it takes much longer than MWPM, which simply picks the most likely, or one of the most likely, patterns of errors without considering classes.

One way to test the error-correcting performance of a QECC and decoder is by Monte Carlo sampling of *memory experiments*, as in Refs. [11,15,32]. A memory experiment is also applicable to performing logical operations on logical qubits when using lattice surgery [8], as lattice surgery requires preserving a logical qubit in memory but changing the order of stabilizer measurements along its boundaries. A memory experiment in the surface code in the Z_L (X_L) basis requires encoding the $|0\rangle_L$ ($|+\rangle_L$)) state by initializing all data qubits to $|0\rangle$ ($|+\rangle$) and then performing the first round of stabilizer measurements. After preserving the state through more rounds of stabilizer measurements, usually d [32] or $3d$ [7,15,33], the logical qubit is measured by measuring the data qubits in the Z (X) basis and then checking their parity under Z_L (X_L). The detection events and measurement outcomes are decoded and the necessary corrections decide whether the measurement outcome should be flipped to interpret the correct result. We refer to this experiment as memory Z (X). If X_L or Z_L is formed an odd number of times during a memory experiment, be it from single-qubit errors on data qubits connecting opposite boundaries or the decoder failing and its suggested corrections forming a logical operator, then a logical error has occurred.

As long as the physical error rate (p) affecting the faulty gates and qubits in the QECC is below a certain probability threshold (p_{th}), increasing the distance of the code decreases the logical error rate p_L [34]. Memory experiments over various d and p values reveal a code's threshold for the noise model and decoder. When judging the performance of a QECC it is better to have a higher p_{th} but a lower p_L.

When $p < p_{\text{th}}$, p_L is said to scale with p as:

$$p_L = \alpha_1 \left(\frac{p}{p_{\text{th}_0}}\right)^{d_e} + \alpha_2 \left(\frac{p}{p_{\text{th}_1}}\right)^{d_e+1} + \alpha_3 \left(\frac{p}{p_{\text{th}_2}}\right)^{d_e+2} + \ldots \tag{1}$$

where the *error dimension* is $d_e = d/2$ for even code distances and $d_e = (d+1)/2$ for odd [23,35]. The reasoning behind this is that d_e errors aligned along a logical minimum-weight chain will cause the decoder's corrections to complete the chain, forming a logical error.

The p_L of a memory experiment is usually reported per d rounds of stabilizer measurements as reporting per round

would result in an overestimate of the threshold [32] and d rounds is the number required for fault-tolerant merge and split operations when using lattice surgery for logical quantum operations [8].

Although the $|0\rangle_L$ ($|+\rangle_L$) state is a $+1$ eigenstate of Z_L (X_L) and in the surface code is unaffected by single-qubit Z (X) errors, both types of stabilizers must be measured in a memory experiment to estimate the code's performance for an unknown state and realistically simulate the error-mixing they introduce. For example, the measurement of X-type stabilizers to detect Z errors can copy a single X error to multiple data qubits. While the Z errors would not affect the $|0\rangle_L$ state, the copied X errors would. Furthermore, if a Y error occurs, then it creates correlations in the separate X and Z detector graphs. Uncorrelated decoders [14] treat these as separate X and Z errors and incorrectly assume if a single-qubit error occurs with probability p, then a Y error occurs with probability p^2, because $Y = iXZ$. Correlated decoders hence boost performance [11,15].

C. CNOT order and hook errors

The order of the CNOT gates in the stabilizer measurement circuits must be chosen carefully so the stabilizer circuits can be performed in parallel and do not unnecessarily introduce errors. This can be summarized into three criteria. Any CNOT order which satisfies these criteria we refer to as a *valid* CNOT order. To explain the criteria we will use CNOT labeling depicted in Fig. 2, in which the data qubits are numbered clockwise from 0 as per their position relative to the auxiliary qubit.

(1) The CNOT order must ensure stabilizers mutually commute so give deterministic measurement outcomes in the absence of errors [36]. Practically, shared data qubits between two or more stabilizers must be interacted with in the same relative order by their shared stabilizers. That is, if one stabilizer's interaction precedes another for any shared qubit, then it must do so for all shared qubits.

(2) Unnecessary idling errors can be avoided by ensuring that all CNOTs in a particular time step are physically parallel (aligned along the same axis). Using the labeling of Fig. 2, this implies that while all the X-type stabilizers are interacting with data qubits 0 or 2 (1 or 3), the Z-type stabilizers must also interact with either 0 or 2 (1 or 3), but not necessarily respectively.

(3) In the rotated surface code, CNOT order should avoid *hook errors* [23,37] (see explanation below).

A hook error, or *horizontal hook error* [23], is the copying of a single physical error to two data qubits that align with a logical operator. As shown in Fig. 3(a), the $X(Z)$-type stabilizer measurement circuits can copy X (Z) errors onto data qubits. Copying to four data qubits is equivalent to applying the stabilizer (a logical identity gate) while to three is only a single error up to multiplication by a stabilizer. Copying to two data qubits which align with a logical operator of the same type causes a hook error as the logical operator can now be formed with half as many physical errors as the code distance implies. Figure 3(b) depicts which two CNOTs result in a hook error for each stabilizer type if they are the final two CNOTs in the stabilizer's measurement circuit. Assuming a surface code oriented as per Fig. 1, X hook error occurs if the last two CNOTs for the X-type stabilizers are 12, 21, 03, or 30, whereas a Z hook error occurs if the last two CNOTs for the Z-type stabilizers are 01, 10, 23, or 32, using the numbering system from Fig. 2. Figure 17 shows the effect of hook errors on the relation between p_L and p for the rotated surface code.

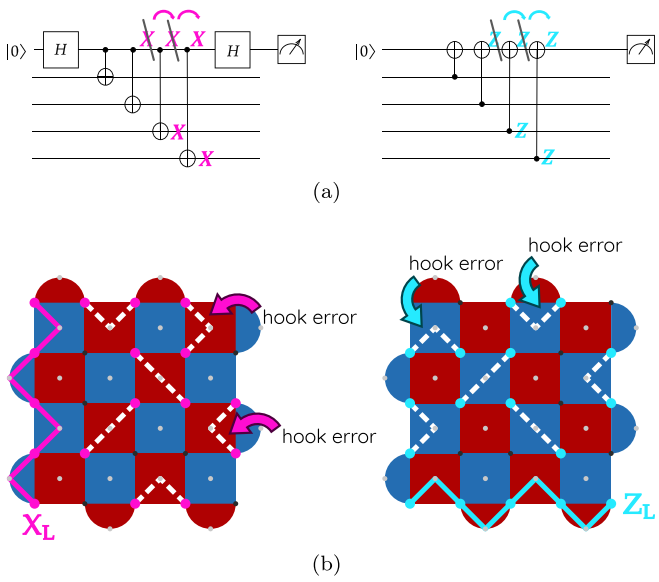


FIG. 3. (a) CNOTs copying an error which precedes them in the stabilizer measurement circuits. Up to multiplication by a stabilizer, copying to four data qubits is the identity, whereas to three is equivalent to a single error. If the error precedes, then the two final CNOTs it is copied to two data qubits. The possible orientations of these two qubits are shown in (b). If these align with a logical operator of the same type as the error, then the logical operator can be formed with half as many single-qubit errors as expected. This is a hook error [23,37].

In the unrotated code, the copied error cannot align with a logical operator so does not have the same effect. We investigated this as an aside and found half the unrotated orders performed marginally better than the other half, but this had no correspondence with hook-error orders in the rotated code (see Fig. 18).

D. Prior work

In this section we review prior work which led to ours, outlining prior threshold values, p_{th}, for the rotated and unrotated surface code as well as previous work comparing the scaling of their logical error rates. All quoted results below used uncorrelated MWPM decoders.

We first consider investigations which implemented code capacity noise models. Criger and Ashraf [13] found the rotated code had a slightly lower threshold than the unrotated. It was suggested that this is because any nonoptimal decoder will be affected by the different number of minimum-weight paths that can cause logical errors in the two codes. These paths are depicted in Fig. 4 for X_L. To form a particular logical operator, a distance-d unrotated code has only d minimum-weight paths because any diversion from a straight path joining opposite boundaries is no longer a minimum-weight path. For the rotated surface code however, all of the edges can

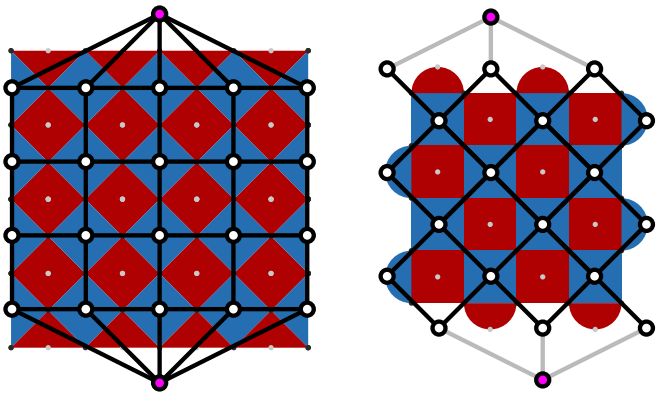


FIG. 4. Graphs for which paths between the two colored nodes correspond to an X_L operator on $d = 5$ unrotated (left) and rotated (right) surface codes. Edges correspond to Pauli X errors, nodes to each Z-type stabilizer. Figure adapted from Ref. [13]. Forming a minimum-weight path in the unrotated surface code cannot be done using horizontal edges meaning only d (5 in this case) minimum-weight paths. In the rotated code all the edges can be used, resulting in a lower bound of $\binom{d}{\lfloor d/2 \rfloor}$ minimum-weight paths (in this case 52) [13].

contribute to a minimum-weight path, implying the number of minimum-weight paths is greater than the unrotated code and lower bounded by $\binom{d}{\lfloor d/2 \rfloor}$ [13].

Beverland *et al.* [12] found that the unrotated code has a higher (worse) p_L than the rotated code when compared at the same number of qubits. This was except for one regime at p very close to threshold. Here the unrotated code achieves a lower p_L than the rotated code, even though at equal qubit numbers it has a lower distance. Like in Ref. [13], the reason suggested for this is that unrotated code has a smaller proportion of possible minimum-weight paths that cause a logical error than the rotated code or, in other words, a higher entropy.

We now consider previous investigations of just the unrotated surface code using SD circuit-level noise. Fowler *et al.* [35] simulated the unrotated code for odd distances. They investigated down to a p_L per round of, at the lowest, $p_L \approx 10^{-5}$ and found logical to physical error scaling $p_L \approx 0.03(p/p_{\text{th}})^{d_e}$, where $d_e = (d-1)/2$. They state that for even distances $d_e = d/2$. Stephens [32] clarified using numerical simulations how estimates of the unrotated code's threshold depend on noise model, the stabilizer measurement circuits, and the decoder. X-type measurement circuits detect Z errors and are of greater depth than the Z-type circuits. Without idling errors, the logical error suppression will hence be worse and p_{th} consequently lower (worse) for memory X as it preserves $|+\rangle_L$, which is sensitve to Z errors. Reporting p_L per d rounds Stephens found $p_{\text{th}} \approx 5 \times 10^{-3}$ ($p_{\text{th}} \approx 5.4 \times 10^{-3}$) for memory X (Z). Paler and Fowler [11] simulated odd code distances from $d = 3$ to $d = 9$ with p_L reported for memory X and showed the unrotated code achieves a lower (better) p_L per round than the rotated surface code for distance 5, 7, and 9, as simulated for p values corresponding to a p_L per round of approximately 10^{-7}. Distance 3 was an exception, but due to being a very low-distance code it suffers *edge effects* in space. That is, the proportion of low-weight stabilizers (weight-2 in the rotated code, weight-3 in the unrotated) on the boundaries of a surface code lattice as compared to the weight-4 stabilizers in the bulk of the lattice is higher for lower-distance codes which have a consequent reduction in performance.

Finally, we consider the investigation of the rotated surface code using SD and SI noise. Gidney *et al.* [15] compared the rotated surface code to another QECC [38] to estimate the number of qubits required by each to reach the *teraquop* regime, which is a p_L per d rounds of one in a trillion. This implies that a trillion logical operations (each requiring d stabilizer measurement rounds) can be performed before, on average, one logical error occurs. Plotting p_L per d rounds they showed the rotated code has $p_{\text{th}} \approx 0.005$, for both noise models, with a slightly lower (worse) p_{th} for SI noise than SD noise.

Prior work shows the unrotated code has a slightly higher threshold than the rotated code and achieves a lower p_L at equal distances, as simulated for relatively high logical error rates. We next present our methods investigating higher code distances, both odd and even, achieving lower p_L under SD and SI circuit-level noise.

III. METHODS

In this work we simulated memory experiments in the rotated and unrotated surface code using both odd and even distances, low physical error rates, and circuit-level noise. Our aim was to compare the scaling of logical to physical error rates in the rotated and unrotated code, quantifying the latter's advantage in terms of the number of qubits used to achieve the same p_L values and provide evidence that this persists for high d and low p.

In this comparative study we implemented SD and SI [15] circuit-level noise, as applied to circuits compiled with CNOT gates, and used the MWPM decoder PyMatching 2 [14]. Details of these noise models are in Table II.

We performed Monte Carlo sampling of trillions of runs of memory experiments in both codes under both memory types and noise models. This took approximately 12 CPU core years. Our generated data, as well as the Python code used to generate the circuits, run simulations, and render plots, is available at our GitHub repository [39]. We used Stim [40], a tool for simulation and analysis of stabilizer circuits; PyMatching 2 [14], a fast MWPM decoder; and Sinter [41], which uses Python multiprocessing for bulk sampling and decoding of Stim circuits.

We generated a Stim circuit file [42] to simulate a memory experiment stabilizer circuit for each code, physical error rate, noise model, memory type, and CNOT order. It contains annotations to assert that the parity of certain measurement sets, i.e., detectors, are deterministic in the absence of noise and to assert which final measurements are combined to calculate the logical observable's measurement outcome. The detection events are converted to a detector graph for PyMatching to decode.

To generate the Stim circuits, we modified a Python translation [43] of Stim's in-built circuit generator. To reduce simulation time we enabled the option to exclude detectors in the opposite basis to that of the measured logical observable

TABLE II. Standard depolarizing and superconducting-inspired [15] noise, as applied to the gates in our circuits.

Qubit operation	Error	SD	SI
CNOT	Perform CNOT then choose an error uniformly from $\{\mathbb{1}, X, Y, Z\}^{\otimes 2} \setminus \{\mathbb{1}\mathbb{1}\}$	p	p
Hadamard (H)	Perform H then choose an error uniformly from $\{X, Y, Z\}$	p	$p/10$
Reset to $\|0\rangle$ ($\|+\rangle$)	Instead resets to $\|1\rangle$ ($\|-\rangle$)	p	$2p$
Measurement	Report incorrect result and project to orthogonal eigenstate	p	$5p$
Idle during gates on other qubits	Choose an error uniformly from $\{X, Y, Z\}$	p	$p/10$
Idle during reset or measurement of other qubits	Choose an error uniformly from $\{X, Y, Z\}$	p	$2p$

because PyMatching, being an uncorrelated decoder, does not use these detection events. We added modifications that enabled the reordering of CNOT gates, suggested as the explanation for the difference in the unrotated surface code's p_L between memory X and Z [44], and added idling errors on any qubits not being operated on while operations were being applied to other qubits. This included adding idling errors on Z-type stabilizer auxiliaries at the same error rate and time step as the H gates in the X-type stabilizer circuit. From a noise perspective this renders both the circuits equivalent to the depth-eight circuits in Ref. [32]. Though they are not technically both of depth eight, the noise model renders them effectively of equal depth. For an example of a circuit displaying our modifications see Fig. 10.

To minimize uncertainty in the results we incrementally scaled the maximum number of samples taken of each memory experiment while correspondingly reducing the maximum number of logical errors seen before sampling would stop. We initiated our simulations with a ceiling of one million shots and one hundred thousand errors, incrementally increasing the shot count to a trillion and tapering down the maximum errors to 20. This was performed on two 64-core computers.

We calculated p_L per d rounds but ran $3d$ rounds of stabilizer measurements to reduce time-boundary edge effects, which arise because the first and last round have less logical errors [33]. The p_L per d rounds is hence the XOR of three independent Bernoulli distributions.

IV. RESULTS AND DISCUSSION

A. Affect of CNOT order

Figure 5 (11) displays p_L per d rounds versus p results implementing SD (SI) noise. Before discussing our main result, in this section we address the noticeable separation between the unrotated code's memory X and Z performance. This occurred despite idling errors meaning both stabilizer measurement circuits (Fig. 2) behave as though they are of depth 8 (see Sec. III).

First considering the rotated code, it had indistinguishable p_L vs p for memory X and Z up to uncertainty, as depicted by the overlaid "X" and pentagonal markers in Figs. 5 and 11. This is because, from the perspective of the noise model, the stabilizer measurement circuits are of equivalent depth. This

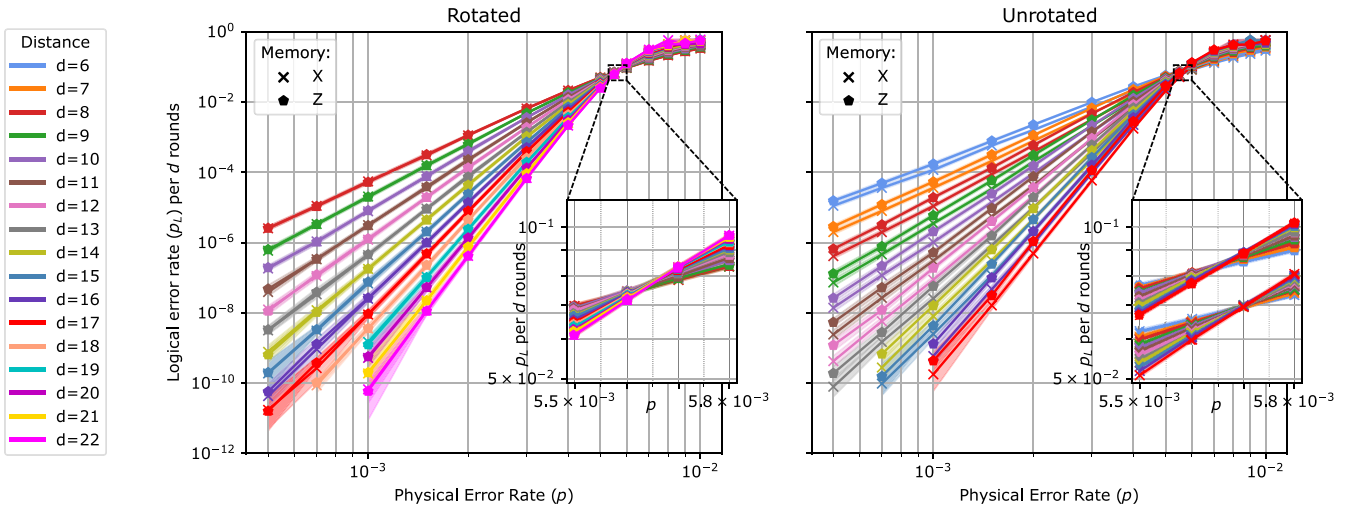


FIG. 5. Logical error rate per d rounds (cycles) of stabilizer measurements vs p, the rate of SD circuit-level noise (see Fig. 11 for SI noise). Memory X (Z) preserves the $|+\rangle_L$ ($|0\rangle_L$) state. The threshold (p_{th}) is the p value where the curves intersect. Insets show a region close to p_{th} with additional data points. The split between memory types in the unrotated code is discussed in Sec. IV A. For subsequent comparisons we take its worst case. Displayed results are from CNOT order 10231203, numbering as per Fig. 2, but for all valid CNOT orders these results generalize (see Sec. IV A). With k logical errors observed, highlighted regions show p_L values for which the conditional probabilities $P(p_L|k)$ are within a factor of 1000 of the maximum likelihood estimate (MLE) $p_L = k/n$, assuming a binomial distribution and converted to per d rounds. We ran $3d$ rounds, with the per d rounds being the XOR of three independent Bernoulli distributions.

reproduced for all valid CNOT orders (see Fig. 17), which avoided hook errors. (Results with hook errors also in Fig. 17.)

Contrastingly, the unrotated code had differing memory X and Z performance depending on its CNOT order. To be a valid order in the unrotated code the first and last of the four CNOTs align with one minimum-weight logical operator, while the second and third ("inner") CNOTs align with the other. Labelling the data qubits as per Fig. 2, if the inner CNOTs act on qubits 1 and 3 (0 and 2), then they align with Z_L (X_L) and we see a higher p_L for memory X (Z). This was observed for all valid unrotated orders, with an additional, seemingly unrelated marginal increase in performance observed for half the orders. Results showing this are in Fig. 19. This affect was not artificially introduced from the orientation of the matching graph given to the decoder, as shown in Fig. 20 which swapped X and Z-type stabilizers.

This alignment of the inner CNOTs correlates with a split in performance but is not causative. If the inner CNOTs increased the occurrence of the logical operator they align with, then increasing total circuit depth to enable interspersing of the alignment of all four gates should symmetrize the unrotated code's performance. This did not occur, as shown in Fig. 21. Further investigation was outside the scope of this paper, especially as the unrotated code did not outperform the rotated even when leveraging this effect (see Sec. IV C).

Usually the worst-case code sets the overall threshold [32], and unless preserving a known Pauli eigenstate, which has limited utility, the CNOT order cannot be chosen to achieve the best-case performance of the unrotated code. Subsequent comparisons hence use an ordering which shows the aforementioned marginal increase in performance but is a worst-case unrotated code for the memory type, featuring a CNOT ordering unfavorable to the preserved state. We will refer to this simply as the unrotated code. For specificity we are reporting p_L results for memory Z and the CNOT order is 10231203 for both codes (creating X_L-aligned inner CNOTs for the unrotated, implying a poorer memory Z performance). However, these results reproduce for memory X when also using a worst-case unrotated order (see Figs. 17 and 18), so any valid CNOT order can be chosen.

B. Thresholds and p_L to p scaling

In this section we present calculated threshold values and p_L to p scaling. Figure 5 (11) depicts plots of our simulated data under SD (SI) noise. We used $d \geqslant 6$ (8) for the unrotated (rotated) code, corresponding to a qubit count of $n \geqslant 121$ (127) for our threshold and scaling calculations. We excluded lower d values which do not follow the same scaling as higher d (see Fig. 22) due to edge effects. Hereafter we refer to the worst-case (see Sec. IV A) unrotated code (memory Z for the depicted CNOT order) simply as the unrotated code.

We first present values for p_{th}, calculated by fitting the near-threshold p values in the insets of Figs. 5 and 11 to the near-threshold scaling ansatz of Refs. [32] and [45]:

$$p_L = A + B(p - p_{\text{th}})d^{1/\nu} + C(p - p_{\text{th}})^2 d^{2/\nu}. \tag{2}$$

Table III presents our results, which are comparable to $p_{\text{th}} \approx 5 \times 10^{-3}$ and $\nu = 1.05 \pm 0.01$ in Ref. [32] for the unrotated code. It also used SD noise and depth-eight stabilizer circuits but a different MWPM decoder.

TABLE III. Fits to Eq. (2) for p_{th} and scaling exponent (ν).

Noise	Code	p_{th}	ν
SD	Rotated	$(5.637 \pm 0.004) \times 10^{-3}$	1.09 ± 0.03
	Unrotated	$(5.652 \pm 0.003) \times 10^{-3}$	1.05 ± 0.03
SI	Rotated	$(5.092 \pm 0.007) \times 10^{-3}$	1.09 ± 0.06
	Unrotated	$(5.077 \pm 0.008) \times 10^{-3}$	1.06 ± 0.07

More applicable to useful quantum computing is p well below threshold, which we now consider. Our results show that for a given d, the unrotated code achieves a lower p_L than the rotated code. This was previously shown [11] using odd d from 5 to 9 inclusive down to a p_L per round of 10^{-7}. We confirm this pattern continues for higher odd d down to a p_L per round of $\approx 10^{-12}$ (a p_L per d rounds $\approx 10^{-11}$) and we add results for even d.

We now present our results for p_L to p scaling. Previous results [35] fit to Eq. (1) with $d_e = (d+1)/2$ ($d_e = d/2$) for odd (even) distances, but this was at low d and consequently higher p_L values. Even when including the higher-order terms in Eq. (1), this d_e failed to fit our data. We hence fit to

$$p_L = \alpha(p/\beta)^{\gamma d - \delta}, \tag{3}$$

where the error dimension can be defined as $d_e = \gamma d - \delta$.

Our fits used $p \leqslant 0.004$ because p values closer to p_{th} were not indicative of subthreshold scaling. For this reason and to avoid confusion we label the term in the denominator β rather than p_{th} as it reflects the region of intersection of line fits based on subthreshold scaling rather than on simulated data close to p_{th}.

The performance for odd-distance codes scales better than even-distance codes at low distances (see Fig. 22); however, when fitting to high distances, their d_e was identical up to uncertainty (see Table IV). We hence combined odd and even d in the fits. We found for SD noise:

$$p_{L,\text{ro}} = 0.08(p/0.0053)^{0.58d-0.28}, \tag{4}$$

$$p_{L,\text{unro}} = 0.08(p/0.0054)^{0.71d-0.70}, \tag{5}$$

and for SI [15] noise:

$$p_{L,\text{ro}} = 0.05(p/0.0049)^{0.63d-0.67}, \tag{6}$$

$$p_{L,\text{unro}} = 0.05(p/0.0049)^{0.75d-0.89}. \tag{7}$$

Plots displaying these fits superimposed on sampled data can be found in Fig. 16, while the parameters and their uncertainties for odd, even, and combined (odd and even) d are in Table IV. While previous results and theory find $d_e \approx 0.5d$, we instead saw d_e values between $0.6d$ and $0.75d$.

C. Logical error rate vs total qubit count

We now present our main result, which, in short, is that the rotated surface code requires ≈75% the total number of qubits used by the unrotated code to achieve the same p_L. We next detail how we arrived at this value and note that the exact ratio depends on p and p_L.

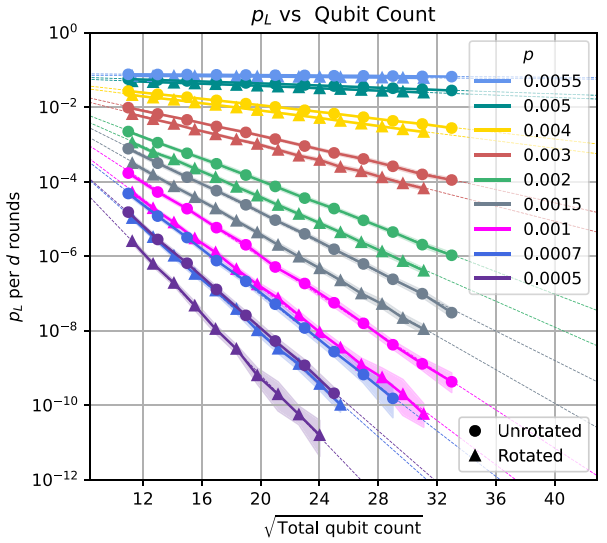


FIG. 6. Reformulation of data in Fig. 5 to show p_L vs qubit count under SD noise (see Fig. 12 for SI noise). We fit to $d \geqslant 8$ (6) for the rotated (unrotated) code to minimize edge effects. The points of intersection between these least-squares line fits and $p_L = 10^{-12}$ are used to generate the "teraquop" plot in Fig. 8. Highlighted regions show p_L values for which the conditional probabilities $P(p_L|k)$ are within a factor of 1000 of the MLE $p_L = k/n$, assuming a binomial distribution. These results report memory Z but reproduce for memory X (see Sec. IV A).

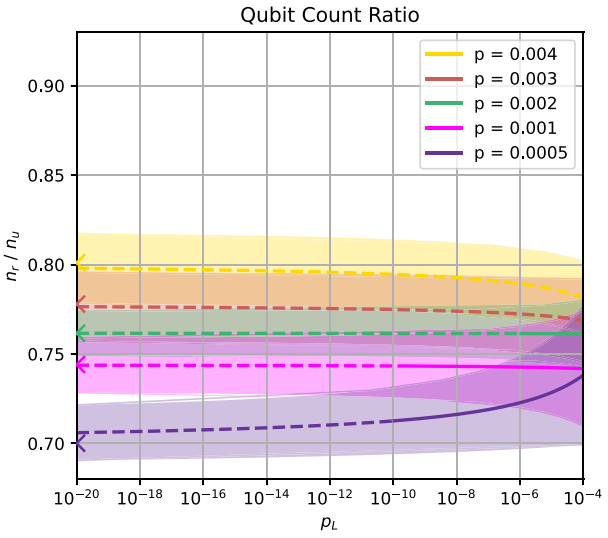


FIG. 7. The ratio of the total number of qubits used by the rotated and unrotated surface code to achieve the same p_L per d rounds for a selection of p values under SD noise (see Fig. 13 for SI noise). Qubit counts are calculated from the line fits of Fig. 6, with projected qubit counts indicated by dashed lines. Colored arrowheads on the y axis indicate the limit as $p_L \to 0$. Highlighted regions indicate uncertainty propagated from the standard error (SE) of the line-fit parameters of Fig. 6. Simulations ran CNOT order 10231203 and memory Z but reproduce for other orders and memory (see Sec. IV A).

Figure 6 (Fig. 12) plots p_L vs n for the rotated and unrotated surface codes under SD (SI) noise. This plot shows that the rotated code achieves a lower p_L at the same n.

The ratio of the number of qubits used by each code to achieve the same p_L is displayed in Fig. 7 (Fig. 13) for SD (SI) noise, with ratios calculated using the line fits from Fig. 6 (Fig. 12). The ratio limit as $p_L \to 0$ (and hence $n \to \infty$) is indicated with an arrowhead for each p. These projections indicate that there is never a point for which the unrotated code's qubit count becomes less than the rotated code for equal p_L. At the operational $p = 10^{-3}$ the rotated code uses about 74% the number of qubits used by the unrotated code to achieve the same p_L. Exact qubit numbers for the choice $p_L = 10^{-12}$ are presented in the next section. While we do not foresee a mechanism by which this ratio would drastically change by using a different decoder from Ref. [29], we note that our results are for our choice of decoder [14] and SD/SI circuit-level noise models.

D. Teraquops

A p_L per d rounds of $p_L = 10^{-12}$ is known as the teraquop regime, as it implies a trillion logical operations (each requiring d rounds of stabilizer measurements) can be performed before, on average, one logical error occurs [15]. Projecting the least-squares line fits from Fig. 6 (Fig. 12) to $p_L = 10^{-12}$ reveals the teraquop qubit counts for SD (SI) noise, as displayed in Fig. 8 (Fig. 14). Under SD noise and at $p = 10^{-3}$, the rotated code uses 1398 ± 19 qubits while the unrotated code uses 1880 ± 33 qubits. That is, the rotated code requires $74.4 \pm 1.7\%$ the number of qubits as the unrotated. Under SI noise the counts are very similar at 1350 ± 16 and 1864 ± 20, respectively, for a ratio of $72.4 \pm 1.2\%$.

These qubit counts do not correspond to an exact d. The inset in Figs. 8 and 14 display teraquop qubit counts rounded up to the next d. Using these, the teraquop regime at $p = 10^{-3}$ under either noise model requires 1457 qubits in the rotated code ($d = 27$) or 2025 ($d = 23$) in the unrotated code: a ratio of 72%.

Very close to p_{th} the codes approach equal qubit counts. This is corresponds to the regime found by Beverland *et al.* [12] but under a code capacity noise model where the unrotated code used less qubits than the rotated at the same p_L when very close to p_{th}. We found that under circuit-level noise this only occurs for the best-case unrotated code and at p extremely close to p_{th} (see Figs. 23–26). However, using a best-case unrotated code requires knowledge of the state being preserved and, furthermore, this region is too close to p_{th} to be practical as its teraquop qubit count is over 10^6.

E. Memory times

We can compare the length of time a quantum memory employing either the rotated or unrotated surface code would last before the probability of a logical error is equal to that of a physical error. This is calculated using the number of rounds n_r that can be performed before p_L per n_r equals to p. We assume each round of stabilizer measurement takes 1 μs, the characteristic time in a superconducting quantum computer [46].

Figure 9 (Fig. 15) displays the results for a selection of p values under SD (SI) noise. The rotated code achieves a

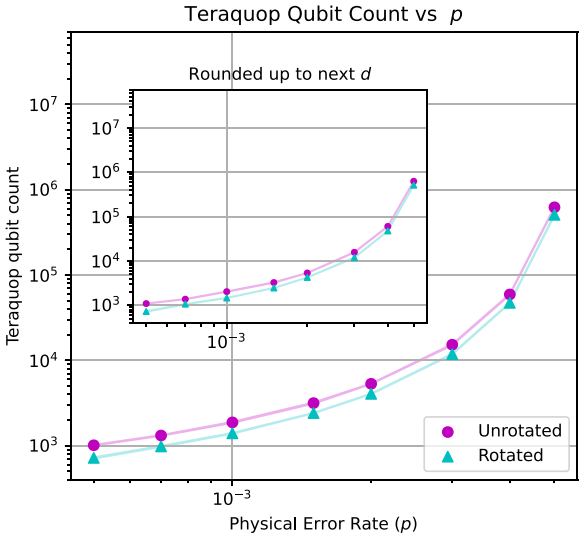


FIG. 8. Number of qubits needed to reach the teraquop regime ($p_L = 10^{-12}$) under SD noise (see Fig. 14 for SI noise), calculated using the weighted line fits from Fig. 6, with weights based on the root-mean-square error (RMSE) of the MLEs for p_L, assuming a binomial distribution. Line thickness indicates uncertainty, propagated from the SEs of the line-fit parameters using these weights. Inset shows n rounded up to the next code distance. Memory Z results are displayed but are equivalent to memory X (see Sec. II C).

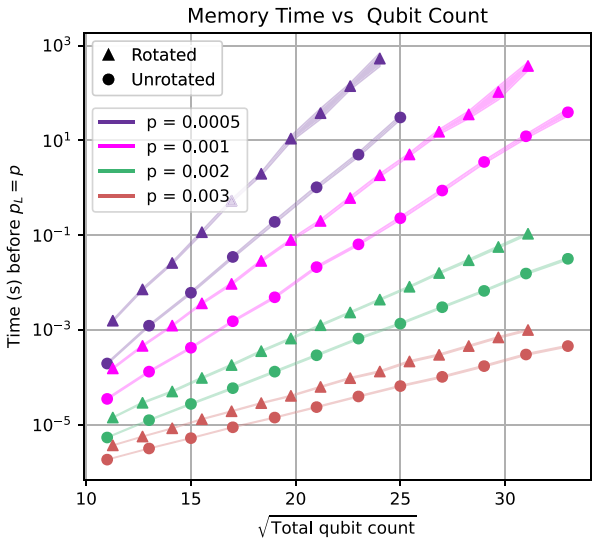


FIG. 9. The achievable memory time versus total qubit count in the rotated and unrotated surface code for a selection of p values under SD noise (see Fig. 15 for SI noise). This is the length of time before a logical error is equally as likely as a physical error and assuming one round takes 1 μs. This figure presents memory Z results, which reproduce up to uncertainty for memory X (see Sec. IV A). Line thickness indicates uncertainty, calculated as the propagated RMSE of the MLEs for p_L, assuming a binomial distribution.

higher memory time than the unrotated code for the same qubit count and is hence more suitable for preserving quantum states such as when performing lattice surgery [8] or using quantum memories to perform quantum communication with a quantum "sneakernet" [47].

V. CONCLUSION AND FURTHER WORK

Our work quantified the low-p_L scaling of p_L to p for the rotated and unrotated surface codes under circuit-level noise using a MWPM decoder [14] for both odd and even distances. Under standard depolarizing (superconducting-inspired) noise, we found the rotated code scales with $0.58d$ ($0.63d$) and the unrotated code with $0.71d$ ($0.75d$), as per Eqs. (4)–(7). We showed that the rotated code uses ≈74% the number of qubits used by the unrotated code to achieve $p_L = 10^{-12}$ at $p = 10^{-3}$, as per Figs. 7 and 13. For all other $p < p_{\text{th}}$, the ratio remains ≈75%, with the rotated code continuing to outperform the unrotated code as $p_L \to 0$. We tested the assumption that because the rotated code uses less qubits to achieve the same distance as the unrotated, it should also use less qubits to achieve the same p_L. We confirmed this, confirmed that it continues to do so for high distances and low p_L. We also quantified the saving, with the exact ratio depending on the exact p and p_L. Our findings justify using the rotated code for future applications of the surface code.

Finding a more precise qubit-saving ratio for particular hardware such as superconductors would require more investigation, such as with circuits compiled with iSWAP gates. These are native to superconductors and compiling circuits with them has been shown to reduce hardware overheads [30]. A further hardware overhead reduction could also be achieved in different realizations of quantum computers by reducing stabilizer measurement circuits to a single step rather than having them comprise numerous gates [48].

The mechanism which led to a best-case and worst-case unrotated code could be useful to explore if it also affects the rotated code in some way not revealed by our investigation. For the unrotated code, we did not see enough of an advantage gained from exploiting this effect to warrant its use over the rotated, especially when considering other noise models such as biased noise. Biased noise is a notable case in which the rotated code outperforms the unrotated, even at the same distance, as shown by Tuckett *et al.* using the code capacity [49] and phenomenological [50] noise models.

While our work suggests a binary choice between the rotated or unrotated code, future work could explore surface codes with qubit numbers somewhere between these choices. While adding more qubits to a rotated code's lattice until it forms an unrotated code reduces the resulting p_L even without increasing the distance, it could follow that adding more qubits, even without adding enough to form the equivalent unrotated code, would also reduce p_L. The inverse has already been shown, namely that using less qubits slightly increases p_L, specifically when reusing auxiliaries to measure boundary stabilizers [37]. We could expect then that adding slightly more qubits would slightly decrease p_L, and future work could look at quantifying the advantage of doing so. This would be relevant when considering that the number of qubits on a fabricated quantum device would usually not exactly correspond to a precise d for the surface code. The caveat to this is that if the noise is highly biased, then adding more qubits until the rotated code more closely resembles the unrotated could decrease performance, because under biased noise an

unrotated code performs worse than a rotated for the same distance [49].

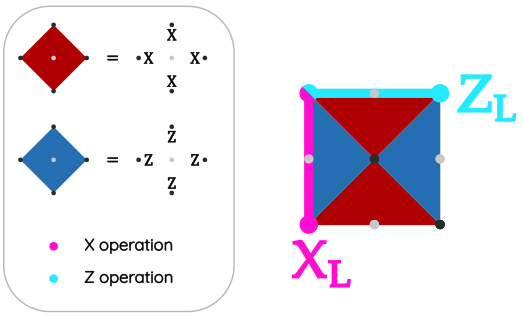
The effect of different decoders on the rotated and unrotated surface code could also be investigated. While there does not seem to be a strong cause to believe that a different decoder from Ref. [29] would ever render the rotated code less efficient than the unrotated code in terms of qubit count, we stress that our results are true for our chosen noise models and MWPM decoder [14]. For instance, while Ref. [11] showed that a correlated decoder reduces p_L for both surface codes when compared to an uncorrelated one, well-below-threshold scaling or the relative reduction for both codes was not quantified. In our work we did not implement a correlated decoder. Future work could look at the affect of a correlated decoder, as well as other types of decoders from Ref. [29] such as maximum likelihood [23] or neural network decoders.

Our work was a comparative study to provided numerical justification for the use of the rotated rather than the unrotated surface code by investigating their low-p_L scaling and and the assumption that the rotated code is advantageous for very low p_L at high odd and even code distances. We ran numerical simulations to test its qubit saving when compared to the unrotated code not in achieving the same distance, but in achieving the same logical error rate, and showed using projections that this saving continues for arbitrarily high distances and consequently low p_L. We did so using circuit-level SD and SI [15] noise and the MWPM decoder PyMatching 2 [14] and can conclude that under these conditions and at an operational physical error rate of $p = 10^{-3}$ the rotated surface code uses $\approx$74% the number of qubits used by the unrotated code.

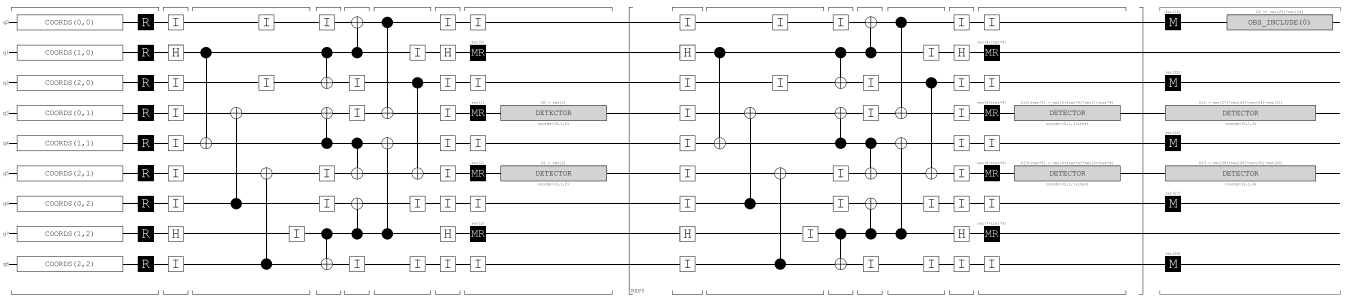
ACKNOWLEDGMENTS

We thank Craig Gidney for valuable discussions and suggesting CNOT order as the cause of the asymmetry in the unrotated code's performance. We thank Alan Robertson, Ben Criger, and Michael Newman for valuable discussions. This research received funding from the Defense Advanced Research Projects Agency Quantum Benchmarking program under Awards No. HR00112230007 and No. HR001121S0026 contracts. We acknowledge the traditional owners of the land on which this work was undertaken at the University of Technology Sydney: the Gadigal people of the Eora Nation.

APPENDIX: SUPPLEMENTARY FIGURES



(a)



(b)

FIG. 10. (a) Two possible logical operators and the stabilizer arrangement for a distance 2, unrotated surface code. (b) The "timeline-svg" generated by Stim [40] of our circuit file realizing a memory Z experiment on the distance-2 unrotated surface code. It uses 21302130 CNOT order. All gates are faulty as per Sec. III. Time steps are indicated with horizontal square brackets. The repeated section is indicated with vertical square brackets. R implies reset to $|0\rangle$, I is the identity gate (an idling error), and CNOT gates are marked as usual from control to target. MR is a sequential combination of a measure and reset gate in the Z basis. Both the measurement and the reset can be faulty. Detectors create the matching graph from stabilizer measurement outcomes to be given to PyMatching [14], and the included observable is a calculation of the state of the measured logical observable based on the measurement outcomes of the data qubits.

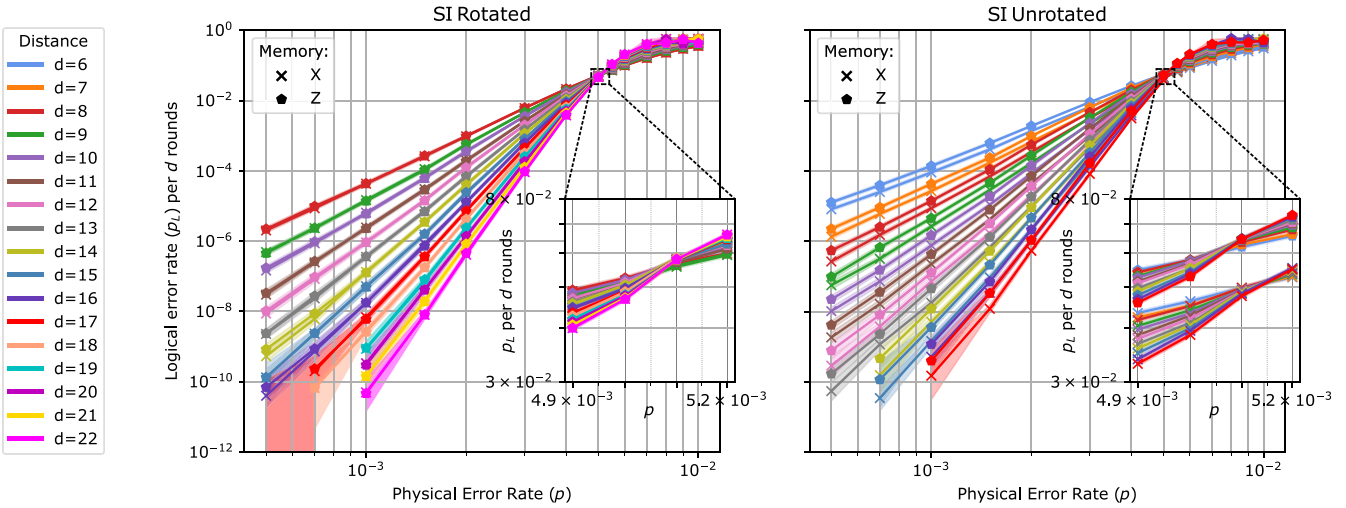


FIG. 11. SI noise: Logical error rate per d rounds (cycles) of stabilizer measurements vs physical error rate under SI noise. Memory X (Z) preserves the $|+\rangle_L$ ($|0\rangle_L$) state. The threshold (p_{th}) is the p value where the curves intersect. Insets show a zoomed-in region close to p_{th} with additional data points. The split between memory types in the unrotated code is discussed in Sec. IV A. Displayed results are from CNOT order 10231203, numbering as per Fig. 2, but for all valid CNOT orders these results generalize (see Sec. IV A). With k logical errors observed, highlighted regions show p_L values for which the conditional probabilities $P(p_L|k)$ are within a factor of 1000 of the MLE $p_L = k/n$, assuming a binomial distribution and converted to per d rounds. We ran $3d$ rounds, with the per d rounds being the XOR of three independent Bernoulli distributions.

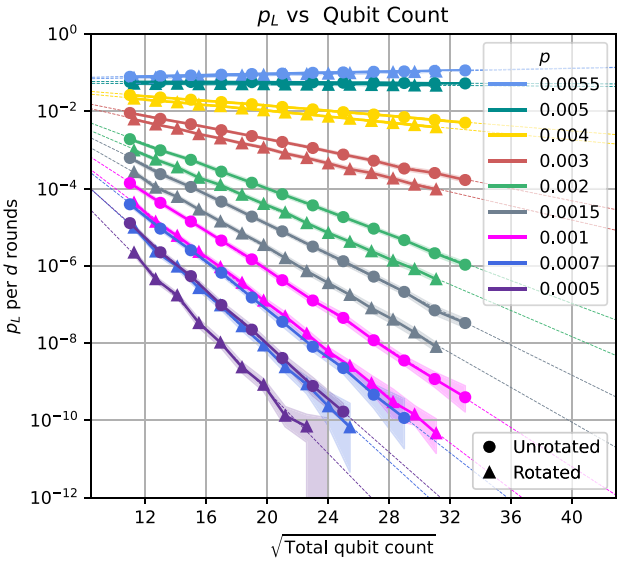


FIG. 12. SI noise: Reformulation of the data if Fig. 11 to show p_L as a function of qubit count under SI noise (see Fig. 6 for SD noise). We fit to $d \geqslant 8$ for the rotated and $d \geqslant 6$ for the unrotated code to minimize edge effects. The points of intersection between these least-squares line fits and $p_L = 10^{-12}$ are used to generate the "teraquop" plot in Fig. 14. Highlighted regions show p_L values for which the conditional probabilities $P(p_L|k)$ are within a factor of 1000 of the MLE $p_L = k/n$, assuming a binomial distribution. These results report memory Z but reproduce up to uncertainty for memory X (see Sec. IV A).

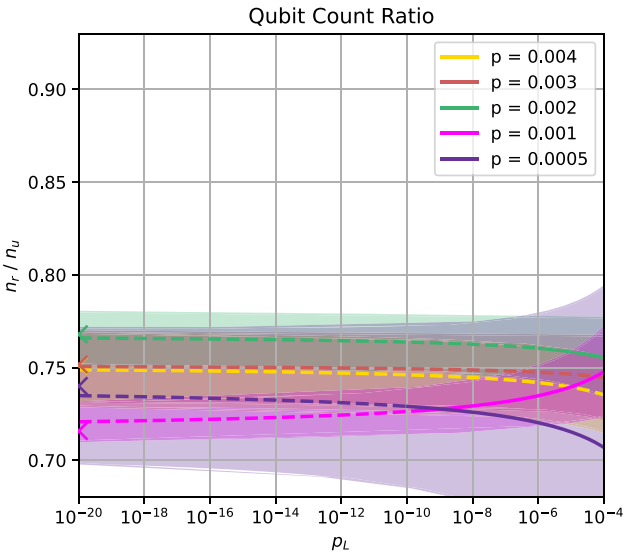


FIG. 13. SI noise: The ratio of the number of qubits used by the rotated versus the unrotated surface code to achieve the same logical error rate (p_L) per d rounds for a selection of p values under SI noise (see Fig. 7 for SD noise). Qubit counts are calculated from the line fits of Fig. 12, with projected qubit counts indicated by dashed lines. Colored arrowheads on the y axis indicate the limit as $p_L \to 0$. Simulations implemented an uncorrelated MWPM decoder [14]. Highlighted regions indicate uncertainty propagated from the standard error of the line-fit parameters of Fig. 12.

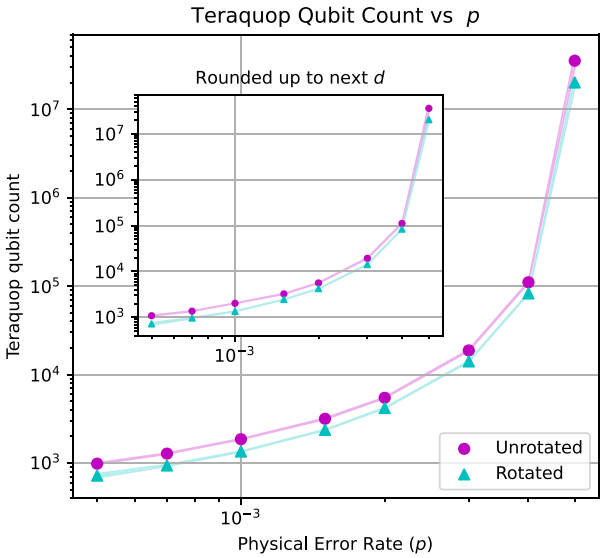


FIG. 14. SI noise: Number of qubits needed to reach the teraquop regime ($p_L = 10^{-12}$) under SI noise (see Fig. 8 for SD noise), calculated using the weighted least-squares line fits from Fig. 12, with weights based on the root-mean-square error of the maximum likelihood estimates for p_L, assuming a binomial distribution. Line thickness indicates uncertainty, propagated from the standard errors of the line-fit parameters using these weights. Inset shows n rounded up to the next code distance. Memory Z results are displayed but are equivalent to memory X (see Sec. II C).

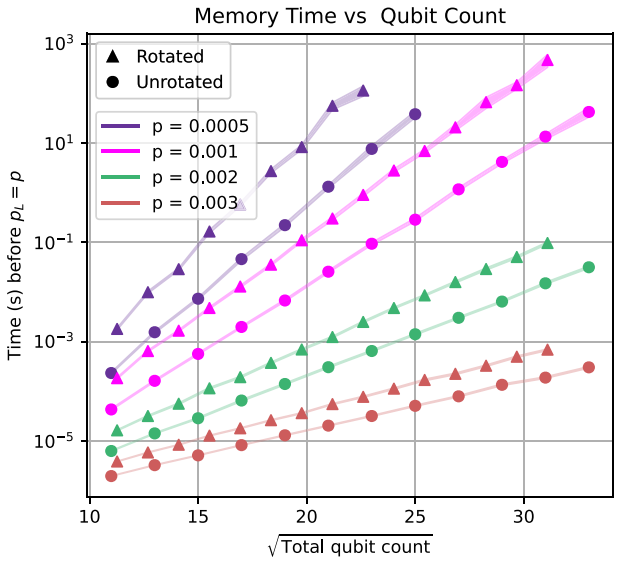
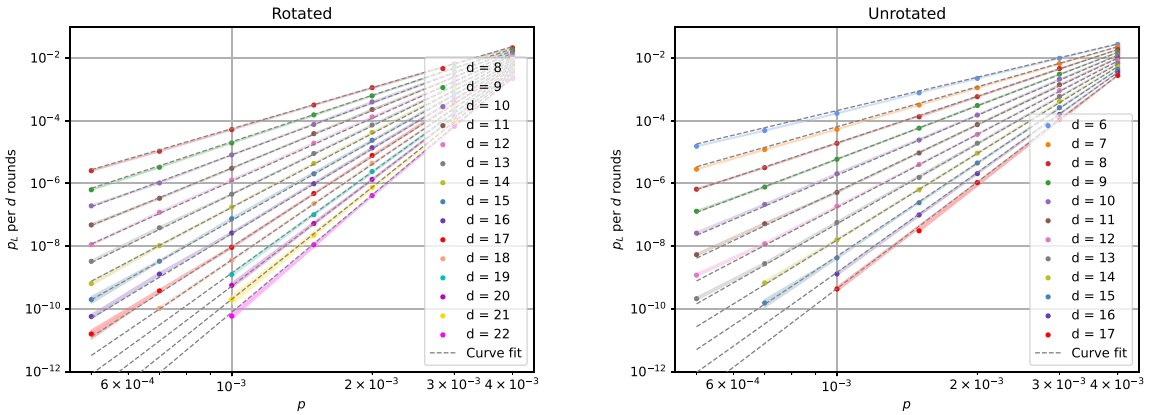
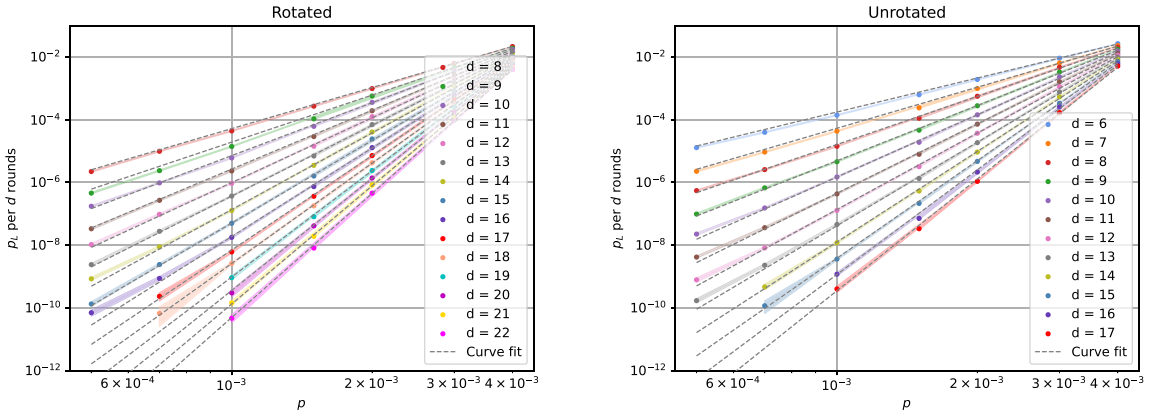


FIG. 15. SI noise: The achievable memory times versus qubit count (n) in the rotated and unrotated surface code for a selection of p values under SI noise (see Fig. 9 for SD noise). This is the length of time before a logical error is equally as likely as a physical error. Calculations of memory time assume one round of stabilizer measurements takes 1 μs. This figure presents memory Z results, which reproduce up to uncertainty for memory X (see Sec. IV A). Line thickness indicates uncertainty, calculated as the propagated RMSE of the MLEs for p_L, assuming a binomial distribution.



(a) SD noise



(b) SI noise

FIG. 16. Function fit plots showing the fitted function $p_L = \alpha(p/\beta)^{\gamma d-\delta}$ as gray dashed lines with parameters for combined odd and even distances as per Table IV. These plots display the same data as Fig. 5 for SD noise and Fig. 11 for SI noise but with the function fits overlaid for physical error rates $p < 0.004$ and with the highlighted uncertainty regions instead being the RMSE of the maximum likelihood estimate for p_L. As in Figs. 5 and 11, displayed results are for memory Z and worst-case unrotated surface code but these are identical up to uncertainty for memory X (see Sec. IV A).

TABLE IV. Fits to $p_L = \alpha(p/\beta)^{\gamma d-\delta}$ [Eq. (3)] using $d \geqslant 6$ (8) for the unrotated (rotated) code and physical error rates $p \leqslant 0.004$, based on memory Z results but these are equivalent up to uncertainty to memory X (see Sec. II C). The parameters for the combined (both odd and even) fits are quoted in the main text (Sec. IV B) because the scaling with d for odd and even distances overlap up to uncertainty. Plots with these fits overlaid on the simulated data are shown in Fig. 16. Line fits were performed on each distance's p_L versus p curve for $p \leqslant 0.004$ and β is the p value of the closest point to all the lines, with its uncertainty being the maximum of the perpendicular distance to each line. The standard errors in the line-fit parameters for each distance were used to perform a weighted average to calculate α and its uncertainty. The best γ and δ were calculated using each distance's line-fit gradient (on a log-log plot), m, and fitting to $m = \gamma d + \delta$. The standard error in these parameters is the square root of the sum of the residual variance and the mean of the variance of each m.

Fits to $p_L = \alpha(p/\beta)^{\gamma d-\delta}$						
Noise model	Code	Distances	α	β	γ	δ
SD	Rotated	Odd	0.079 ± 0.011	0.00530 ± 0.00001	0.577 ± 0.009	0.27 ± 0.14
		Even	0.077 ± 0.010	0.00529 ± 0.00003	0.579 ± 0.008	0.28 ± 0.13
		Combined	0.078 ± 0.008	0.00529 ± 0.00003	0.578 ± 0.006	0.28 ± 0.09
	Unrotated	Odd	0.082 ± 0.011	0.00540 ± 0.00001	0.707 ± 0.016	0.72 ± 0.20
		Even	0.079 ± 0.009	0.00538 ± 0.00002	0.704 ± 0.013	0.69 ± 0.15
		Combined	0.081 ± 0.007	0.00539 ± 0.00002	0.706 ± 0.010	0.70 ± 0.11
SI	Rotated	Odd	0.050 ± 0.009	0.00484 ± 0.00001	0.619 ± 0.011	0.51 ± 0.17
		Even	0.050 ± 0.009	0.00485 ± 0.00004	0.632 ± 0.015	0.77 ± 0.24
		Combined	0.050 ± 0.006	0.00485 ± 0.00004	0.627 ± 0.010	0.67 ± 0.15
	Unrotated	Odd	0.056 ± 0.015	0.00495 ± 0.00002	0.743 ± 0.021	0.86 ± 0.26
		Even	0.053 ± 0.013	0.00489 ± 0.00001	0.756 ± 0.019	0.94 ± 0.22
		Combined	0.054 ± 0.010	0.00492 ± 0.00003	0.748 ± 0.014	0.89 ± 0.16

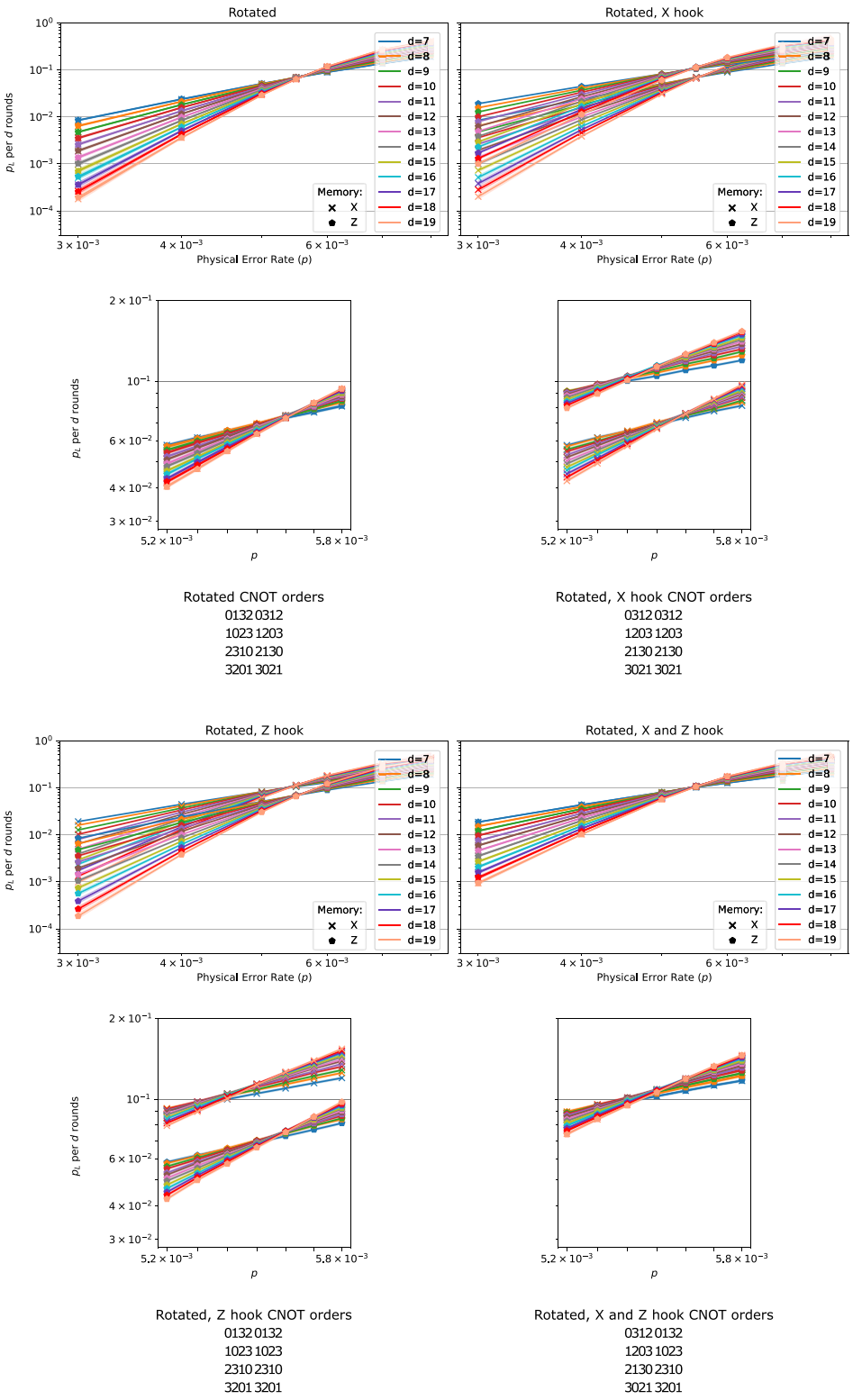


FIG. 17. Rotated surface code plots of p_L versus p from Monte Carlo sampling of simulated memory experiments implementing SD noise using all valid (see Sec. II C) CNOT orders, as well as orders with hook errors. Lower plots show zoomed-in regions of the upper plots with additional data points. The numbering used to designate CNOT order is explained in Fig. 2. Listed CNOT orders give identical plots up to uncertainty so are grouped. $X(Z)$ hook errors (Sec. II C) reduce the number of single physical errors required to form an X_L (Z_L) logical operator so increase the p_L and decrease the threshold when preserving the $|0\rangle_L$ ($|+\rangle_L$) state, corresponding to memory Z (X).

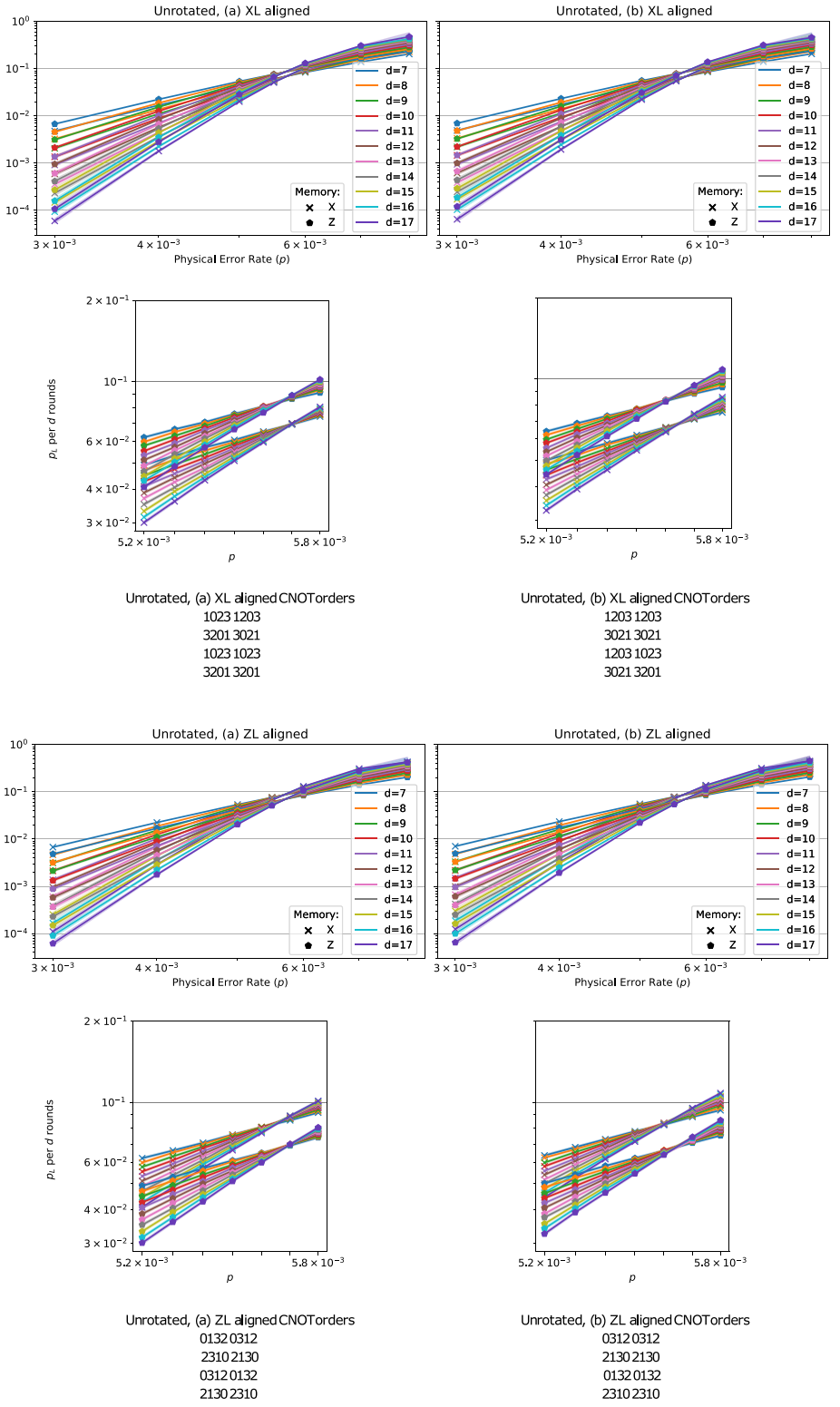


FIG. 18. Unrotated surface code plots of p_L versus p from Monte Carlo sampling of simulated memory experiments implementing SD noise using all valid (see Sec. II C) CNOT orders. Lower plots show zoomed-in regions of the upper plots with additional data points. The numbering used to designate CNOT order is explained in Fig. 2. Listed CNOT orders give identical plots up to uncertainty so are grouped. The second and third CNOT gates aligning with Z_L (X_L) correlates with reduced performance in memory X (Z) (see Sec. IV A). Additionally, (a) CNOT orders slightly outperform (b) CNOT orders, having a marginally higher threshold and lower p_L. These do not correspond to hook-error orders in the rotated code.

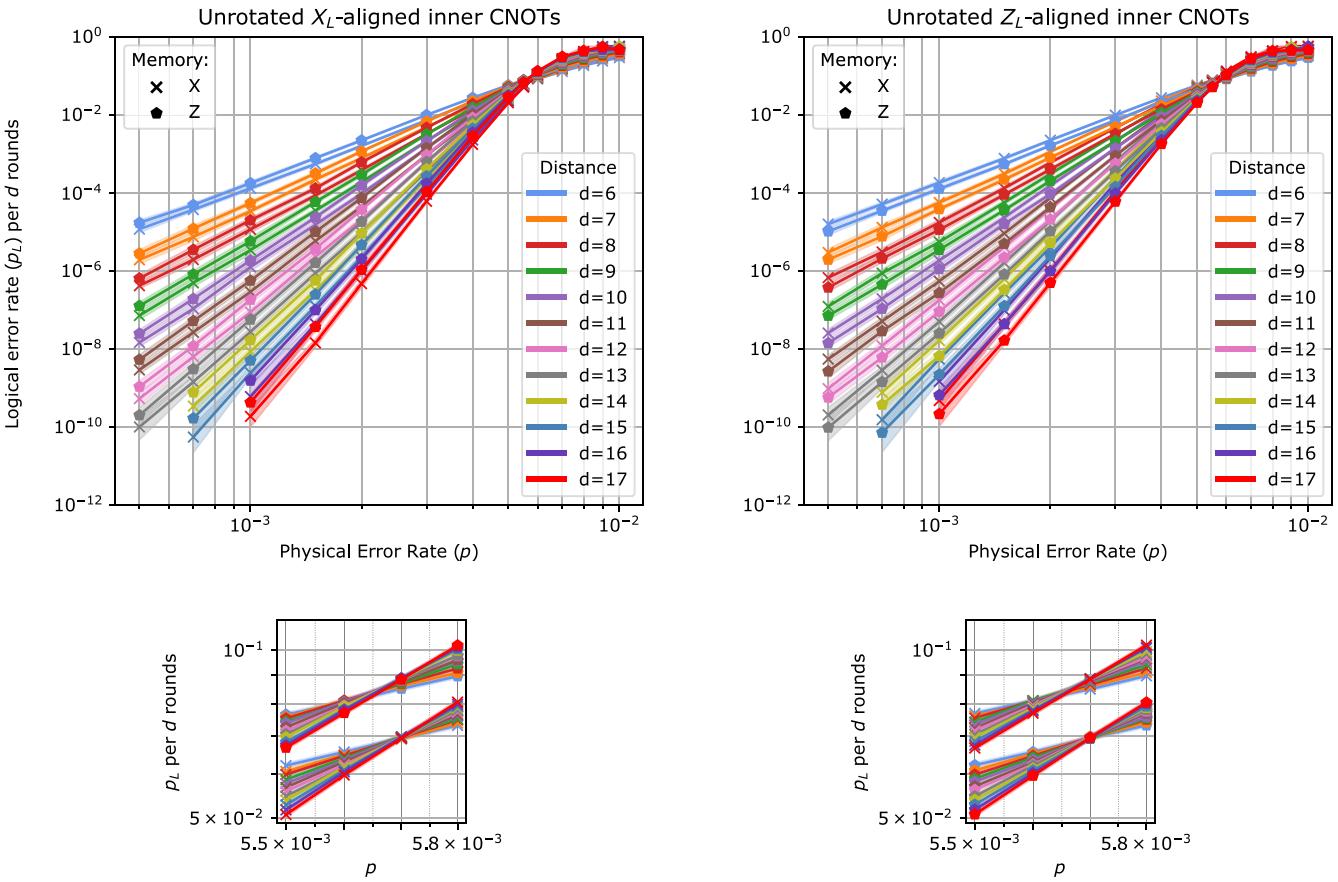


FIG. 19. Comparing the X_L-aligned inner CNOT unrotated surface code to the Z_L-aligned inner CNOT unrotated surface code. We see that their memory X and Z results invert and that the worst-case and best-case codes in each memory type are identical up to uncertainty. The Z_L-aligned plot is a repeat of the plot in the main text (Fig. 5) included for comparison. These graphs plot logical error rate (p_L) per d rounds of stabilizer measurements versus physical error rate (p) from Monte Carlo sampling of numerical simulations in unrotated surface code. Memory X (Z) preserves the $|+\rangle_L$ ($|0\rangle_L$) state. The threshold (p_{th}) is the p value below which increasing the code distance decreases p_L, i.e., where the curves intersect. Lower plots show a zoomed-in region close to p_{th} with additional simulated data points. The unrotated code shows a split in p_L between memory types depending on CNOT gate order. This was despite simulations for both codes having noise-model-equivalent X and Z-type stabilizer measurement circuits due to idling errors. If the second and third CNOT gates (from the four per stabilizer measurement circuit) were aligned with Z_L (X_L), then the p_L for memory X (Z) increased. This reproduced in all valid CNOT orders (see Fig. 18). Highlighted regions show p_L values for which the conditional probabilities $P(p_L|k)$ are within a factor of 1000 of the MLE $p_L = k/n$, assuming a binomial distribution. These simulations were under SD noise. While the results from these CNOT orders generalize (see Sec. IV A), for specificity they are 10231203 for X_L-aligned and 23102130 for Z_L aligned, using the numbering described in Fig. 2.

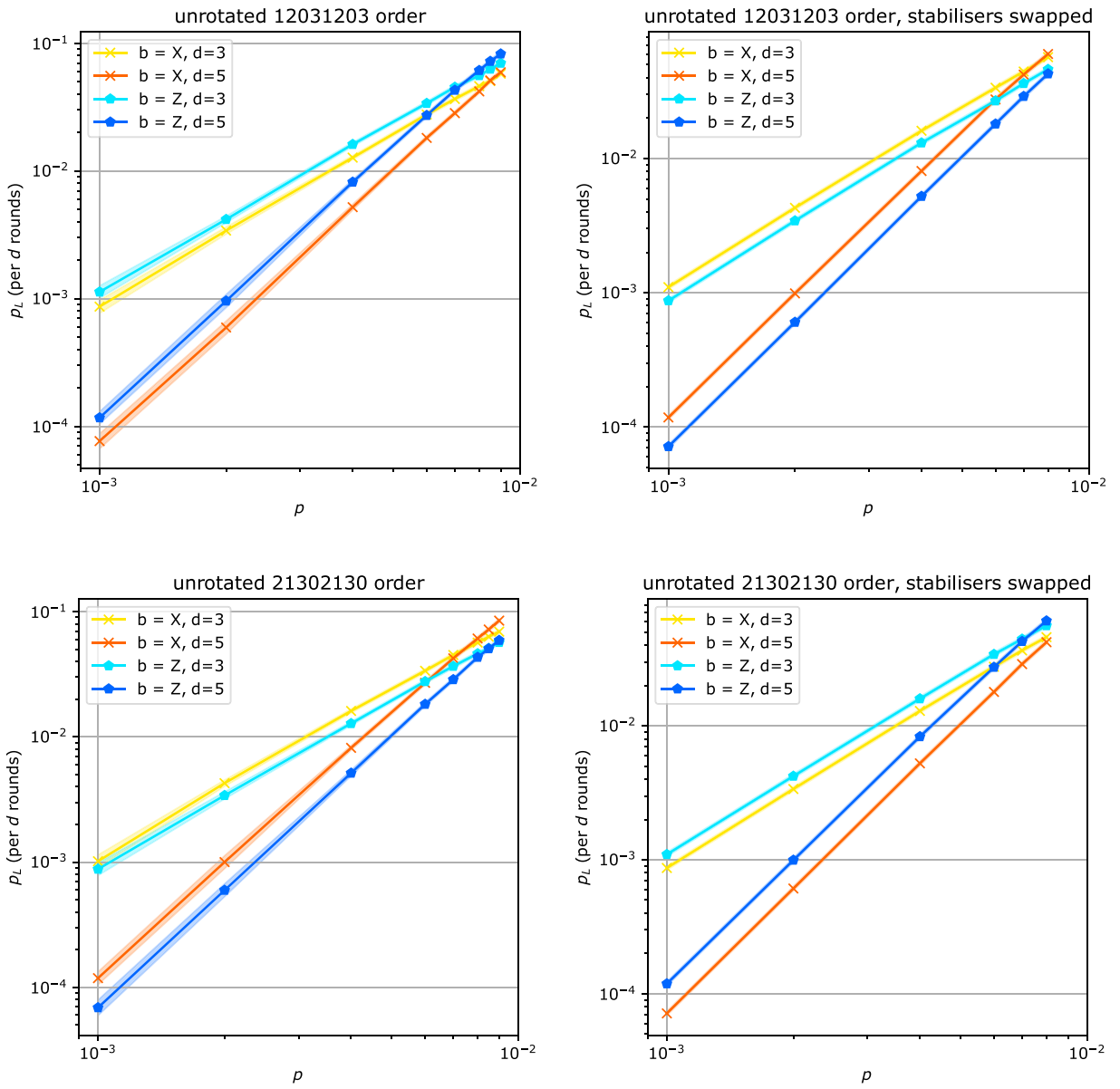


FIG. 20. Investigation of the effect on the unrotated code of inverting the stabilizers. This was to rule out the observed CNOT order affect on the unrotated code, discussed in Sec. IV A, being artificially introduced by Stim or PyMatching from the way we chose to orientate our code, namely with X_L being a vertical chain and Z_L a horizontal. Inverting the stabilizers effectively rotates the matching graph given to PyMatching by ninety degrees. The graphs on the left show the usual orientations, the graphs on the right show the orientations with the stabilizers swapped. X_L-aligned inner CNOTs become Z_L-aligned inner CNOTs and vice versa when the stabilizers are swapped, and this is reflected in the perfect inversion (up to uncertainty) between the memory X and memory Z performance. This indicates that it is not an artificial affect from, for example, a possible bias in decoding in matching along a particular orientation. Note that to isolate the effect of CNOT alignment on the unrotated code these simulations were run without errors on the Hadamard gates and without idling errors, which is inconsistent with the other results in this paper. Consequently, the exact quantities for p and p_L can be compared between plots in this figure but should not be compared to other plots in the paper which *do* simulate idling errors. These simulations implemented SD noise.

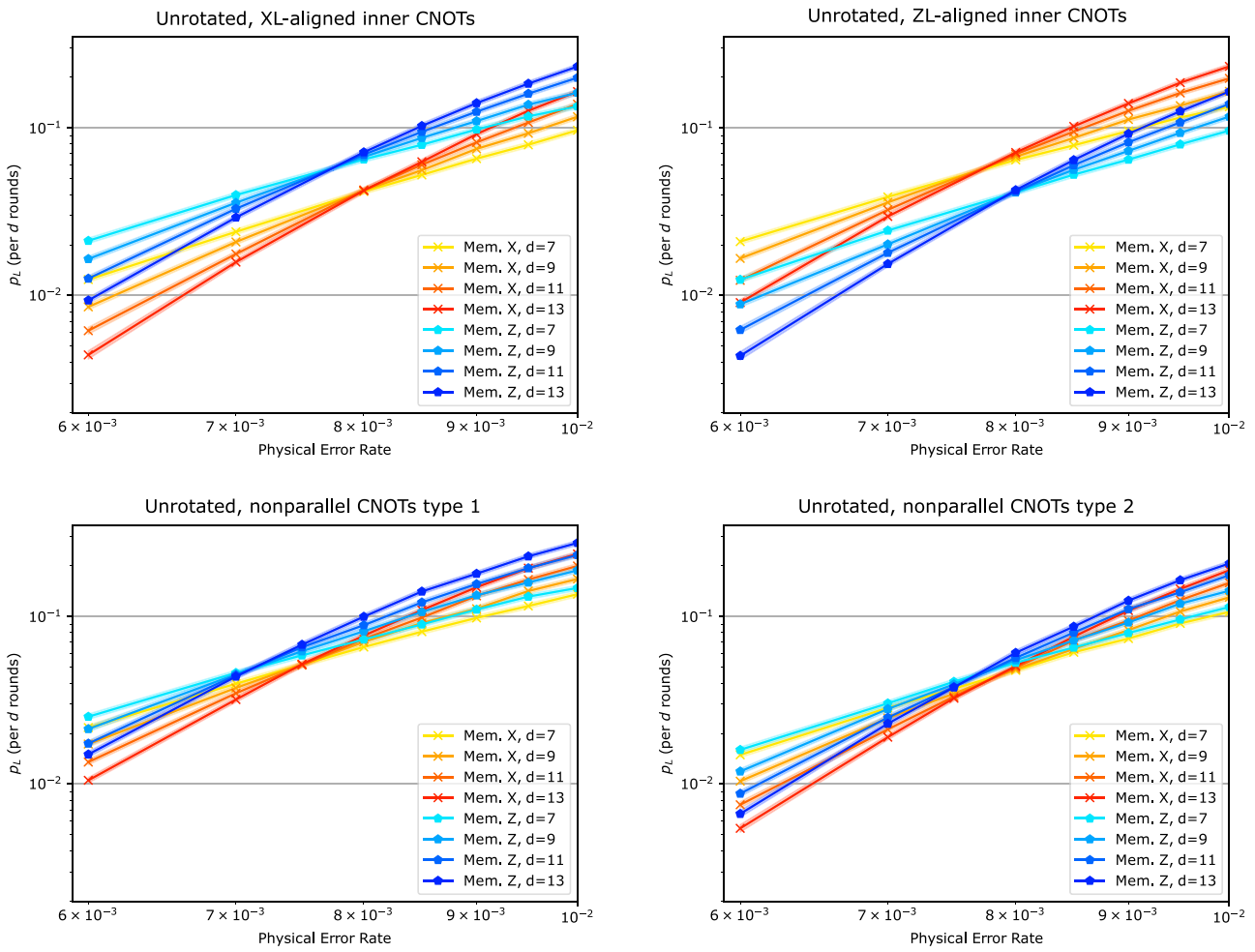


FIG. 21. Investigation of the effect on the unrotated code of using CNOT orders in the stabilizer measurement circuits which intersperse CNOT alignment between X_L and Z_L (and as a consequence feature nonparallel alignment of CNOTs; see Sec. II C). The top two graphs feature usual CNOT orders so are just for comparison. They show the inversion noted in the main text (Sec. IV A) in performance between memory X and memory Z which correlates with the inner CNOTs of the stabilizer extraction circuits being aligned with Z_L and X_L, respectively. If this inner CNOT alignment was a causative effect, then we would expect interspersing the CNOT alignments, as was done in the lower two plots, to balance out the effect. However, we see an overall increase in p_L when we do this, as shown in the lower two figures. Additionally, memory Z then consistently had a higher p_L for these CNOT orders. The CNOT orders which we refer to as "type 1" resulted in the plot in the lower left. These were 01321023, 01231032, 10230132, and 10320123. The lower right plot was for "type 2" orders 02131302 and 13020213. Note that to isolate the effect of CNOT alignment on the unrotated code these simulations were run using equal depth stabilizer-extraction circuits but without idling errors, which is inconsistent with the other results in this paper. Consequently, the exact quantities for p and p_L can be compared between plots in this figure but should not be compared to other plots in the paper which *do* simulate idling errors. These simulations implemented SD noise.

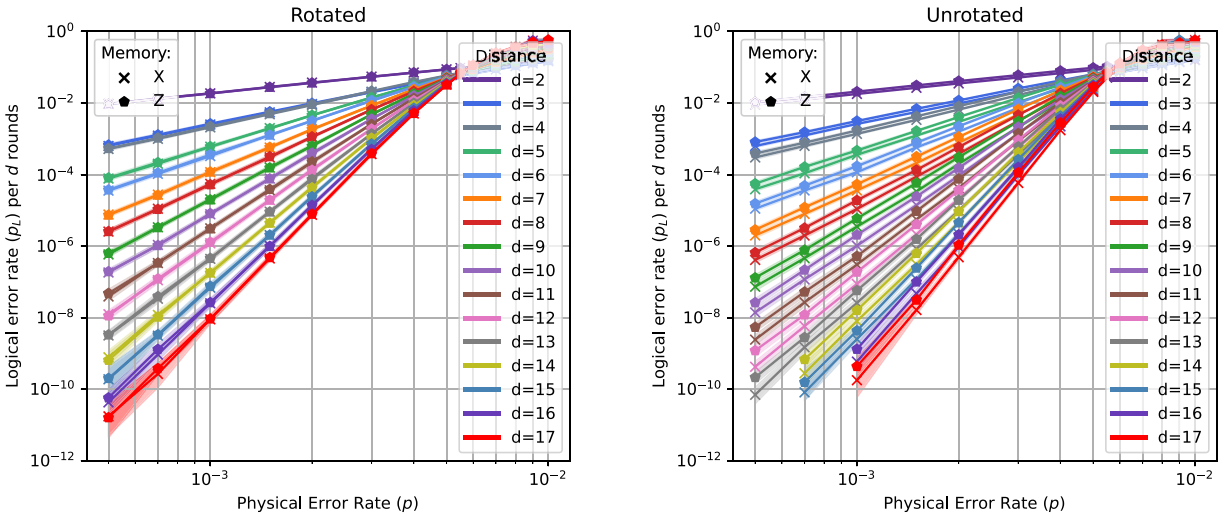


FIG. 22. Plots showing the p_L vs p (SD noise) for the rotated and unrotated surface codes including lower distance codes than presented in the main text. Plots present the same data as Fig. 5 but for $d = 2$ to $d = 17$ for both codes. We note that low-distance codes do not fit the scaling relationships of higher-distance codes and that there is a marked difference between the scaling relationship of odd and even codes at low distances but this diminishes at higher distances. With k logical errors observed, highlighted regions show p_L values for which the conditional probabilities $P(p_L|k)$ are within a factor of 1000 of the MLE $p_L = k/n$, assuming a binomial distribution and converted to per d rounds. We ran $3d$ rounds, with the per d rounds being the XOR of three independent Bernoulli distributions.

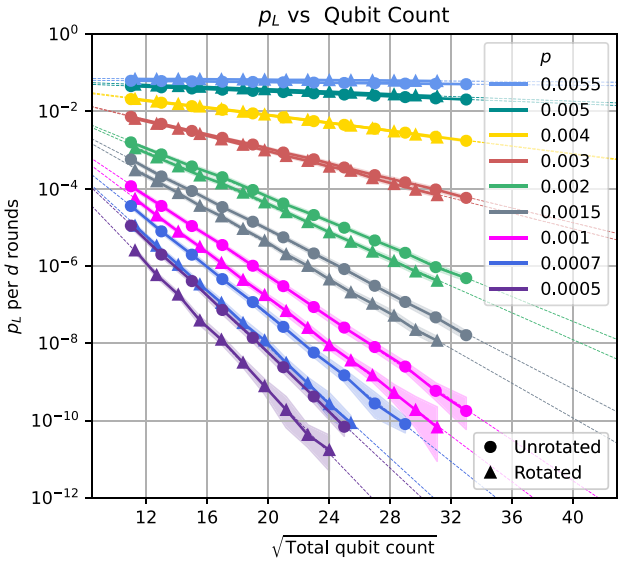


FIG. 23. Best-case unrotated code (mem. X for 10231203 order): Line-fit plot projecting p_L as a function of qubit count at various physical error rates for the rotated and best-case unrotated surface code under SD noise. We see that the best-case unrotated surface code uses less qubits than the rotated for high p very close to threshold, but at this p the qubit numbers are intractable. We fit to $d \geqslant 8$ for the rotated and $d \geqslant 6$ for the unrotated code to minimize edge effects. The points of intersection between these least-squares line fits and $p_L = 10^{-12}$ are used to generate the teraquop plot in Fig. 25. Highlighted regions show p_L values for which the conditional probabilities $P(p_L|k)$ are within a factor of 1000 of the MLE $p_L = k/n$, assuming a binomial distribution. Figure presents memory X results but is identical up to uncertainty to memory Z results when also using a best-case unrotated code CNOT ordering.

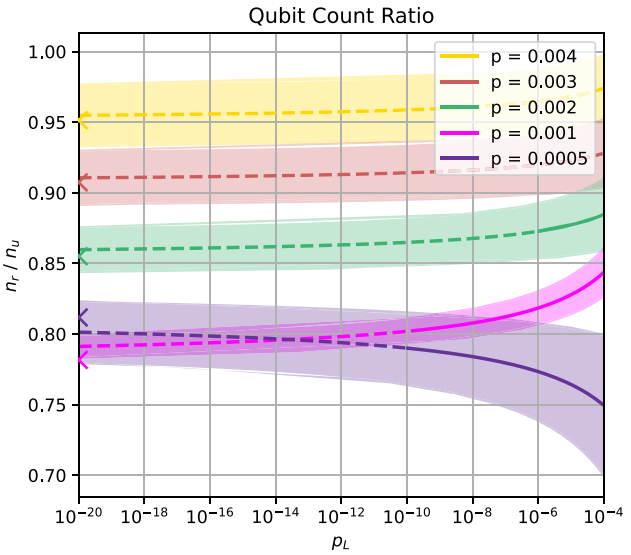


FIG. 24. Best-case unrotated code (memory X for 10231203 CNOT order): The ratio of the number of qubits used by the rotated versus the unrotated surface code to achieve the same logical error rate (p_L) per d rounds for a selection of p values. Qubit counts are calculated from the line fits of Fig. 23, with projected qubit counts indicated by dashed lines. Colored arrowheads on the y axis indicate the limit as $p_L \to 0$. Simulations implemented an uncorrelated MWPM decoder [14]. Highlighted regions indicate uncertainty propagated from the standard error of the line-fit parameters of Fig. 6. Figure presents memory X results but generalize to best-case memory Z.

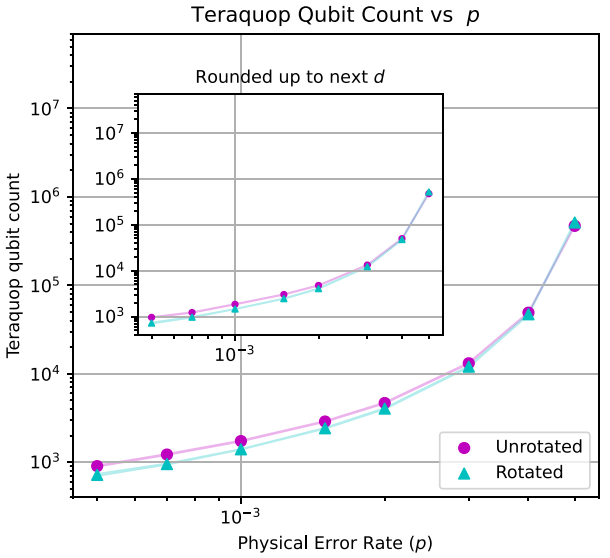


FIG. 25. Best-case unrotated code (memory X for 10231203 CNOT order): Number of qubits required to reach the teraquop regime (a p_L per d rounds of 10^{-12}) as a function of p for the rotated and best-case unrotated surface code under SD noise. The inset displays qubit counts rounded up to the next achievable code distance and has some discretization effects. Memory X results are displayed but generalize to best-case memory Z (see Sec. II C). Teraquop qubit counts are calculated using weighted least-squares line fits from from Fig. 23, with weights based on the RMSE of the MLEs for p_L, assuming a binomial distribution. Line thickness indicates uncertainty, propagated from the standard errors of the line-fit parameters using these weights.

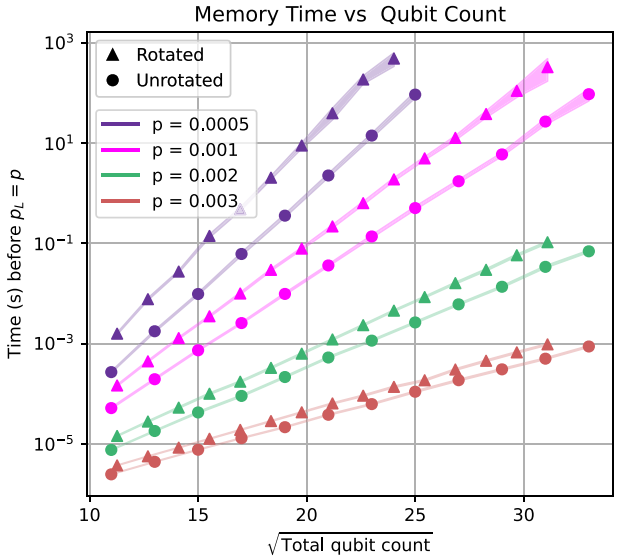


FIG. 26. Best-case unrotated code: The achievable memory times versus qubit count in the rotated and best-case unrotated surface code under SD noise. This is the length of time a state can be encoded before a logical error is equally as likely as a physical error. Calculations of memory time assume one round of stabilizer measurements takes 1 μs. Figure presents memory X results but generalize to best-case memory Z. Line thickness indicates uncertainty, calculated as the propagated RMSE of the MLEs for p_L, assuming a binomial distribution.

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Sci. Statist. Comput. **26**, 1484 (1997).

[2] I. M. Georgescu, S. Ashhab, and F. Nori, Quantum simulation, Rev. Mod. Phys. **86**, 153 (2014).

[3] D. Gottesman, *Stabilizer Codes and Quantum Error Correction* (California Institute of Technology, Pasadena, 1997).

[4] A. M. Steane, Active stabilization, quantum computation, and quantum state synthesis, Phys. Rev. Lett. **78**, 2252 (1997).

[5] S. J. Devitt, W. J. Munro, and K. Nemoto, Quantum error correction for beginners, Rep. Prog. Phys. **76**, 076001 (2013).

[6] A. Y. Kitaev, Fault tolerant quantum computation by anyons, Ann. Phys. **303**, 2 (2003).

[7] R. Acharya, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev, J. Atalaya *et al.*, Quantum error correction below the surface code threshold, Nature **638**, 920 (2025).

[8] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter, Surface code quantum computing by lattice surgery, New J. Phys. **14**, 123011 (2012).

[9] H. Bombin and M. A. Martin-Delgado, Optimal resources for topological two-dimensional stabilizer codes: Comparative study, Phys. Rev. A **76**, 012305 (2007).

[10] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, arXiv:quant-ph/9811052.

[11] A. Paler and A. Fowler, Pipelined correlated minimum weight perfect matching of the surface code, Quantum **2023**, 1205 (2022).

[12] M. E. Beverland, B. J. Brown, M. J. Kastoryano, and Q. Marolleau, The role of entropy in topological quantum error correction, J. Stat. Mech. (2019) 073404.

[13] B. Criger and I. Ashraf, Multi-path summation for decoding 2d topological codes, Quantum **2**, 102 (2018).

[14] O. Higgott and C. Gidney, Sparse Blossom: Correcting a million errors per core second with minimum-weight matching, Quantum **9**, 1600 (2025).

[15] C. Gidney, M. Newman, A. Fowler, and M. Broughton, A fault-tolerant honeycomb memory, Quantum **5**, 605 (2021).

[16] C. Gidney, M. Newman, and M. McEwen, Benchmarking the planar honeycomb code, Quantum **6**, 813 (2022).

[17] P.-J. H. Derks, A. Townsend-Teague, A. G. Burchards, and J. Eisert, Designing fault-tolerant circuits using detector error models, arXiv:2407.13826.

[18] B. M. Terhal, Quantum error correction for quantum memories, Rev. Mod. Phys. **87**, 307 (2015).

[19] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, Parallel window decoding enables scalable fault tolerant quantum computation, Nat. Commun. **14**, 7040 (2023).

[20] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Phys. Rev. A **90**, 032326 (2014).

[21] D. Bacon, S. T. Flammia, A. W. Harrow, and J. Shi, Sparse quantum codes from quantum circuits, IEEE Trans. Inf. Theory **63**, 2464 (2017).

[22] O. Higgott, T. C. Bohdanowicz, A. Kubica, S. T. Flammia, and E. T. Campbell, Improved decoding of circuit noise and fragile boundaries of tailored surface codes, Phys. Rev. X **13**, 031007 (2023).

[23] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[24] A. G. Fowler, Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time, arXiv:1307.1740.

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

[26] S. Huang, M. Newman, and K. R. Brown, Fault-tolerant weighted union-find decoding on the toric code, Phys. Rev. A **102**, 012419 (2020).

[27] Y. Wu and L. Zhong, Fusion blossom: Fast mwpm decoders for qec, in *Proceedings of the IEEE International Conference on Quantum Computing and Engineering (QCE'23), Vol. 1* (IEEE, Los Alamitos, CA, 2023), pp. 928–938.

[28] N. Liyanage, Y. Wu, A. Deters, and L. Zhong, Scalable quantum error correction for surface codes using FPGA, in *Proceedings of the IEEE International Conference on Quantum Computing and Engineering (QCE'23), Vol. 1* (IEEE, Los Alamitos, CA, 2023), pp. 916–927.

[29] A. deMarti iOlius, P. Fuentes, R. Orús, P. M. Crespo, and J. E. Martinez, Decoding algorithms for surface codes, Quantum **8**, 1498 (2024).

[30] M. McEwen, D. Bacon, and C. Gidney, Relaxing hardware requirements for surface code circuits using time-dynamics, Quantum **7**, 1172 (2023).

[31] D. Gottesman, *Stabilizer Codes and Quantum Error Correction* (California Institute of Technology, Pasadena, CA, 1997).

[32] A. M. Stephens, Fault-tolerant thresholds for quantum error correction with the surface code, Phys. Rev. A **89**, 022321 (2014).

[33] G. Q. AI, Suppressing quantum errors by scaling a surface code logical qubit, Nature (London) **614**, 676 (2023).

[34] J. Preskill, Reliable quantum computers, Proc. Roy. Soc. Lond. Ser. A: Math. Phys. Eng. Sci. **454**, 385 (1998).

[35] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).

[36] A. G. Fowler, A. M. Stephens, and P. Groszkowski, High-threshold universal quantum computation on the surface code, Phys. Rev. A **80**, 052312 (2009).

[37] Y. Tomita and K. M. Svore, Low-distance surface codes under realistic quantum noise, Phys. Rev. A **90**, 062320 (2014).

[38] M. B. Hastings and J. Haah, Dynamically generated logical qubits, Quantum **5**, 564 (2021).

[39] A. R. O'Rourke, Repository for compare the pair, github.com/wamud/compare_the_pair.

[40] C. Gidney, Stim: A fast stabilizer circuit simulator, Quantum **5**, 497 (2021).

[41] C. Gidney, Sinter source code https://github.com/quantumlib/Stim/tree/main/glue/sample (2022).

[42] C. Gidney, The stim circuit file format (.stim) https://github.com/quantumlib/Stim/blob/main/doc/file_format_stim_circuit.md.

[43] O. Higgott and F.-M. Le Régent, Stimcircuits https://github.com/oscarhiggott/StimCircuits.

[44] C. Gidney, Inbuilt stim circuits showing unexpected memory x and memory z logical error rates for unrotated surface code, StackExchange answer in https://quantumcomputing.stackexchange.com/questions/32941/inbuilt-stim-circuits-showing-unexpected-memory-x-and-memory-z-logical-error-rat/32959#32959 (2023).

[45] C. T. Chubb, General tensor network decoding of 2d Pauli codes, arXiv:2101.04125.

[46] S. Saadatmand, T. L. Wilson, M. Field, M. K. Vijayan, T. P. Le, J. Ruh, A. S. Maan, I. Moflic, A. Caesura, A. Paler *et al.*, Fault-tolerant resource estimation using graph-state compilation on a modular superconducting architecture, arXiv:2406.06015.

[47] S. J. Devitt, A. D. Greentree, A. M. Stephens, and R. Van Meter, High-speed quantum networking by ship, Sci. Rep. **6**, 36163 (2016).

[48] G. Üstün, A. Morello, and S. Devitt, Single-step parity check gate set for quantum error correction, Quantum Sci. Technol. **9**, 035037 (2024).

[49] D. K. Tuckett, A. S. Darmawan, C. T. Chubb, S. Bravyi, S. D. Bartlett, and S. T. Flammia, Tailoring surface codes for highly biased noise, Phys. Rev. X **9**, 041031 (2019).

[50] D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, Fault-tolerant thresholds for the surface code in excess of 5% under biased noise, Phys. Rev. Lett. **124**, 130501 (2020).