Quantum Information and Computation, Vol. 22, No. 11&12 (2022) 0947–0964
© Rinton Press

AUTOMATED DISCOVERY OF LOGICAL GATES FOR QUANTUM ERROR CORRECTION

HONGXIANG CHEN
Department of Computer Science, University College London Gower Street London
Gower St, London WC1E 6EA
Odyssey Therapeutics, Inc.
301 Binney Street, Cambridge, MA 02142
h.chen.17@ucl.ac.uk

MICHAEL VASMER
Perimeter Institute for Theoretical Physics
Waterloo, ON N2L 2Y5, Canada
Institute for Quantum Computing, University of Waterloo
Waterloo, ON N2L 3G1, Canada
mvasmer@pitp.ca

NIKOLAS P. BREUCKMANN
Department of Physics & Astronomy, University College London
Gower St, London WC1E 6EA
n.breuckmann@ucl.ac.uk

EDWARD GRANT
Department of Computer Science, University College London Gower Street London
Gower St, London WC1E 6EA
Odyssey Therapeutics, Inc.
301 Binney Street, Cambridge, MA 02142
edward.grant.16@ucl.ac.uk

Received November 30, 2021
Revised June 16, 2022

Quantum error correcting codes protect quantum computation from errors caused by decoherence and other noise. Here we study the problem of designing logical operations for quantum error correcting codes. We present an automated procedure that generates logical operations given known encoding and correcting procedures. Our technique is to use variational circuits for learning both the logical gates and the physical operations implementing them. This procedure can be implemented on near-term quantum computers via quantum process tomography. It enables automatic discovery of logical gates from analytically designed error correcting codes and can be extended to error correcting codes found by numerical optimization. We test the procedure by simulating small quantum codes of four to fifteen qubits showing that our procedure finds most logical gates known in the current literature. Additionally, it generates logical gates not found in the current literature for the [[5,1,2]] code, the [[6,3,2]] code, the [[8,3,2]] code, and the [[10,1,2]] code.

Keywords: Quantum computation, Error correction, Machine learning

1 Introduction

Quantum errors stem from undesired interactions with an outside environment. There are many different interactions that may occur and their nature as well as their strength depend on the particular hardware architecture. It was shown by Shor that despite the existence of quantum errors, we can preserve quantum information by using *quantum error correcting codes* [1]. The main idea is similar to what is done in classical error correction: redundancy is introduced by encoding some number of qubits k into a larger number of *physical qubits* n. This is done by mapping the states of the k logical qubits into non-local degrees of freedom of a highly entangled state of the physical qubits. A construction due to Calderbank, Shor and Steane (CSS) generates a quantum code from two linear classical codes [2, 3].

In quantum computation, in addition to preserving quantum information, we need to manipulate quantum states. This is done by using a small set of unitary operators called the *quantum gates*. Quantum gates applied to the encoded qubits are called *logical gates*. As the application of gates is prone to errors itself we would like to implement the logical gates using shallow depth circuits, meaning that only a few gates are being applied to the physical qubits to produce the logical gate. Finding such circuits is a major challenge in quantum error correction and has thus far been done on a case-by-case basis. For example, codes which are generated by the CSS construction (CSS codes) and encode a single logical qubit are known to have a fault-tolerant CNOT gate: it can be implemented on two copies of the same code by applying a CNOT between all pairs of physical qubits.

Here we apply the technique of variational circuit optimization to find fault tolerant logical gates for a given quantum error correcting code. Variational circuits have been used to parametrize wavefunctions, i.e. parametrize a unitary circuit transforming the $|0\rangle$ initial state. This approach has been used successfully to perform quantum chemistry calculations on Noisy Intermediate-Scale Quantum (NISQ) computers [4–8]. The popular basic building blocks, called *unit cells*, of these variational circuits are rotational gates R_x, R_y, R_z and CNOT gates (see the nomenclature and notation section of Ref. [9] for a definition). The specific mathematical properties of these gates makes optimizing the variational circuit relatively easy [10, 11]. In our case, we propose using variational circuits to find fault-tolerant logical gates for a given quantum error correcting code. Our procedure finds logical gates and their quantum circuit implementations by numerically optimizing the variational circuit ansatz for both logical gates and their physical implementations (Fig. 1).

Our procedure offers several benefits and much flexibility. The ansatz used for the physical operation can be tailored to take advantage of the properties of a specific quantum computing architecture. The ansatz for the logical gate is variational and thus our procedure automates the discovery of logical gates given a quantum error correcting code. The procedure can also target a specific logical gate when we fix the ansatz to this gate. Therefore, for stabilizer codes and in particular for non-CSS codes, our procedure provides a straightforward first choice to find logical gates. Furthermore, the procedure can be implemented on a quantum computer using quantum process tomography and is resource friendly for quantum codes requiring a small number of physical qubits. It is hence feasible for an implementation on near-term quantum computers.

We note that in the literature, previous research has applied numerical optimization techniques to the field of quantum error correction for different purposes. Work in Ref. [12–15] used optimization algorithms (mostly convex optimization algorithms) to find error correcting quantum channels. Work in Ref. [16] demonstrated learning a circuit for preserving quantum information using a variational ansatz circuit. In Ref. [17, 18] authors constructed quantum error correcting codes using neural networks. There is also a body of research using neural networks for decoding [19–23]. In this work, we applied similar optimization techniques to the novel problem of finding logical operators for quantum error correcting codes.

Paper structure. In Section 2, we describe our procedure for finding logical gates. We then discuss the numerical experiments on a classical computer where we apply the procedure to several CSS codes and non-CSS codes in Section 3. We present in detail the experiment configurations (ansatz circuit structure and optimization algorithm) and the logical gates we found using our method. Among these results there are several new logical gates for the [[5,1,2]] code, the [[6,3,2]] code, the [[8,3,2]] code, and the [[10,1,2]] code, which to our knowledge have not appeared previously in the literature, and which we discuss in detail in the Section 3.2. Finally, we make several comments on the benefits and disadvantages of our procedure in Sec 4, where we mention specifically the scalability of our procedure. The new logical gates we found are attached with this paper in the Supplementary Materials.

2 Method

Here we present the procedure to find circuits that implement logical gates for error correcting codes. The procedure is inspired by the idea of circuit learning and uses ansatz circuits for both the logical gate and the physical operations that implement this logical gate in the encoded Hilbert space. Before we define this procedure, we first introduce the notation that we use throughout this paper.

Commonly, an error correcting code encodes logical qubits (whose corresponding Hilbert space will be denoted as H_A), into the subspace, denoted by L, of another Hilbert space $H' = H_A \otimes H_B$. This mapping is unitary and is denoted as $E : H_A \to L \subset H'$. We call G a physical operation implementing the logical gate g if it is a unitary automorphism on L such that $E^{-1}GE\,|\psi\rangle = g\,|\psi\rangle$ for states $|\psi\rangle \in H_A$.

Now we present the procedure. We first describe how the procedure can be performed via simulation on a classical computer, and we describe its extension to quantum computer afterwards. On a classical computer, we simulate the encoding E, the physical operation, and the inverse encoding E^{-1}. While the encoding/inverse encoding circuit is fixed by the choice of a particular error correcting code, we use ansatz circuits for both the physical operation G and the logical gate g. For the physical operation we choose an ansatz that we know is fault-tolerant for the code under consideration. In most cases this is just a transversal gate, i.e., a gate that does not couple physical qubits within the same code block. The ansatz for the physical operation is variational and may map logical states outside the logical space. Because of this possibility, we apply a projector onto the codespace after the physical operation. This is achieved by measuring the stabilizer generators and applying a correction, where the correction is the minimum weight error compatible with the observed syndrome. The stabilizer measurements and corrections are simulated classically as a unitary circuit acting on the extended Hilbert space $H' \otimes H_C$. Here the qubits in the H_C are all initialized to the $|0\rangle$ state. We use a variational circuit as the ansatz for the logical gate, to enable automatic discovery of logical gates for the quantum error correcting code. Alternatively, we can set a fixed unitary gate as the logical gate, and try to vary the ansatz for physical operation to find an implementation for this unitary gate.

Our goal is to optimize the parameters for the ansatz circuits such that the for all possible input quantum states, a physical operation G together with the encoding, syndrome removal, and decoding circuit, act in the same way as a logical operation g (See Fig.1). Specifically, we minimize a loss function $\mathcal{L}$ defined by

$$\mathcal{L}(\theta_1, \theta_2) = \sum_i (1 - \mathrm{F}\left(\mathrm{tr}_{H_B \otimes H_C}\left(|\phi_{1,i}\rangle\langle\phi_{1,i}|\right), |\phi_{2,i}\rangle\langle\phi_{2,i}|\right)). \tag{1}$$

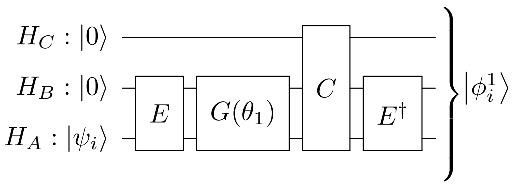
Here ϕ_i^1 (ϕ_i^2) is the output of the trial logical circuit (comparison circuit) (see Fig.1). The input states $\{\psi_i\}_i^N$ form a tomographically complete set (the particular set of states we used in simulation is described in the Section 3.1). The trace (tr) is taken over the Hilbert space $H_B \otimes H_C$. For two quantum states ρ and σ, we measure the distance between them by $\mathrm{F}(\rho, \sigma)$. In our case, we define F to be the fidelity between two input density matrices:

$$F(\rho, \sigma) = \frac{1}{2}\,\mathrm{tr}\left[\sqrt{(\rho - \sigma)^2}\right] \tag{2}$$

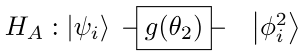
When the minimization is successful and the average distance is zero excluding floating-point errors, the procedure succeeds in finding a logical gate for this code.

Our method can be adapted to run on a quantum computer in order to find logical gates for larger error correcting codes and to tailor the ansatz to the specific quantum computing architecture. We briefly outline such an extension here. The main change in regard to the classical implementation concerns the encoding and inverse encoding circuits. Instead of implementing a (non fault-tolerant) encoding circuit, we need a certain method of reliably preparing encoded Pauli eigenstates on the quantum computer. Given these states, we apply the logical operation ansatz in the same way as the classical implementation. We can also implement an error correcting procedure after applying the logical operation ansatz. Finally, instead of implementing the inverse encoding circuit, we envisage performing logical measurements of the encoded states in the X, Y and Z bases. Using these measurement outcomes, we perform logical state tomography [9, 24] on the output states, and compare the tomography results with the unencoded states obtained via classical simulation, using an analogous loss function to the one shown in Eq. (1).

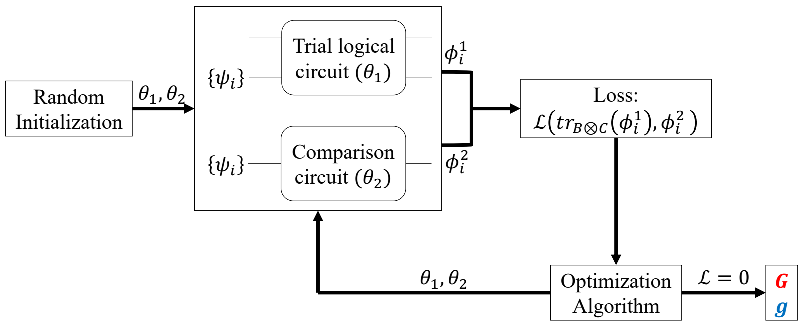
The requirements of implementing our method on a quantum computer are relatively minor, as long as the number of logical qubits in the code is modest. The only subroutines we would need to implement on the quantum computer are: preparing encoded Pauli eigenstates, applying a physical operation ansatz, and measuring Pauli observables. However, the number of required logical Pauli measurements grow exponentially in the number of logical qubits, which might be a bottleneck of our proposed method. Potentially one can redefine the distance function F in the loss function (Eq. (1)) to be the infidelity between two density matrices, and utilize a swap test [25, 26] for calculation of the loss function. Specifically, we may prepare in another error-corrected quantum computer the unencoded states, and analogously calculate the loss function in Eq. (1) by a cross-device swap test. We emphasize that our procedure can be applied to any code that can prepare encoded Pauli eigenstates and measure Pauli observables, i.e. it is not limited to qubit stabilizer codes.



(a) Trial logical circuit



(b) Comparison circuit



(c) Procedure

Fig. 1. Learning logical gates

To learn logical gates, we optimize angles θ_1 and θ_2 such that the trial logical circuit and the comparison circuit perform effectively the same unitary as measured by our loss function $\mathcal{L}$. Inside the trial logical circuit (Fig. 1a), the encoding/inverse encoding circuit $E/E^\dagger$ maps input states $\{\psi_i\}_1^N$ into/out of the logical space. The ansatz $G(\theta_1)$ performs a series of physical operations and the circuit C performs the stabilizer measurements and minimum weight error correction. In the comparison circuit (Fig. 1b), only the ansatz circuit $g(\theta_2)$ for the logical gate is performed. The whole procedure starts by randomly initializing the two angles in both circuits, and then a tomographically complete set of quantum states are fed to the two circuits as inputs. Their outputs are gathered and fed to the loss function $\mathcal{L}$. The calculated loss function values are fed to a minimization algorithm, which outputs new angles for θ_1 and θ_2. Then we rerun the trial logical circuit and the comparison circuit again and the whole loop continues until $\mathcal{L}$ is zero excluding float point errors, in which case the circuit $G(\theta_1)$ will perform the logical gate $g(\theta_2)$ on the encoded space.

3 Numerical experiments for small codes

We applied our procedure to a variety of small codes, as summarized in Table 1. Of particular interest are the cases where non-Pauli logical gates were found by the experiment. In this section, we first describe the detailed experimental configurations for all the numerical experiments, and then we highlight the results of the experiments for the following codes: [[8,3,2]], [[8,2,2]], [[6,3,2]], and [[7,1,3]] surface code with a twist.

3.1 Experiment configurations

Ansatz: We start with transversal ansatz for physical operations for most of the simulation experiments shown on Table 1, except for the [[5,1,2]] code and the [[10,1,2]] code. A transversal ansatz is a natural starting point for experimentation because it is naturally fault-tolerant. A transversal ansatz is formed by using three single qubit rotation gates ($\mathrm{R_j} = e^{-i\theta\sigma_j/2}$, where

Table 1. Summary of quantum gates found by our procedure using classical simulation for a variety of small quantum error correcting codes.

Code	**Logical gate found by the procedure**	**Percentage of non-Pauli gates**
[[4,1,2]] [27], [[8,2,3]] [28], [[8,3,3]] [29–31], [[11,5,3]] [30], [[12,6,3]] [28], [[13,7,3]] [28], [[14,8,3]] [28], [[15,7,3]] [2, 30]	Pauli group	0 %
[[4,2,2]] [32, 33]	Pauli group, CNOT	N/A
[[5,1,2]] [34]	Pauli group, $\mathbf{S}^{\dagger}$, $\mathbf{H}$	71.4%
[[5,1,3]] (Five-qubit code) [35, 36]	Pauli group, $e^{i\pi/4}SH$, $e^{i3\pi/4}XHXS^{\dagger}$, $e^{-i3\pi/4}XHXS$, $e^{-i\pi/4}HS^{\dagger}$, $e^{-i\pi/4}S^{\dagger}H$, $e^{-i3\pi/4}SXHX$, $e^{i3\pi/4}S^{\dagger}XHX$	88.0%
[[6,3,2]] [28]	Pauli group, $\mathbf{H}_{12}\mathbf{CZ}_{12}\mathbf{H}_{12}$	60.0%
[[7,1,3]] (Steane code) [2, 3]	Pauli group, Generators for the group generated by H and S	N/A
[[7,1,3]] (Surface code with a twist) [37]	Pauli group, $e^{-i3\pi/4}SXHX$, $e^{i3\pi/4}S^{\dagger}XHX$	11.1%
[[8,2,2]] (Projective plane 2D color code) [38]	Pauli group, CZ, $H^{\otimes 2}CZH^{\otimes 2}$, $H^{\otimes 2}SWAP$	20.0%
[[8,3,2]] [39, 40]	Pauli group, $\mathbf{CZ}_{12}$, $\mathbf{CZ}_{13}$, $\mathbf{CZ}_{23}$, CCZ	71.4%
[[10,1,2]] [34]	Pauli group, $\mathbf{S}^{\dagger}$, $\mathbf{T}^{\dagger}$	51.6%

[1] The new logical gates we found, which have not been reported in the literature, are highlighted in blue and in boldface. For these logical gates, the parity check matrix of the corresponding quantum code and the physical operations which implement them are provided in OpenQASM [41] format in the Supplementary Materials.

[2] The exact experimental configuration and optimization algorithm we used is discussed in Section 2.

[3] We found a generating set of logical Pauli gates for the codes that we have labeled Pauli group in the column *Logical gate found by the procedure*.

[4] In the third column, we list the percentage of non-Pauli gates founded among all optimization runs. Note that repeated discoveries of the same logical gate count towards the percentage. Entries equal to ‘N/A’ are cases where we set the logical gate ansatz to specific gates.

$\{\sigma_j\}_{j=x,y,z}$ are the three Pauli matrices) on each physical qubit, where each rotation gate has its own angle that can be adjusted independently of other rotation gates. An illustration of transversal ansatz is provided in Fig. 2. For the parameterization of the logical gate $g(\theta_2)$, we use the ansatz that parametrizes an arbitrary unitary transformation on the Hilbert space H_A. Denote the number of qubits in H_A as n_a. When $n_a = 1$, the ansatz is simply the three single qubit rotation gates mentioned before. When $n_a = 2$, the ansatz is the circuit shown in Fig.2 in Ref. [42]. When $n_a = 3$, we obtain a circuit parameterization for arbitrary three qubit unitary gates using the QSD decomposition provided by Ref. [43]. It consists of 17 CNOT gates and 69 single qubit rotation gates whose rotation angles can be adjusted independently. A visualization of the circuit parameterization where the 69 parameters are labelled with integers from 0 to 68 is available online [44]. When $n_a > 3$, only the first three qubits are selected on which we apply the three qubit ansatz mentioned previously. This is because the exponential increase of the possible ansatz circuits makes experimentation infeasible.

The rationale behind our choice of ansatz for the different values of n_a is the following. We want to give our procedure as much freedom as possible in finding logical gates. Therefore, the ansatz for the logical gate is always a parameterization of arbitrary unitary gates on n_a qubits. This way we make no assumptions about the structure of the code and the fault-tolerant logical gates it admits. In addition, the parameterization is based on a gate library consisting of single qubit rotation gates and fixed two qubit gates, which makes it possible to use Rotosolve optimization algorithm. We expect that we would obtain similar results if we used different parameterizations of arbitrary unitary gates as long as satisfy the condition required by the Rotosolve algorithm, but we have not experimentally confirmed this.

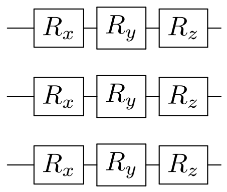


Fig. 2. An illustration of a transversal ansatz on three qubits. It is formed by using three single qubit rotation gates (R$_j$ = $e^{-i\theta\sigma_j/2}$, where $\{\sigma_j\}_{j=x,y,z}$ are the three Pauli matrices) on each physical qubit. Each rotation gate has its own angle that can be adjusted independently of other rotation gates.

We note that for the experiments on the $[[4, 2, 2]]$ code, the encoder used only encodes one of the two logical qubits, and we only experimented on this logical qubits. In addition, as an early stage proof of principle that our procedure can find entangling gates, we targeted the logical gate CNOT for this code $[[4, 2, 2]]$ in one experiment. In this experiment, we encoded a pair of qubits in two copies of the [[4,2,2]] code (one encoded qubit per code). We then used a transversal two-qubit gate ansatz, where each gate coupled corresponding qubits in the different codes. Using this ansatz, we were able to find a transversal logical CNOT gate between the codes. The physical operation that implemented this logical gate is simply CNOT gates between corresponding qubits in the different codes. This shows that our method is capable of finding gates acting between separate codes.

In the case of the [[5,1,2]] code, we used a non-transversal but still fault-tolerant ansatz for the logical gate. Specifically, we used a transversal ansatz (as described above) for the

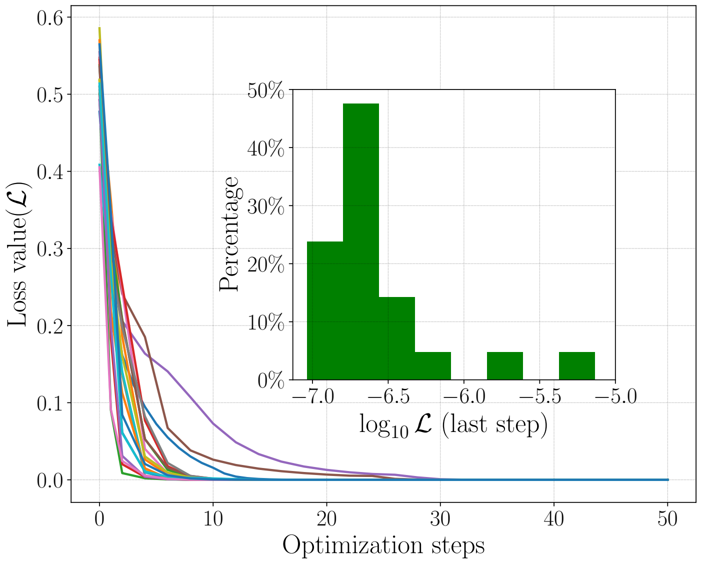


Fig. 3. Loss function value as optimization progress step-by-step for the experiments for the code [[5,1,2]]. The inset plot shows the percentage distribution of the logarithm of loss function value $\mathcal{L}$ in the last optimization step.

first three of the qubits, and a parameterization of arbitrary two-qubit unitary gates for the final two qubits. This choice of ansatz was motivated by the structure of the code in Ref. [34]. The fact that we found logical gates (including non-Pauli gates, see Table 1) using this ansatz shows that our procedure can work well with ansatz circuits that are tailored to a particular code.

Similarly, for the [[10,1,2]] code, we also used a non-transversal but still fault-tolerant ansatz for the logical gate. Specifically, we used a parameterization for arbitrary three qubit unitary gates (obtained by the `QSD` decomposition in Ref. [43][44]) on qubits numbered 4, 6, and 8. We used a transversal ansatz (as described above) for the rest of the qubits.

Optimization: We used the algorithm `Rotosolve` [10] to minimize $\mathcal{L}$. `Rotosolve` specializes in minimizing any function whose dependency with respect to any of its variables x is a sine function of the form $a\sin(x+b)+c$, where a, b and c are constants. One particular function of this form is $f(\theta) = \langle 0|\, U(\theta)^\dagger H U(\theta)\, |0\rangle$, where U is the unitary transformation of a quantum circuit containing constant unitary gates and variational rotation gates Rx, Ry, and Rz. It is easy to see the loss function defined in Eq. (1) follows the same form up to a constant if we consider $\mathrm{tr}_{\mathrm{ancilla}}(|\phi_{2,i}\rangle\langle\phi_{2,i}|)$ to be the H, and $g(\theta_2)$ to be the U.

To confirm the success of optimization, we have plotted the value of loss function $\mathcal{L}$ (Eq. (1)) for 21 optimization runs with random independent initial parameters for the code [[5,1,2]] in Fig. 3. For experiments on other codes, their loss function plots look similar. We can observe in Fig. 3 that the loss function quickly got close to the minimal value with the help of `Rotosolve` optimization algorithm. We also note that the loss function values after the final step are within the range $(10^{-7}, 10^{-4})$. For comparison, we calculated the loss function $\mathcal{L}$ using decomposition of known logical gates in the literature and found $\mathcal{L} \in (10^{-7}, 10^{-6})$. Therefore, we can conclude that for $L \approx 10^{-7}$ or 10^{-6}, the optimization is successful. We also believe that for $L \in (10^{-5}, 10^{-4})$, the loss function can be further decreased if we allow optimization process to run with more iterations. However, in this case, we took a different but more economical approach, in which we stopped the algorithm, took the last angle θ_1 to give $g(\theta_1)$ and $G(\theta_1)$, found the closest analytical expression for $g(\theta_1)$ and $G(\theta_1)$, and tested the loss function $\mathcal{L}$ using the analytical expressions. Typically, we can find analytical expression using common Clifford gates, and the loss function calculated is 0. This shows that the optimization is only correcting for the numerical errors in the last few steps.

We note that for performance purposes, the simulation was carried out using proprietary software written by the author H.C. for Rahko Ltd.

Tomographically Complete Set: For calculating the cost function in Eq. (1), we need to provide the initial tomographically complete set of quantum states, since for any quantum channel $\mathcal{E}$, its action on the any quantum state ρ can be uniquely determined by its action on a tomographically complete set $\{\psi_i\}_i$[9].

A straight-forward example of a tomographically complete set is $\{\psi_i\}$ such that the set $\{|\psi_i\rangle\langle\psi_i|\}_i$ spans the vector space of all density matrices. In our case, we only need a set $S_1 = \{|\psi_i\rangle\langle\psi_i|\}_i$ of one-qubit states such that S_1 spans all 2×2 density matrices, since for $2^n \times 2^n$ density matrices representing n-qubit states, tensor products of the same set S of one-qubit states will span the vector space of all n-qubit states.

Now we try to construct S_1 from well-known quantum states. Consider the vector space of one-qubit density matrices as a subspace of $\mathbb{C}^4$. When S_1 spans the whole vector space $\mathbb{C}^4$, clearly S_1 will span the smaller subspace of one-qubit density matrices. Then, it suffice to check whether the elements of the set S_1, when considered as vectors in $\mathbb{C}^4$, could span the whole vector space $\mathbb{C}^4$.

Now we describe the six states, which we call the *six Bloch states*, which we use as the initial states for our experiments. These states are:

$$S_1 = \{|0\rangle\langle 0|\,, |1\rangle\langle 1|\,, |+\rangle\langle +|\,, |-\rangle\langle -|\,, |+i\rangle\langle +i|\,, |-i\rangle\langle -i|\}. \tag{3}$$

It is easy to check that the six Bloch states, considered as six 4-dimensional vectors, span the whole $\mathbb{C}^4$ space, by checking the rank of the matrix formed by putting each of the six density matrices into a column of the matrix. We have therefore used tensor products of the six Bloch states to construct the tomographically complete set for inputs to our loss function Eq. (1).

We note that with hindsight, we could have reduced the size of the tomographically complete set for n-qubits from 6^n to 4^n by keeping only four out of the six Bloch states, i.e. $S_1 = \{|0\rangle\langle 0|\,, |1\rangle\langle 1|\,, |+\rangle\langle +|\,, |+i\rangle\langle +i|\}$. It is easy to check that this remaining four states also span $\mathbb{C}^4$.

3.2 Illustrative results on small codes

From all the logical gates found in Table 1, we highlight the results for a few codes here.

[[8,3,2]]: The [[8,3,2]] code is the smallest non-trivial 3D color code [39, 40]. We found three transversal Clifford gates for this code which (to the best of our knowledge) have not previously appeared in the literature: CZ_{12}, CZ_{13}, and CZ_{23}, where CZ_{ij} denotes a logical CZ acting on logical qubits i and j. Our optimization procedure also provided a simple implementation of these gates, which we now explain.

Logical gates in the [[8,3,2]] code can be understood geometrically. Suppose we place qubits on the vertices of a cube, as shown in Fig. 4. The stabilizer group of the [[8,3,2]] code can be generated by an operator consisting of Pauli-X operators acting on every qubit, alongside operators that are associated with the faces of the cube, i.e. for each face, f, we have an operator $\Pi_{v\in f}Z_v$, where Z_v denotes a Pauli-Z operator acting on the qubit on vertex v. With this definition, the logical X operators of the code are associated with the faces of the cube ($\overline{X}_1 = \Pi_{v\in f}X_v$ etc.) and the logical Z operators are associated with the edges of the cube. We note that opposite faces support logical X operators that act on the same encoded qubit, and the corresponding logical Z operators are supported on the edges linking these faces.

To implement a logical $\overline{CZ}_{ij}$ gate, we apply $S = \text{diag}(1, i)$ and $S^\dagger$ gates in an alternating pattern to the vertices of a face that supports $\overline{X}_k$. We show this operator in Fig. 4b and we denote it by U. We now show that U implements a $\overline{CZ}_{ij}$ gate: namely it maps $\overline{X}_i$ to $\overline{X}_i\overline{Z}_j$, and has no effect on all other logical operators. First, we note that all operators that consist exclusively of Pauli-Z operators are unaffected by U because S and Z commute. Therefore, Z-type stabilizers and logical Z operators are mapped to themselves by U. Next we consider operators that contain Pauli-X, which are transformed by S as follows: $SXS^\dagger = Y = iXZ$. Consider the X stabilizer of the code (X on all the qubits). It is straightforward to see that U maps this operator to a product of itself and a Z stabilizer associated with the face where we applied U (the alternating pattern causes the factors of i and $-i$ to cancel). Finally, consider $\overline{X}_i$ (the blue face in Fig. 4b). This operator is mapped to a product of itself and a $\overline{Z}_j$ operator with support on an edge that links the $\overline{X}_j$ faces (the green edge in Fig. 4b). Similarly, $\overline{X}_j$ is mapped to $\overline{X}_j\overline{Z}_i$

In addition, our procedure found a transversal implementation of CCZ for the [[8,3,2]] code. This gate was known previously [39, 40], however the fact that our procedure found a non-Clifford gate (CCZ) with no prior knowledge about the structure of the code is notable. It is often relatively straightforward to implement Clifford gates and Pauli measurements in error correcting codes. Such operations are classically simulable, but they can be promoted to universality by adding a single non-Clifford gate. However, codes with transversal non-Clifford gates are rare, and understanding the structure of such codes is an active area of research [38, 39, 45–47]. Most examples of constructions of code families with non-Clifford gates exploit some specific structure of the code family, such as tri-orthogonality [48–50]. Our procedure may be capable of finding fault-tolerant non-Clifford gates for codes whose structure is opaque to us. In particular, the structure of non-CSS codes with non-Clifford gates is poorly understood, so our procedure may be able to shed some light on this area given enhanced computational resources.

[[8,2,2]]: We successfully found non-Pauli gates for codes related to 2D color codes: the [[8,2,2]] code (2D color code defined on a projective plane [38]) and the [[6,3,2]] code (subcode

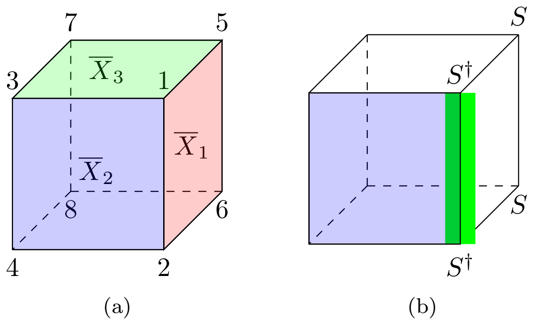


Fig. 4. CZ gates in the [[8,3,2]] code. Qubits are placed on the vertices of the cube. In (a), we highlight three logical X operators of the code, namely $\overline{X}_1 = X_1X_2X_5X_6$ (red face), $\overline{X}_2 = X_1X_2X_3X_4$ (blue face), and $\overline{X}_3 = X_1X_3X_5X_7$ (green face). Three corresponding logical Z operators are: $\overline{Z}_1 = Z_1Z_3$, $\overline{Z}_2 = Z_1Z_5$, and $\overline{Z}_3 = Z_1Z_2$. (b) Suppose that we apply S and $S^\dagger$ to the qubits on the red face. This unitary maps $\overline{X}_2$ (blue face) to $\overline{X}_2\overline{Z}_3$ (blue face and green edge). Likewise, $\overline{X}_3$ is mapped to $\overline{X}_3\overline{Z}_2$. Therefore, this unitary implements a logical $\overline{CZ}_{23}$ gate.

of a 2D color code defined on a hexagon [51]). For the [[8,2,2]] code, we found the gates that we would expect to find in a 2D color code with two logical qubits: logical CZ implemented by transversal S and $S^\dagger$, logical $H^{\otimes 2}CZH^{\otimes 2}$ implemented by transversal $\sqrt{X}^\dagger$ and logical $H^{\otimes 2}SWAP$ implemented by transversal H. And we found that the [[6,3,2]] code inherits one of the transversal gates of its parent 2D color code (the [[6,4,2]] code [51]): CZ_{12} implemented by transversal S and $S^\dagger$.

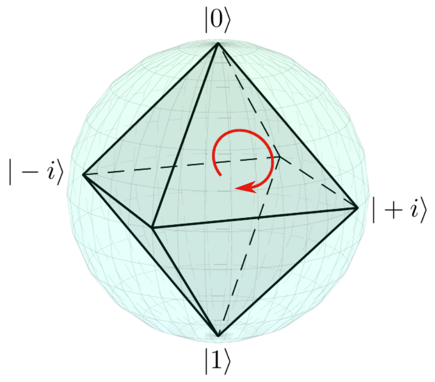


Fig. 5. An illustration of the Clifford gate $K_{1,1,1} = e^{i\pi/4}SH$. This gate maps $|0\rangle \to |+i\rangle$, $|+i\rangle \to |+\rangle$, and $|+\rangle \to |0\rangle$ (up to global phases). It can be understood geometrically by considering an octahedron embedded in the Bloch sphere. The operation performed by the gate is a clockwise rotation of $2\pi/3$ around the axis normal to the face marked with a red circle. The $|+\rangle$ state is at the front unlabeled vertex of the octahedron.

[[7,1,3]]: Our procedure also found transversal Clifford gates for the [[7,1,3]] surface code with a twist. As was noted in [37], this code has transversal implementations of the octahedral

Clifford gates, which are defined as follows:

$$K_{x,y,z} = \exp\left\{ i\frac{\pi}{3\sqrt{3}}(xX + yY + zZ) \right\}, \tag{4}$$

where $x, y, z \in \{-1, 1\}$ (See Fig. 5 for an illustration of $K_{1,1,1}$). These gates cyclically permute the Pauli operators. We can decompose these octahedral gates into products of more familiar Clifford gates as follows:

$$\begin{aligned} K_{1,1,1} &= e^{i\pi/4}SH, & K_{-1,-1,-1} &= e^{-i\pi/4}HS^\dagger, \\ K_{1,-1,1} &= e^{i\pi/4}HS, & K_{-1,1,-1} &= e^{-i\pi/4}S^\dagger H, \\ K_{1,1,-1} &= e^{i3\pi/4}XHXS^\dagger, & K_{-1,-1,1} &= e^{-i3\pi/4}SXHX, \\ K_{-1,1,1} &= e^{-i3\pi/4}XHXS, & K_{1,-1,-1} &= e^{i3\pi/4}S^\dagger XHX. \end{aligned} \tag{5}$$

The specific gates we found in our experiments were transversal realizations of $K_{-1,-1,1}$ and $K_{1,-1,-1}$.

4 Discussion and Conclusion

Here we discuss a few interesting properties of our method, and point to some potential future research directions.

4.1 Fault-tolerance of the logical gate

In general, the ansatz for logical circuit G (Fig. 1) is not guaranteed to be fault-tolerant. To bring the quantum state after the logical circuit back to code space, we apply the additional stabilizer measurement and minimal weight error correction procedure (C) in simulation. To adapt to a quantum computer, we suggest implementing a version of C using a fault-tolerant error correction protocol [52–55], applying a recovery operator conditioned on the error syndrome.

Interestingly, we find empirically that when the loss function is minimized, our procedure finds fault-tolerant logical gates G, which we verify by computing the action of G on the logical states.

Still, it is possible that for other error correction codes, after optimization one will have cases where the combined action of G and C is a logical gate, while G itself is not. In nearly all fault-tolerant logical gate constructions, error correction is applied immediately after a logical gate and so we argue that the combined action of G and (a fault-tolerant version of) C would constitute a fault-tolerant logical gate.

4.2 Resource estimate of our method

We should note that our method does not scale well to larger codes. For the calculation of the loss function in Eq. (1), we require either the full process tomography of the encoded logical space, using at least $\mathcal{O}(4^k)$ quantum circuits, or a noise-resilient way to implement a cross-device swap test between two quantum computers.

However, we still believe our method can have wide potential applications. First, many families of quantum codes have a (small) constant k (surface codes, color codes, quantum Reed-Muller codes etc. [56, 57]). In addition, even for codes that do have k growing with the number of physical qubits n, there is usually a symmetry relating the logical operators.

This means we only need to apply our method to a single representative with respect to this symmetry, effectively reducing k inversely proportional to the size of the symmetry group.

Therefore, this cost is benign when k is small and can be further mitigated by running these circuits in parallel. It is worth noting that in the classical simulations we have done, one optimization usually converged in less than 5 hours running on a commercially available GPU (Nvidia RTX 2080 Ti).

Also, due to the fact that transversal ansatz is mostly likely to be used as the ansatz for logical gates, we expect a friendly linear growth of the number of parameters which we need to optimize with respect to the number of physical qubits (n) in Eq. (1).

4.3 Potential future research

Firstly, our optimization benefits from the use of `Rotosolve` [10] optimization algorithm, which depends largely on the fact that the functional dependency of the lost function in Eq. (1) is sinusoidal due to the parametrized rotation gates we have used for the ansatz circuits (see Section 3.1 for details and numerical performance). Adapting our procedure for specific quantum computation hardware might require a different set of parametrized gates and might invalidate the use of `Rotosolve`. In this case, other optimization algorithms will be needed for finding the logical gates.

Also, it would be interesting to penalize Pauli gates or in general logical gates that have been discovered previously during the optimization of the loss function in Eq. (1). Such a penalty would violate the requirement of `Rotosolve` algorithm and is therefore not explored here. For reference, we have estimated the percentage of non-Pauli gates we would expect to observe (assuming that all gates have equal probability, see Table 1), by comparing the number of the Pauli gates generated with the total number of the logical gates generated by optimization runs found for a specific code. However, the number of experiments for each code is rather small ($\leqslant 40$) for the purpose of estimating whether there is a bias towards finding Pauli gates (Table 1).

Finally, a promising future research avenue would be to use our procedure to explore quantum codes that have not been as extensively studied as qubit stabilizer codes. Qudit quantum codes are a natural example, where a qudit is the d-dimensional analogue of a qubit. The stabilizer formalism can be extended to prime (or prime power) qudits [58], which means that it would be straightforward to generalize our procedure to these cases. Considerably less research has been done into implementing logical gates in qudit stabilizer codes compared with qubit stabilizer codes, so we may be able to find more unknown fault-tolerant gates in the qudit context. In addition, we emphasize that our procedure is not limited to stabilizer codes, and can be applied to non-stabilizer codes e.g. the codes described in Ref. [59].

In summary, we have proposed a procedure to automate the discovery of logical gates using shallow quantum circuits for a given quantum error correcting code. The ansatz for the logical gate can be tailored to a specific quantum computing architecture to take advantage of this architecture. We have shown that it can find logical gates available in the current literature for a number of error correcting codes and it additionally produces new logical gates for the [[5,1,2]] code, the [[6,3,2]] code, the [[8,3,2]] code, and the [[10,1,2]] code. Although the procedure is simulated classically, we have proposed an extension of this procedure for quantum computers, and we believe an implementation on near-term quantum computers for error correcting codes requiring a few qubits is feasible.

5 Acknowledgments

We wish to acknowledge the usage of high performance computing cluster from Department of Computer Science, University College London in completion of this project. H.C. acknowledges the support through a Teaching Fellowship from UCL. E.G. is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) [EP/P510270/1] M.V. at the Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Economic Development, Job Creation and Trade. N.P.B. is supported by the UCLQ Fellowship.

6 Author Contribution

H.C. carried out the numerical experiments. M.V. and N.P.B. provided the error correcting codes. All authors contributed extensively to the design of the procedure, and the discussions and interpretations of the results.

7 Data Availability

All data needed to evaluate the conclusions are available from the corresponding author upon reasonable request.

8 Code Availability

All code needed to evaluate the conclusions are available from the corresponding author upon reasonable request.

References

[1] P. W. Shor, *Scheme for reducing decoherence in quantum computer memory,* Phys. Rev. A **52** (Oct, 1995) R2493–R2496.

[2] A. R. Calderbank and P. W. Shor, *Good quantum error-correcting codes exist,* Phys. Rev. A **54** (Aug, 1996) 1098–1105.

[3] A. Steane, *Multiple-particle interference and quantum error correction,* Proc. R. Soc. Lond. A. **452** no. 1954, (1996) 2551–2577.

[4] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, *A variational eigenvalue solver on a photonic quantum processor,* Nature Communications **5** no. 1, (2014) 4213.

[5] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets,* Nature **549** no. 7671, (2017) 242–246.

[6] Y. Li and S. C. Benjamin, *Efficient variational quantum simulator incorporating active error minimization,* Phys. Rev. X **7** (Jun, 2017) 021050.

[7] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, *Error mitigation extends the computational reach of a noisy quantum processor,* Nature **567** no. 7749, (2019) 491–495.

[8] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, *Quantum computational chemistry,* Rev. Mod. Phys. **92** (Mar, 2020) 015003.

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, New York, NY, USA, 10th ed., 2011.

[10] M. Ostaszewski, E. Grant, and M. Benedetti, *Structure optimization for parameterized quantum circuits,* Quantum **5** (January, 2021) 391.

[11] K. M. Nakanishi, K. Fujii, and S. Todo, *Sequential minimal optimization for quantum-classical hybrid algorithms,* Phys. Rev. Research **2** (Oct, 2020) 043158.

[12] R. L. Kosut, A. Shabani, and D. A. Lidar, *Robust quantum error correction via convex optimization,* Phys. Rev. Lett. **100** (Jan, 2008) 020502.

[13] S. Taghavi, R. L. Kosut, and D. A. Lidar, *Channel-optimized quantum error correction,* IEEE Transactions on Information Theory **56** no. 3, (March, 2010) 1461–1473.

[14] R. L. Kosut and D. A. Lidar, *Quantum error correction via convex optimization,* Quantum Information Processing **8** no. 5, (Oct, 2009) 443–459.

[15] M. Berta, F. Borderi, O. Fawzi, and V. B. Scholz, *Semidefinite programming hierarchies for constrained bilinear optimization,* Mathematical Programming (Apr, 2021) . `https://doi.org/10.1007/s10107-021-01650-1`.

[16] P. D. Johnson, J. Romero, J. Olson, Y. Cao, and A. Aspuru-Guzik, *Qvector: an algorithm for device-tailored quantum error correction,* `arXiv:1711.02249 [quant-ph]`.

[17] J. Bausch and F. Leditzky, *Quantum codes from neural networks,* New Journal of Physics **22** no. 2, (Feb, 2020) 023005.

[18] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, *Reinforcement learning with neural networks for quantum feedback,* Phys. Rev. X **8** (Sep, 2018) 031084.

[19] G. Torlai and R. G. Melko, *Neural decoder for topological codes,* Phys. Rev. Lett. **119** (Jul, 2017) 030501.

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

[21] R. Sweke, M. S. Kesselring, E. P. L. van Nieuwenburg, and J. Eisert, *Reinforcement learning decoders for fault-tolerant quantum computation,* Machine Learning: Science and Technology **2** no. 2, (Jan, 2021) 025005.

[22] P. Baireuther, M. D. Caio, B. Criger, C. W. J. Beenakker, and T. E. O'Brien, *Neural network decoder for topological color codes with circuit level noise,* New Journal of Physics **21** no. 1, (Jan, 2019) 013003.

[23] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, *Machine-learning-assisted correction of correlated qubit errors in a topological code,* Quantum **2** (January, 2018) 48.

[24] K. Vogel and H. Risken, *Determination of quasiprobability distributions in terms of probability distributions for the rotated quadrature phase,* Phys. Rev. A **40** (Sep, 1989) 2847–2849.

[25] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, *Quantum fingerprinting,* Phys. Rev. Lett. **87** (Sep, 2001) 167902.

[26] D. Gottesman and I. Chuang, *Quantum digital signatures,* `arXiv:quant-ph/0105032`.

[27] M. S. Kesselring, F. Pastawski, J. Eisert, and B. J. Brown, *The boundaries and twist defects of the color code and their applications to topological quantum computation,* Quantum **2** (Oct, 2018) 101.

[28] M. Grassl, "Bounds on the minimum distance of linear codes and quantum codes." Online available at `http://www.codetables.de`, 2007. Accessed on 2019-12-09.

[29] D. Gottesman, *Class of quantum error-correcting codes saturating the quantum hamming bound,* Phys. Rev. A **54** (Sep, 1996) 1862–1868.

[30] A. M. Steane, *Simple quantum error-correcting codes,* Phys. Rev. A **54** (Dec, 1996) 4741–4751.

[31] A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane, *Quantum error correction and orthogonal geometry,* Phys. Rev. Lett. **78** (Jan, 1997) 405–408.

[32] M. Grassl, T. Beth, and T. Pellizzari, *Codes for the quantum erasure channel,* Phys. Rev. A **56** (Jul, 1997) 33–38.

[33] L. Vaidman, L. Goldenberg, and S. Wiesner, *Error prevention scheme with four particles,* Phys. Rev. A **54** (Sep, 1996) R1745–R1748.

[34] M. Vasmer and A. Kubica, *Morphing quantum codes,* `arXiv:2112.01446 [quant-ph]`.

[35] R. Laflamme, C. Miquel, J. P. Paz, and W. H. Zurek, *Perfect quantum error correcting code,* Phys. Rev. Lett. **77** (Jul, 1996) 198–201.

[36] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, *Mixed-state entanglement and quantum error correction,* Phys. Rev. A **54** (Nov, 1996) 3824–3851.

[37] T. J. Yoder and I. H. Kim, *The surface code with a twist,* Quantum **1** (April, 2017) 2.

[38] C. Vuillot and N. P. Breuckmann, *Quantum pin codes,* `arXiv:1906.11394 [quant-ph]`.

[39] A. Kubica, B. Yoshida, and F. Pastawski, *Unfolding the color code,* New Journal of Physics **17** no. 8, (Aug, 2015) 083026.

[40] E. Campbell, "The smallest interesting colour code." Online available at `https://earltcampbell.com/2016/09/26/the-smallest-interesting-colour-code/`, 2016. Accessed on 2019-12-09.

[41] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, *Open quantum assembly language,* `arXiv:1707.03429 [quant-ph]`.

[42] V. V. Shende, I. L. Markov, and S. S. Bullock, *Minimal universal two-qubit controlled-not-based circuits,* Phys. Rev. A **69** (Jun, 2004) 062321.

[43] R. Iten, O. Reardon-Smith, L. Mondada, E. Redmond, R. S. Kohli, and R. Colbeck, *Introduction to UniversalQCompiler,* `arXiv:1904.01072 [quant-ph]`.

[44] H. Chen, *Online visualization of the ansatz for three-qubit quantum gates,* 2020. `https://web.archive.org/web/20201214091330/https://quantum-circuit.com/api/embed/html/iFT77LGrkWxSKthtz`.

[45] A. Kubica and M. E. Beverland, *Universal transversal gates with color codes: A simplified approach,* Phys. Rev. A **91** (Mar, 2015) 032330.

[46] M. Vasmer and D. E. Browne, *Three-dimensional surface codes: Transversal gates and fault-tolerant architectures,* Phys. Rev. A **100** (Jul, 2019) 012312.

[47] N. Rengaswamy, R. Calderbank, M. Newman, and H. D. Pfister, *On optimality of CSS codes for transversal t,* IEEE Journal on Selected Areas in Information Theory **1** no. 2, (Aug, 2020) 499–514.

[48] S. Bravyi and J. Haah, *Magic-state distillation with low overhead,* Phys. Rev. A **86** (Nov, 2012) 052329.

[49] E. T. Campbell and M. Howard, *Unified framework for magic state distillation and multiqubit gate synthesis with reduced resource cost,* Phys. Rev. A **95** (Feb, 2017) 022316.

[50] E. T. Campbell and M. Howard, *Unifying gate synthesis and magic state distillation,* Phys. Rev. Lett. **118** (Feb, 2017) 060501.

[51] B. Criger and B. M. Terhal, *Noise thresholds for the [[4, 2, 2]] concatenated toric code,* Quantum Information and Computation **16** no. 15-16, (Apr, 2016) 1261–1281, `arXiv:1604.04062`.

[52] P. Shor, *Fault-tolerant quantum computation,* in *Proceedings of 37th Conference on Foundations of Computer Science*, pp. 56–65. 1996.

[53] A. M. Steane, *Active stabilization, quantum computation, and quantum state synthesis,* Phys. Rev. Lett. **78** (Mar, 1997) 2252–2255.

[54] E. Knill, *Quantum computing with realistically noisy devices,* Nature **434** no. 7029, (Mar., 2005) 39–44.

[55] R. Chao and B. W. Reichardt, *Quantum error correction with only two extra qubits,* Phys. Rev. Lett. **121** (Aug, 2018) 050502.

[56] E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards fault-tolerant universal quantum computation,* Nature **549** no. 7671, (Sep, 2017) 172–179.

[57] B. M. Terhal, *Quantum error correction for quantum memories,* Rev. Mod. Phys. **87** (Apr, 2015) 307–346.

[58] D. Gottesman, *Fault-tolerant quantum computation with higher-dimensional systems,* in *Quantum Computing and Quantum Communications*, C. P. Williams, ed., pp. 302–313. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[59] A. Cross, G. Smith, J. A. Smolin, and B. Zeng, *Codeword stabilized quantum codes,* IEEE Transactions on Information Theory **55** no. 1, (2009) 433–438.