

Satisfying Quantum Codes: Physics-Informed and Hardware-Aware Code Design with SAT Solvers

Ben DalFavero,^{1,2} William M. Watkins,³ Margarite L. LaBorde,² Vincent Russo,⁴ Ethan Egger,⁵ Gregory Quiroz,^{2,3} and Ryan LaRose^{1,6,7,8,*}

¹*Department of Computational Mathematics, Science, and Engineering,
Michigan State University, East Lansing, MI 48823, USA*

²*Johns Hopkins Applied Physics Laboratory, Laurel, MD 20723, USA*

³*William H. Miller III Department of Physics & Astronomy,
Johns Hopkins University, Baltimore, MD 21218, USA*

⁴*Unitary Foundation, 315 Montgomery St, Fl 10 San Francisco, California 94104, USA*

⁵*Center for Quantum Information and Control, University of New Mexico, Albuquerque, NM 87131, USA*

⁶*Department of Electrical and Computer Engineering,
Michigan State University, East Lansing, MI 48823, USA*

⁷*Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48823, USA*

⁸*Center for Quantum Computing, Science, and Engineering,
Michigan State University, East Lansing, MI 48823, USA*

Although quantum error correction is widely believed to be necessary for impactful applications of quantum computers, the design of quantum error correction codes is largely done by hand, without respect to problem or hardware constraints. In this work, we present a highly general and flexible framework for the computational design of both physics-inspired and hardware-aware quantum codes. To do so, we formulate code design as a Boolean satisfiability (SAT) problem and show how to incorporate all required error correction criteria. We prove that code design is NP-complete, ruling out any efficient algorithm for designing codes in general. Nonetheless, we show that state-of-the-art SAT solvers are able to effectively find solutions for many practical problems. Notably, we are able to design physics-inspired codes with up to 100 physical qubits in minutes, and we design new hardware-aware codes for biased noise which have a lower logical error rate than state-of-the-art surface codes.

I. INTRODUCTION

It is widely accepted that quantum error correction will be necessary to implement impactful applications on quantum computers. This need has motivated substantial effort toward designing quantum codes with lower resource overheads and greater resilience to realistic noise. Existing analytical techniques often search for quantum codes from their classical counterparts [1–3] or using inspiration from mathematics and physics [4, 5]. The impact of this search is reflected in the steady reduction of resource requirements for error-corrected implementations of Shor’s algorithm [6–14], as well as in key experimental milestones, including demonstrations of break-even and sub-threshold logical memories [15–19] and logical gate operations [20–26]. However, although new codes are continuously being developed, the design of good quantum error correcting codes has proven to be difficult. There has not been a clear improvement over the surface code in the more than three decades since it was introduced. Designing good quantum codes appears to be hard in general [27], even more so when trying to incorporate real-time error rates, qubit inhomogeneity, and other practical considerations about quantum hardware.

In this work, we develop a computational framework to design quantum codes that is capable of overcoming these

challenges. Specifically, we formulate quantum code design as a satisfiability (SAT) problem and leverage state-of-the-art SAT solvers to design novel quantum codes. Our SAT formulation is extremely general and is able to incorporate problem symmetries, which we refer to as physics-inspired code design, as well as detailed noise models, which we refer to as hardware-aware code design. Through this framework, we design quantum codes which demonstrate superior accuracy when computing expectation values in quantum error detection experiments, and quantum codes that have lower logical error rate than the XZZX surface code for biased noise models. We demonstrate the practical scalability of this approach by designing novel codes with over 100 physical qubits in minutes to hours, depending on the application, on a laptop.

In addition to these findings, we study the complexity of designing codes in general and prove that this is an NP-complete problem by a reduction from the set cover problem. Beyond this worst-case complexity, we analyze the average-case runtime for relevant problem instances and characterize a phase transition in the probability of finding a satisfiable solution, as well as the runtime, for practical codes. In short, we present both a general theoretical framework for quantum code design with rigorous complexity results and average-case performance, in addition to practical tools for quantum code design used to find novel codes with superior performance.

While there is prior work in the computational design of quantum codes, our work significantly extends

* Corresponding author: rmlarose@msu.edu

current state-of-the-art methods in several ways. Notably, our framework is, as mentioned, extremely general and capable of designing new codes from existing problem symmetries and/or hardware-tailored noise models, in addition to designing new codes without any additional constraints. We also exceed the scale of state-of-the-art methods. These methods include Ref. [28], in which reinforcement learning was used to discover codes with constrained connectivity on up to 25 physical qubits, and Ref. [29], in which a multi-agent system was used to optimize quantum codes and enable discovery of a $[[100, 50, 4]]$ code in under an hour. Using our SAT formulation, we can design a $[[100, 50, 4]]$ code in under five minutes of CPU time, in addition to designing novel physics-inspired codes which start from the symmetries of a 100 qubit Fermi-Hubbard model in minutes to hours, depending on the application. Recently, SAT has been used in Ref. [30] to discover so-called phantom codes. Our work enables code discovery as a special case, but we also enable code extension, physics-inspired code design, and hardware-aware code design in a highly general, flexible, and performant framework. Also, SAT has been used in Ref. [31] to find state preparation circuits for CSS codes, and in Ref. [32] to certify distances in generated quantum codes. Our work again extends this by incorporating physical symmetries, hardware information, and error correction criteria to enable end-to-end quantum code design, and we demonstrate the advantages of this through benchmarks in higher fidelity error-detected quantum algorithms and lower logical error rates in error-corrected memory experiments.

The structure of the paper is as follows. In Sec. II, we summarize the main results of our work. In Sec. III, we provide background on quantum error correction codes, satisfiability, and our notation. In Sec. IV, we define the problem of quantum code design and discuss particular cases which we focus on — code discovery and code extension. In Sec. V, we prove that the problem of code design is NP-complete using a reduction from the set cover problem. In Sec. VI, we formulate the quantum code design problem in terms of Boolean satisfiability (SAT), showing how to represent error correction criteria in SAT clauses. In Sec. VII, we analyze the SAT-UNSAT transition in the average-case complexity of random instances of the code design problem. In Sec. VIII, we demonstrate our formalism by designing physics-inspired codes, in particular novel codes stemming from the Fermi-Hubbard model with up to 100 physical qubits. In Sec. IX, we demonstrate our formalism by designing hardware-aware codes, notably designing novel surface codes which outperform the XZZX surface code for biased noise.

II. MAIN RESULTS

In this section we provide a summary of our main results of this work. An overview of our code design framework is shown in Fig. 1. We formally define the problem

of quantum code design in Def. 1 and provide existence proofs for important cases in Sec. IV. These cases are *code discovery*, in which we start from an empty initial stabilizer group, and *code extension*, in which we start from a non-empty initial stabilizer group. Code discovery is concerned with the generation of new quantum codes, while code extension is concerned with building a code from an existing set of symmetries, either from the physics of a particular problem or from an initial code. We refer to the former case of starting from a set of problem symmetries as *physics-inspired code design*. Both code discovery and code extension can incorporate physical noise models of devices, and we refer to this case as *hardware-aware code design*. Building on the notion of classical code extension, we define extensions of quantum codes in Def. 2 and prove that distance-increasing extensions exist for quantum codes, stated in Theorem 2. We emphasize that a distance-increasing extension C' of a code C satisfies $C \subseteq C'$ in the sense that C' adds additional qubits and stabilizers to the initial set of stabilizers to increase the distance. We discuss particular examples including extensions with mutations (i.e., in which some initial stabilizers are changed) and extensions without mutations (i.e., in which stabilizers are unchanged). Interestingly, both examples exist for quantum low density parity check (QLDPC) codes. Our work provides a general framework to produce code extensions beyond these few examples.

Next, we analyze the general complexity of the quantum code design problem, and prove the following:

Theorem 1 (Informal). *The problem of designing a quantum code is NP-complete.*

The formal version and proof of this statement is given in Theorem 4. Our proof uses a reduction from k -SET COVER, one of Karp’s original NP-complete problems [33]. This rules out an efficient solution for designing quantum codes in the general case, and provides a theoretical foundation to prior work in computational code design. Nonetheless, in the same way that decoding is NP-hard (in fact, $\#P$ -complete) [34] yet many algorithms have been found for practical cases, we next explore algorithms for solving the quantum code design problem in practically-relevant cases. To do so, we formulate the quantum code design problem as a SAT instance in Sec. VI. This works by writing conditions on stabilizer codes including commutativity and error correction/detection of desired errors as Boolean functions, then creating an overall Boolean formula in conjunctive normal form. This formulation is very general and encompasses code discovery, physics-informed code design, and hardware-aware code design.

The SAT formulation allows us to apply state-of-the-art SAT solvers [35, 36] for code design and also to analyze the satisfiability probability and the practical runtime of this approach. We do this in Sec. VII in which we empirically find a phase transition in the satisfiability probability, and in average-case time complexity, for in-

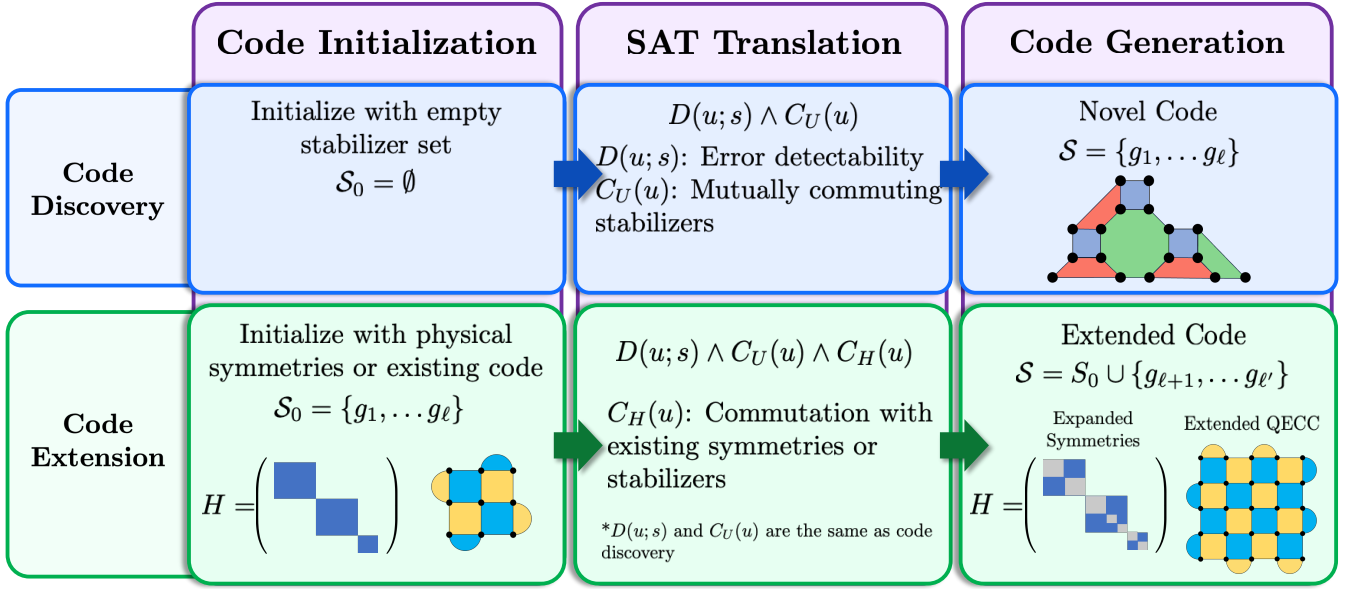


Figure 1. Overview of our quantum code design framework. Our framework enables two categories of code design: code discovery and code extension. In the former, the code is initialized with an empty stabilizer set, while in the latter the initial set $\mathcal{S}_0$ is determined by a predetermined code or a set of physics-informed symmetries captured within the Hamiltonian H . Our SAT-based framework enables translation between both code design scenarios through the symplectic form, which encodes information about both the error set $\mathcal{E}$ and stabilizers/symmetries. Solutions to the SAT problem lead to novel codes “from scratch” or extended codes that contain the initial stabilizers. Both code discovery and code design incorporate a set of errors $\mathcal{E}$ to be detectable/correctable, enabling hardware-aware code design.

stances with randomly-drawn sets of errors. This phase transition is illustrated in Fig. 3 where the order parameter is the number of errors per stabilizer. Additionally, we compute a phase diagram for distance-three codes, shown in Fig. 4. Here, while the Gilbert-Varshamov and Singleton bounds constrain the low- and high-rate regions, respectively, the fitted SAT-UNSAT transition provides an empirical prediction of code existence in the contested region. Generally, the SAT formulation provides a rigorous and powerful framework to prove the (un)satisfiability of a given set of code parameters.

We then apply our code design formalism to creating novel physics-inspired codes in Sec. VIII. Here, we demonstrate our framework by starting from the symmetries of the Fermi-Hubbard Hamiltonian and augmenting them to correct certain errors. We demonstrate that this improves the accuracy of expectation values computed in noisy simulation in Fig. 5. Next, we show that we scale this approach and find physics-informed code extensions for an $n = 100$ Fermi-Hubbard Hamiltonian in minutes to hours, shown in Fig. 6. This significantly extends the problem-size of prior work in state-of-the-art computational code design mentioned in Sec. I.

Last, we apply our code design formalism to creating novel hardware-aware codes in Sec. IX. We first demonstrate this by augmenting the physics-informed code design problem for the Fermi-Hubbard model with biased noise. In this case, our SAT formulation finds novel codes that are tailored to the hardware noise model, and we

show these lead to reduced undetected error rates in noisy simulation, shown in Fig. 7. Next, we demonstrate our framework by discovering novel surface codes for biased noise models which have a lower logical error rate than current state of the art, under a maximum likelihood decoder. Specifically, we design novel surface codes using our formalism which have logical error rates lower than the XZZX surface code, which is tailored for biased noise. These results, shown in Fig. 8, further demonstrate the utility of our code design framework as a tool capable of accelerating the timeline to an experimental demonstration of a large-scale error corrected quantum computer.

III. PRELIMINARIES

We begin by reviewing foundational concepts that are relevant to the development and discussion presented in this work. In particular, we provide an overview of stabilizer codes and describe their representation in binary symplectic form, which will be used throughout our analysis. We also introduce the notation and basic definitions associated with SAT problems that are needed to formulate the code-design problems considered in subsequent sections.

A. Stabilizer Codes

A quantum stabilizer code is defined by a stabilizer group

$$\mathcal{S} = \langle g_1, \dots, g_{n-k} \rangle \quad (1)$$

where operators g_i are the stabilizer generators of the code. Throughout the paper we assume all codes are stabilizer codes unless otherwise noted. The triple $\llbracket n, k, d \rrbracket$ refers to a quantum code that encodes k logical qubits into n physical qubits and has distance d . When the distance of a code is not known or is unimportant, the double $\llbracket n, k \rrbracket$ is used to specify the code. The same three parameters for a classical code are presented as $[n, k, d]$. It is desirable for codes to have both a high encoding rate k/n and high distance d . Intuitively, the encoding rate constitutes the number of logical qubits encoded per physical qubit, and the distance defines how many errors it takes to map one codeword onto another. A distance d code can correct arbitrary single-qubit errors on up to $t = \lfloor (d-1)/2 \rfloor$ physical qubits.

The binary symplectic representation maps the n -qubit Pauli group $\mathcal{P}_n$ to the binary symplectic space $\mathbb{F}_2^{2n}$, *i.e.*, the $2n$ -dimensional vector space over the field of integers modulo 2. In this representation, a Pauli string $P \in \mathcal{P}_n$ is represented as the binary vector $P \cong u \equiv [\vec{x}|\vec{z}]$ with $\vec{x}, \vec{z} \in \mathbb{F}_2^n$, where

$$\begin{aligned} x_i &= 0, \quad z_i = 0, & \text{if } P_i &= I, \\ x_i &= 1, \quad z_i = 0, & \text{if } P_i &= X, \\ x_i &= 1, \quad z_i = 1, & \text{if } P_i &= Y, \\ x_i &= 0, \quad z_i = 1, & \text{if } P_i &= Z. \end{aligned} \quad (2)$$

For $P_1, P_2 \in \mathcal{P}_n$, let $P_1 \cong u_1$ and $P_2 \cong u_2$. The product of two Pauli strings is the bitwise exclusive-OR (XOR) of their binary symplectic representations:

$$P_1 P_2 \cong u_1 \oplus u_2. \quad (3)$$

Whether P_1 and P_2 commute can be tested using a bilinear form. Let

$$\Lambda = \begin{bmatrix} \mathbb{0}_{n \times n} & \mathbb{I}_{n \times n} \\ \mathbb{I}_{n \times n} & \mathbb{0}_{n \times n} \end{bmatrix} \quad (4)$$

where $\mathbb{I}_{n \times n}$ is the $n \times n$ identity matrix and $\mathbb{0}_{n \times n}$ is the $n \times n$ matrix of all zeros. Using this matrix, we define the symplectic inner product

$$\langle u_1, u_2 \rangle := u_1^T \Lambda u_2. \quad (5)$$

The symplectic inner product serves as a commutation test since

$$\langle u_1, u_2 \rangle = \begin{cases} 0 & \text{if } [P_1, P_2] = 0, \\ 1 & \text{if } \{P_1, P_2\} = 0. \end{cases} \quad (6)$$

The check matrix of a stabilizer code is the matrix that has rows made up of the binary symplectic representations of the code's stabilizer generators, *i.e.*, the check

matrix H of the stabilizer code given in (1) is

$$H = \begin{bmatrix} - & u_1 & - \\ & \vdots & \\ - & u_{n-k} & - \end{bmatrix}, \quad (7)$$

where $g_i \cong u_i$ and the code has stabilizer generators $g_1, \dots, g_{n-k}$.

B. Boolean Satisfiability

SAT problems are defined in terms of N Boolean variables and a set of M constraints where each constraint takes the special form of a clause. In the SAT formulation of code design (Sec. VI), we use the symbols $\wedge$, $\vee$, and $\oplus$ to denote logical conjunction, logical disjunction, and exclusive-OR, respectively. SAT instances are commonly expressed in conjunctive normal form (CNF), in which a formula over Boolean variables $x_1, \dots, x_N$ is written as a conjunction of clauses. Each clause is a disjunction of literals, where a literal is either a variable or its negation. For example, the formula

$$(x_1 \vee x_5 \vee \neg x_9) \wedge (\neg x_2 \vee x_{12}) \wedge (x_2) \quad (8)$$

is in CNF. Moreover, the conjunction of two or more CNF formulas also yields a CNF formula.

IV. QUANTUM CODE DESIGN

Designing a quantum code entails building a stabilizer group which satisfies a certain set of criteria. Nominally, the main criterion is the set of errors to correct, but additional criteria can include restrictions on minimum or maximum stabilizer weight, (geometric) locality, code distance, and similar properties. Formally, we define the quantum code design problem as follows.

Definition 1 (k -CODE DESIGN). *Given a set of n -qubit Pauli operators $\Sigma \subseteq \mathcal{P}_n$, a set of n -qubit Pauli errors $\mathcal{E}$, and an integer k , find a subset of at most $n - k$ linearly independent operators from Σ that generates the stabilizer group of an $\llbracket n, k \rrbracket$ stabilizer code detecting all errors in $\mathcal{E}$, that is, every $E \in \mathcal{E}$ either has a non-zero syndrome or belongs to the stabilizer group.*

We distinguish between two cases of code design — code discovery, in which the initial set of stabilizer generators is empty, and code extension, in which the initial set of stabilizer generators is non-empty. Both paradigms are addressed by our code design framework (Sec. VI), and notably both paradigms can be made hardware-aware by incorporating error characterizations. Code extension can generate physics-informed codes by populating the initial set of stabilizer generators with problem symmetries. Framed purely in terms of error correction, code extension can generate new codes from old codes that can detect more errors.

A. Discovery of Quantum Codes

The problem of code discovery is to find some set of stabilizer generators constituting a quantum code encoding k logical qubits in n physical qubits that is able to detect or correct some desired set of Pauli errors. Let

$$\mathcal{E} = \{E_1, \dots, E_s\} \quad (9)$$

be the set of (Pauli) errors we would like to be correctable. Let C be the proposed code encoding k logical qubits using n physical qubits with stabilizer group $\mathcal{S}$ as in Eq. (1). Then C detects the errors in $\mathcal{E}$ if, for any $E \in \mathcal{E}$, either E anticommutes with at least one stabilizer generator g_i and is thus detectable, or $E \in \mathcal{S}$ such that it does not corrupt the encoded information (i.e., E is a logical identity). For C to be an error-correcting code for the set of errors $\mathcal{E}$, the stabilizer group must fulfill the Knill-Laflamme conditions [37, 38]; for all pairs of errors $E_i, E_j \in \mathcal{E}$, it must be true that

$$E_i E_j^\dagger \notin \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}, \quad (10)$$

where $\mathcal{N}(\mathcal{S})$ is the normalizer of $\mathcal{S}$. Generating the normalizer $\mathcal{N}(\mathcal{S})$ can be done in polynomial time with Gaussian elimination — we give an explicit algorithm for this in Appendix A.

B. Extensions of Quantum Codes

Instead of discovering a new code every time one wishes to correct more errors in encoded information, it is appealing to modify an existing code in such a way that the new code has a higher distance. In classical error correction, one such modification is code *extension*, where m bits are added to the end of the codewords of an $[n, k, d]$ classical error-correcting code [39–41]. If the new bits are chosen judiciously, the extended code can correct more errors. Previous work on extending quantum codes used techniques from machine learning [28, 42] and concatenated codes [43, 44], but these works do not employ the generalized formalism presented here.

We begin by reviewing the concept of classical code extensions as introduced in Ref. [41]. Let C be an $[n, k, d]_q$ classical code over $\mathbb{F}_q$ with minimum distance d . Let $G \in \mathbb{F}_q^{n \times k}$ be a generator matrix for C of full rank. (Note that we set codewords to be columns of the generator matrix whereas Ref. [41] sets codewords to be rows.) By $\mathcal{S}_d = \{c \in C : |c| = d\}$ we denote the set of all codewords of Hamming weight d and by $\mathcal{I}_d = \{v \in \mathbb{F}_q^k : |Gv| = d\}$ we denote the set of corresponding information vectors, i.e., the unencoded vectors $v \in \mathbb{F}_q^k$ that generate weight- d codewords. Then, Ref. [41] shows that the code C can be extended to a code C' with parameters $[n + m, k, d + 1]_q$ if and only if there exists a matrix $X \in \mathbb{F}_q^{m \times k}$ such that

$$\sum_{i=1}^k X_i v_i \neq 0, \quad \forall v \in \mathcal{I}_d, \quad (11)$$

where X_i denotes the i^{th} column of X . The extended code C' has generator matrix

$$G' = \begin{bmatrix} G \\ X \end{bmatrix}. \quad (12)$$

Here, we generalize this notion to quantum code extension. For this, it is helpful to rewrite Eq. (1) in terms of the check matrix.

Lemma 1 (Check Matrix for Classical Code Extension). *Let C be an $[n, k, d]$ classical code with check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$. Let the code $C' = [n + m, k, d + 1]$ be an extension of C . Then the check matrix H' of the extended code C' can be expressed in the form*

$$H' = \begin{bmatrix} H & \mathbf{0}_{(n-k) \times m} \\ U_1 & U_2 \end{bmatrix} \quad (13)$$

where $U_1 \in \mathbb{F}_2^{m \times n}$ and $U_2 \in \mathbb{F}_2^{m \times m}$ together form the m extra checks $U = [U_1 \mid U_2]$. The original checks are unchanged, with zeros on the m new bits.

Proof. Let G' be the generator matrix of the extended code as in Eq. (12). To form a valid check matrix for C' , the rows of the check matrix H' must be orthogonal to the columns of G' . Let h_i be the i^{th} row of H , and construct $h'_i = [h_i \mid \mathbf{0}_{1 \times m}]$ from h_i . Automatically, h_i is orthogonal to any column g_j of G . For any column g_j of G , the corresponding column of G' has the form $g'_j = [g_j \mid x_j]$, where x_j is the j^{th} column of X . The inner product of h'_i and g'_j is

$$h'_i \cdot g'_j = \bigoplus_{l=1}^{n+m} h'_{i[l]} g'_{j[l]} = (h_i \cdot g_j) \oplus (\mathbf{0}_{1 \times m} \cdot x_j) = 0. \quad (14)$$

In this way, $n - k$ vectors orthogonal to the columns of G' have been constructed. The new check matrix H' has $n + m - k$ rows. To complete the check matrix, m more vectors orthogonal to the columns of G' are found and appended to H' . Let these new rows form the matrix U . Then the matrix H' defined in Eq. (13) has $n + m - k$ rows that are all orthogonal to the columns of G' , and thus is a valid check matrix for the code C' . $\square$

This allows us to generalize code extension to quantum codes.

Definition 2 (Quantum Code Extension). *Let C be a quantum code with parameters $[[n, k, d]]$ and a parity check matrix $H = [H_X \mid H_Z]$. A code C' with parameters $[[n + m, k', d']]$ is an extension of C if and only if it admits a parity check matrix of the form*

$$H' = \left[\begin{array}{cc|cc} H_X & W_X & H_Z & W_Z \\ -U_X & - & -U_Z & - \end{array} \right]. \quad (15)$$

where $W_X, W_Z \in \mathbb{F}_2^{(n-k) \times m}$ and $U_X, U_Z \in \mathbb{F}_2^{m \times (n+m)}$ are matrices that represent new stabilizers added to the code C to form C' .

Note that while the classical code extension Eq. (13) leads to the original checks being 0 on the new bits added during the extension, we do not make such a restriction for quantum code extensions. Instead, we simply require that the upper-left block of the check matrix remains unchanged in the new code and keep $W_X, W_Z \in \mathbb{F}_2^{(n-k) \times m}$ arbitrary. In the case that $W_X, W_Z = \mathbf{0}_{(n-k) \times m}$ we refer to the extension as an *extension without mutation*, and in the case that $W_X, W_Z \neq \mathbf{0}_{(n-k) \times m}$ we refer to the extension as an *extension with mutation*.

As an example, consider a three-qubit repetition code with stabilizer generators $g_1 = ZZI$ and $g_2 = IZZ$. The code defined by these stabilizer generators is not able to detect any phase flip errors, since they commute with the stabilizer generators. In order to detect these errors, an extension can be employed that adds one qubit (*i.e.* $m = 1$) and one stabilizer generator $g_3 = \bigotimes_{i=1}^4 \sigma_i$, where σ_i is a single-qubit Pauli operator. To detect weight-one phase flip errors on the original three qubits of the code, g_3 must be made up of Pauli- X operators. In order to keep the stabilizer group abelian, choose $\sigma_1 = \sigma_2 = \sigma_3 = X$. Because the code has three stabilizer generators and encodes one logical qubit, we must have $n = 4$ for the extended code. At least one stabilizer must have non-trivial support on the fourth physical qubit, and the original stabilizer generators from the repetition code do not have this feature, so it is required that $\sigma_4 \neq I$. For simplicity, choose $\sigma_4 = X$. The new stabilizer $g_3 = X^{\otimes 4}$ enables the detection of weight-one phase flip errors on each of the qubits, and the code with stabilizer group $\langle g_1, g_2, g_3 \rangle$ can still detect the bit flip errors on the first three qubits.

Since the purpose of extending a classical code is to increase the number of errors that it can correct, it is natural to ask this question for quantum code extensions. This is not immediately obvious because, although the new checks of the extended code are designed to detect higher-weight errors, in order to prevent the distance from decreasing, the new checks must also be able to detect errors on the new qubits at the same time. We show below that this is indeed possible.

Theorem 2 (Existence of Distance-Increasing Code Extensions). *Given an $[[n, k, d]]$ code C with $d > 1$, there exists an extension C' of C with parameters $[[n + m, k', d']]$ where $1 \leq m \leq n(n - 1)$ with $d' > d$ and $k' \geq k$.*

Proof. Let C' be the code that consists of C concatenated with itself. It is known [45, Chp. 3.5] that C' is an $[[n^2, k^2, d^2]]$ code. Furthermore, all stabilizers of C are also stabilizers of C' . (Each inner block of C' is a copy of C .) Since $d > 1$, it follows that $d' = d^2 > d$, and similarly $k' = k^2 \geq k$. So, letting $m = n^2 - n = n(n - 1)$, we see that C' is an extension of C with parameters $[[n + m, k', d']]$ with $d' > d$ and $k' \geq k$. $\square$

Our proof here relies on code concatenation, which can be both resource intensive (in the number of logical qubits) as well as difficult to decode relative to

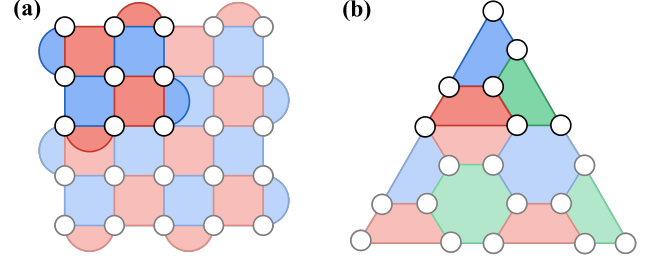


Figure 2. Illustration of distance-increasing extensions of quantum codes with mutation, shown here for two QLDPC codes. (a) A distance $d = 3$ rotated surface code overlaid on a $d = 5$ distance surface code. (b) A distance $d = 3$ triangular color code (the Steane code) overlaid on a distance $d = 5$ triangular color code. Note that, in both cases, there are checks along the boundary of the smaller code that must be altered. Thus, these are examples of distance-increasing extensions with mutation.

Calderbank-Shor-Steane (CSS) codes. A lower overhead route is through quantum low density parity check (QLDPC) codes such as surface codes [17], color codes [46], and bivariate bicycle (BB) codes [47]. These code families are defined on a lattice, and increasing the lattice size (instead of concatenation) is used to increase their distance. For example, Fig. 2(a) shows rotated surface codes with $d = 3$ and $d = 5$. The nine physical qubits of the $d = 3$ code are associated with those in the upper-left corner of the larger $d = 5$ code. Every four-body check of the $d = 3$ code is also a four-body check of the $d = 5$ code. The two-body checks along the top and left edges of the $d = 3$ code's lattice are present in the $d = 5$ code as well. However, the two-body checks along the right and bottom edges correspond to four-body checks in the $d = 5$ code. We say that the $d = 5$ code is an extension *with mutation* of the $d = 3$ code. The same is true in Fig. 2(b) for triangular color codes with distance $d = 3$ (the Steane code) and $d = 5$. Associating the qubits of the $d = 3$ code with the upper part of the triangle of the $d = 5$ code, these two codes are also related by an extension. Mutation is again necessary, since a trapezoidal face in the $d = 3$ code must be turned into a hexagon in the $d = 5$ code. Interestingly, bivariate bicycle codes are a class of QLDPC codes which admit extension without mutation.

Theorem 3. *Let C be the bivariate bicycle code defined by the quadruple (ℓ, m, A, B) . Let $A'(x, y) = A(x, y^t)$ and $B'(x, y) = B(x, y^t)$ for some integer $t > 1$. The bivariate bicycle code C' defined by (ℓ, tm, A', B') is an extension of C without mutation up to a permutation of the physical qubits in C' .*

A proof of Theorem 3, including background on bivariate bicycle codes, is given in Appendix C.

Beyond these examples, there is no general framework for producing extensions of quantum codes (up to the discussion of prior work in Sec. I, but these papers deal

with code discovery and not code extension). In this paper, we provide such a framework, and discover interesting classes of quantum codes that are both physics-informed and hardware-aware, having certain advantages with respect to state-of-the-art quantum codes. As our framework is computational, it is crucial to study the complexity of the code design problem, which is done in the following section.

V. COMPLEXITY OF QUANTUM CODE DESIGN

Previously-studied approaches to designing quantum codes often employ heuristic search through the set of possible check matrices [3, 48] or the solution to difficult optimization problems such as game-theoretic methods [29], maximum clique cover [49], or the use of reinforcement learning to optimize a cost function [28]. In Ref. [31], it is shown that the problem of designing a quantum code bears a strong resemblance to a set cover problem, an NP-complete problem. It has been shown that finding the minimum distance of a code (though not necessarily designing a code with a certain distance) was previously shown to be NP-Hard [27]. In the realm of classical error correction, SAT has been proposed as a design strategy [50], and solutions to systems of Boolean polynomial equations have been used to compute extension of classical codes [41]. However, to our knowledge, no work to date has rigorously analyzed the computational complexity of *designing* quantum codes. While mappings *from* code design to an NP-complete problem are widely used, a mapping from an NP-complete problem *to* a code design problem (and thus a proof of complexity) has not been demonstrated in prior work. Such a proof is necessary to rule out the existence of more efficient algorithms. In this section, we prove that the code design problem is NP-complete.

Before analyzing the time complexity of the problem, we outline the memory resources required to find a solution. When discovering a quantum code, every element of the check matrix $H \in \mathbb{F}_2^{(n-k) \times 2n}$ is unknown, defining a $2n(n-k)$ -dimensional search space. The matrix H is constrained by the facts that the resulting stabilizer group must commute and that some desired set of errors must be detectable (or, equivalently, correctable). The problem of code extension outlined in Sec. IV B is solved by finding values of the free variables U and W given some initial stabilizers and a set of errors. The free variables must be chosen such that the whole stabilizer group is abelian and all desired errors, both those that were previously detectable and those that were undetectable, can be detected by the new code. Given that $U_X, U_Y \in \mathbb{F}_2^{m \times n+m}$ and $W_X, W_Y \in \mathbb{F}_2^{(n-k) \times m}$, the number of variables in the problem scales as $O((n-k)m + nm^2)$. The source of the worst spatial scaling is storing the Pauli errors that the code must correct. If one desires to discover a distance d code, then all errors of weight less than d must

be enumerated. The total number of errors is

$$M = \sum_{w=1}^{d-1} 3^w \binom{n+m}{w}, \quad (16)$$

where $\binom{n+m}{w}$ counts the number of ways to choose the w qubits comprising the support of a weight- w Pauli error. For each such support, each of the w qubits can independently be assigned to one of the three non-identity Pauli operators, resulting in 3^w distinct Pauli errors.

Discovering a high-distance code or extending a code to detect a large set of high-weight errors is thus memory-intensive. These memory scaling concerns are mitigated if we restrict to discovering low-distance codes or only correcting a few additional errors in a code extension problem.

In order to prove that the code design problem itself is NP-complete, we define the following decision problem version of k -CODE DESIGN.

Definition 3 (k -CODE DESIGN (Decision problem)). *Given a set of n -qubit Pauli operators $\Sigma \subseteq \mathcal{P}_n$, a set of n -qubit Pauli errors $\mathcal{E}$, and an integer k , decide whether there exists a subset of at most $n-k$ linearly independent operators from Σ that generates the stabilizer group of an $[[n, k]]$ stabilizer code detecting all errors in $\mathcal{E}$, that is, every $E \in \mathcal{E}$ either has a non-zero syndrome or belongs to the stabilizer group.*

To prove the problem is NP-complete, we utilize a reduction from k -SET COVER, which was shown to be NP-complete by Karp [33].

Definition 4 (k -SET COVER). *Given a finite universe of values $U = \{1, \dots, r\}$, a set of n finite subsets of U , $\{S_j\}_{j=1}^n$, and an integer k , the problem k -SET COVER asks to decide if there is a smaller set of at most k subsets $S_{i_1}, \dots, S_{i_k}$ s.t. $\bigcup_{i=1}^k S_{i_i} = U$.*

Using this, we can show the following:

Theorem 4. k -CODE DESIGN is NP-complete.

Proof. To prove that the problem k -CODE DESIGN is NP-complete, we will first prove that it is NP-hard by reducing k -SET COVER to k -CODE DESIGN, then we prove membership in NP.

First, we reduce an instance of k -SET COVER to an instance of k -CODE DESIGN. Suppose we are given a set of subsets $\{S_j\}_{j=1}^n$ drawn from a universe $U = \{1, \dots, r\}$ and an integer k ; we will show that this comprises an instance of k -CODE DESIGN. We construct candidate stabilizers $\Sigma = \{\hat{S}_1, \dots, \hat{S}_n\}$ and errors $E_1, \dots, E_r$, together with some number of logical qubits $\bar{k}$ and physical qubits $\bar{n}$ to specify an instance of k -CODE DESIGN whose solution encodes a solution to our instance of k -SET COVER.

Let a Pauli error E_i be assigned to each element u_i of the universe U . Let $\mathcal{E} = \{E_1, \dots, E_r\}$, and define $E_i = X_i$. Each stabilizer $\hat{S}_j$ is constructed from a subset

S_j s.t. $\hat{S}_j$ anticommutes with the error E_i iff $i \in S_j$. One valid choice is $\hat{S}_j = \bigotimes_{i=1}^{\bar{n}} \hat{S}_{j[i]}$, where

$$\hat{S}_{j[i]} = \begin{cases} I & i \notin S_j, \\ Z & \text{else.} \end{cases} \quad (17)$$

For all qubits with index $i > r$, the value does not affect the ability to detect the errors E_i , so the values on those qubits will be determined later. Since all stabilizers defined this way are in $\{I, Z\}^{\otimes \bar{n}}$, they all commute. Since all errors are in $\{I, X\}^{\otimes \bar{n}}$, the errors cannot be in any subgroup generated by some subset of Σ . To find a code that encodes $\bar{k}$ qubits and detects all errors in $\mathcal{E}$, it suffices to choose generators $\{\hat{S}_{i_1}, \dots, \hat{S}_{i_{\bar{n}-\bar{k}}}\} \subset \Sigma$ s.t. for each $E_i \in \mathcal{E}$, there is at least one generator $\hat{S}_{i_l}$ in the subset that anticommutes with E_i .

Any subset of l of our candidate stabilizers Σ should be independent so that they can generate a group of size 2^l . If the whole set Σ is independent, then any subset of Σ will also be independent. Taking a value of $\bar{n} > r$ enables this requirement to be satisfied. Represent each stabilizer generator $\hat{S}_j$ by a binary vector $b_j \in \mathbb{F}_2^{\bar{n}}$. For now, $b_{j[i]} = 0$ for $i > r$. Suppose there is some algorithm that could find the linearly independent subset of these binary vectors in polynomial time (analogous to Gram-Schmidt). Then there are potentially some vectors b_j that are not in the independent set. We can change values on the bits $r+1$ through $\bar{n}$ to make them linearly independent. These changes do not affect the ability to detect errors on the first r qubits. Some value of $\bar{n}$ will be needed for this procedure; indeed, only one extra bit is needed. If some vector b_l is in the linearly independent set, set $b_{l[r+1]} = 0$, and $b_{l[r+1]} = 1$ otherwise.

Finally, we must know how many logical qubits $\bar{k}$ the code must encode. Since we desire k stabilizers in a valid solution to k -CODE DESIGN, $k = \bar{n} - \bar{k}$, which implies that $\bar{k} = \bar{n} - k$. The errors E_i can be constructed from the universe U in their binary symplectic forms in $O(r\bar{n})$ time. The stabilizers $\hat{S}_j$ can be constructed from the subsets S_j in $O(n\bar{n})$ time.

Suppose there is an error-detecting code that solves the constructed instance of k -CODE DESIGN. This code is mapped onto the solution to the original instance of k -SET COVER in the following way: For each stabilizer $\hat{S}_{i_l}$ in the code, we enumerate the errors $E_{j_1}, \dots, E_{j_q}$ that $\hat{S}_{i_l}$ anticommutes with. Equivalently, enumerate all qubits with index $l \leq r$ s.t. $\hat{S}_{j[l]} = Z$. The detectable errors for stabilizer $\hat{S}_j$ correspond to a subset $S_j = \{j_1, \dots, j_q\}$. Because the code detects all errors, there is at least one stabilizer that anticommutes with any given E_i , and thus there is at least one corresponding subset that contains the value i . The set of the subsets corresponding to the chosen stabilizer generators is a set covering and has size k . The existence of an error detecting code for the constructed instance of k -CODE DESIGN implies the existence of a set covering for the original instance of k -SET COVER.

Suppose, however, that no code to detect errors $\mathcal{E}$ and

encode $\bar{k}$ qubits can be made from the candidate generators Σ . For any subset σ of Σ with size k , there is at least one error $E_i \in \mathcal{E}$ that commutes with all operators in σ . Let $\sigma = \{\hat{S}_{j_1}, \dots, \hat{S}_{j_k}\}$. By construction, $i \notin S_{j_l}$ for all $l = 1, \dots, k$. Since i is not in the set of subsets corresponding to any set of stabilizers σ , a set covering does not exist. Thus any instance of k -SET COVER can be reduced to an instance of k -CODE DESIGN in polynomial time, meaning k -CODE DESIGN is NP-hard.

Finally, we show that any solution to k -CODE DESIGN can be checked in polynomial time. Let $\mathcal{S} = \langle \hat{S}_{j_1}, \dots, \hat{S}_{j_{\bar{n}-\bar{k}}} \rangle$ be a proposed stabilizer group for the code. For any error $E \in \mathcal{E}$, either there must be a stabilizer $\hat{S}_{j_l}$ that anticommutes with E or $E \in \mathcal{S}$. To check for anticommutation, if $|\mathcal{E}| = r$, then $(\bar{n} - \bar{k})r$ anticommutation relations must be evaluated. Using the binary symplectic form of the Pauli strings, these checks can be done in $O(\bar{n})$ time. Checking whether or not $E \in \mathcal{S}$ can also be done in the binary symplectic form. If $E \in \mathcal{S}$, then $\exists d \in \mathbb{F}_2^{\bar{n}-\bar{k}}$ s.t. $E = \hat{S}_{j_1}^{d_1} \dots \hat{S}_{j_{\bar{n}-\bar{k}}}^{d_{\bar{n}-\bar{k}}}$. Let E have binary symplectic form $\tilde{E}$ and $\hat{S}_{j_l}$ have a binary symplectic form $\tilde{S}_{j_l}$. The equivalent group membership condition is then

$$\exists d \in \mathbb{F}_2^{\bar{n}-\bar{k}} \text{ s.t. } \tilde{E} = \bigoplus_{l=1}^{\bar{n}-\bar{k}} d_l \tilde{S}_{j_l} \quad (18)$$

by the fact that multiplication in the Pauli subgroup is equivalent to bitwise addition of the binary symplectic forms modulo 2. The problem, then, is to prove that there is a solution to a system of Boolean linear equations with $2\bar{n}$ equations and $\bar{n} - \bar{k}$ unknowns. This check can be done in polynomial time by rewriting the problem as an instance of XOR-SAT and proving whether the formula is satisfiable [51]. Both steps (checking for anticommutation and checking for group membership) take polynomial time, and must be repeated once for each error. A candidate solution to an instance of k -CODE DESIGN can thus be checked in polynomial time in the number of physical qubits, logical qubits, and errors. Thus the problem k -CODE DESIGN is in NP. $\square$

VI. QUANTUM CODE DESIGN AS A SATISFIABILITY PROBLEM

In this section, we formulate the quantum code design problem as an instance of SAT. This allows us to apply state-of-the-art SAT solvers to design codes, and to analyze the practical runtime of relevant problem instances.

A. SAT-Based Code Discovery

Consider the quantum code design problem (Def. 3), such that the goal is to build an $[[n, k]]$ quantum code that detects each error in the set $\mathcal{E} = \{E_1, \dots, E_s\}$. If

the goal is to find an $[[n, k, d]]$ code, then $\mathcal{E} = \bigcup_{w=1}^{d-1} \{P : P \in \mathcal{P}_n, |P| = w\}$, where $|\cdot|$ denotes the weight of a Pauli string. The solution to this problem is a set of stabilizer generators $g_1, \dots, g_{n-k}$. Let the unknown stabilizer generators have binary symplectic forms $u_1, \dots, u_{n-k}$ and let the errors have binary symplectic forms $\epsilon_1, \dots, \epsilon_s$. We can also refer to the parts of the binary vectors that correspond to X or Z Pauli operators: $u_i = [u_i^X | u_i^Z]$ and $\epsilon_j = [\epsilon_j^X | \epsilon_j^Z]$.

The stabilizer group formed by the unknown stabilizers must be abelian. In terms of their operator representations, this condition reads $[g_i, g_j] = 0, \forall i, j = 1, \dots, n-k$. The equivalent constraint on the binary symplectic forms is given by the function

$$\mathbf{C}_U(u) = \bigwedge_{i=1, j>i}^{n-k} \neg \langle u_i, u_j \rangle. \quad (19)$$

If the stabilizer generators mutually commute for some values $u_1, \dots, u_{n-k}$, then $\mathbf{C}_U(u) = 1$; otherwise $\mathbf{C}_U(u) = 0$.

In order to detect all errors in $\mathcal{E}$, for every error $E \in \mathcal{E}$, it must be true that either E anticommutes with at least one stabilizer generator of the code or E is in the stabilizer group. A syndrome bit corresponding to generators g_i can be evaluated as $\langle u_i, \epsilon_j \rangle$. To assert that the syndrome is not all zero, we take the logical disjunction of all syndrome bits

$$\mathbf{S}_j(u) = \bigvee_{i=1}^{n-k} \langle u_i, \epsilon_j \rangle. \quad (20)$$

Alternatively, we can check if the operator E_j is in the stabilizer group $\mathcal{S}$. If an error $E \in \mathcal{S}$ up to a phase, then $\exists s \in \mathbb{F}_2^{n-k}$ s.t. $E = g_1^{s_1} \dots g_{n-k}^{s_{n-k}}$. In terms of their binary symplectic forms, this expression becomes

$$\epsilon_j = s_1 u_1 \oplus \dots \oplus s_{n-k} u_{n-k}. \quad (21)$$

Using the fact that $a \oplus b = 1$ if $a \neq b$ and $a \oplus b = 0$ if $a = b$, Eq. (21) can be rewritten as

$$\epsilon_j \oplus s_1 u_1 \oplus \dots \oplus s_{n-k} u_{n-k} = \vec{0}, \quad (22)$$

where $\vec{0}$ is the vector of zeros with $2n$ elements. In the SAT formulation, the vector s is treated as a set of slack variables. The Boolean function that checks for group membership conditioned on the slack variables is

$$\mathbf{G}_j(u; s) = \bigwedge_{i=1}^{2n} \neg(\epsilon_j \oplus s_1 u_1 \oplus \dots \oplus s_{n-k} u_{n-k})_{[i]}, \quad (23)$$

where the notation $[i]$ refers to the i^{th} element of a binary vector. If the error E_j is in the stabilizer group, then the above formula is satisfiable for some assignment of the variables s . Otherwise, the formula is unsatisfiable. When Eqs. (20) and (23) are combined, we obtain the

condition requiring all errors in the set $\{\epsilon_j\}_{j=1}^s$ to be detectable

$$\mathbf{D}(u; s) = \bigwedge_{j=1}^s (\mathbf{S}_j(u) \vee \mathbf{G}_j(u; s)). \quad (24)$$

The stabilizer group forms a valid code to detect all errors in $\mathcal{E}$ given that

$$\mathbf{D}(u; s) \wedge \mathbf{C}_U(u). \quad (25)$$

evaluates to true.

B. SAT-Based Code Extension

We now consider the problem of extending a code to detect errors in $\mathcal{E}$, some of which may already be detected by the original code. Let $h_1, \dots, h_{n-k}$ denote the binary symplectic forms of the stabilizers of the original code and $u_1, \dots, u_m$ denote the m unknown stabilizers. The final stabilizer group, which is made up of the original and unknown stabilizers, must be abelian. As in the code discovery problem, the constraint that the unknown stabilizers commute is represented by Eq. (19). The constraint that the new stabilizer commute with the existing stabilizers is given by the function.

$$\mathbf{C}_H(u) = \bigwedge_{i=1}^{n-k} \bigwedge_{j=1}^m \neg \langle h_i, u_j \rangle. \quad (26)$$

The constraints enforcing error detectability have similar functional forms, differing only by minor modifications that account for the partition of the stabilizers into existing and unknown sets. The formula that ensures the syndrome is nonzero is now given by

$$\mathbf{S}_j(u) = \bigvee_{i=1}^{n-k} \langle h_i, \epsilon_j \rangle \vee \bigvee_{i=1}^m \langle u_i, \epsilon_j \rangle. \quad (27)$$

In addition, the check for group membership is updated to

$$\mathbf{G}_j(u; s) = \bigwedge_{i=1}^{2(n+m)} \neg(\epsilon_j \oplus s_1 h_1 \oplus \dots \oplus s_{n-k} h_{n-k} \oplus s_{n-k+1} u_1 \oplus \dots \oplus s_{n-k+m} u_m)_{[i]}. \quad (28)$$

We then impose the constraint in Eq. (24), with the updated forms of the functions $\mathbf{S}_j(u)$ and $\mathbf{G}_j(u; s)$, yielding a new form of the function $\mathbf{D}(u; s)$. All together, the total constraint for the code extension problem is

$$\mathbf{D}(u; s) \wedge \mathbf{C}_U(u) \wedge \mathbf{C}_H(u). \quad (29)$$

To support extension with mutation, we treat the existing stabilizer generators as Pauli strings where some of the entries — namely the X and Z bits for the original n qubits — are fixed and the rest are left variable. The equations for the SAT problem remain mostly unchanged, except for the omission of Eq. (26), which is now handled by Eq. (19).

C. From Detection to Correction

To correct errors instead of detecting them, the stabilizer group must obey the Knill-Laflamme conditions (Eq. (10)). For the case where all errors are Pauli strings, the errors are Hermitian and the condition can be rewritten as

$$E_i E_j \notin \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}, \quad \forall E_i, E_j \in \mathcal{E}. \quad (30)$$

The conditions to detect errors in the code discovery or extension problems (Eqs. (25) and (29), respectively) could equivalently be expressed as

$$E_i \notin \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}, \quad \forall E_i \in \mathcal{E}. \quad (31)$$

Note that Eq. (30) has much of the same structure as Eq. (31); the difference being that the Knill-Laflamme conditions apply to pairs of errors instead of single errors. In the binary symplectic representation, $E_i E_j \cong \epsilon_i \oplus \epsilon_j$. Substituting $\epsilon_i \rightarrow \epsilon_i \epsilon_j$ in all constraint functions, the updated constraints for code discovery become

$$\mathbf{S}_{i,j}(u) = \bigvee_{l=1}^{n-k} \langle u_l, \epsilon_i \oplus \epsilon_j \rangle, \quad (32)$$

$$\mathbf{G}_{i,j}(u; s) = \bigwedge_{l=1}^{2n} \neg(\epsilon_i \oplus \epsilon_j \oplus s_1 u_1 \oplus \dots \oplus s_{n-k} u_{n-k})[l], \quad (33)$$

$$\mathbf{D}(u; s) = \bigwedge_{i=1}^s \bigwedge_{j=i}^s (\mathbf{S}_{i,j}(u) \vee \mathbf{G}_{i,j}(u; s)). \quad (34)$$

Comparatively, the conditions for a code extension problem become

$$\mathbf{S}_{i,j}(u) = \bigvee_{l=1}^{n-k} \langle h_l, \epsilon_i \oplus \epsilon_j \rangle \vee \bigvee_{l=1}^m \langle u_l, \epsilon_i \oplus \epsilon_j \rangle, \quad (35)$$

$$\mathbf{G}_{i,j}(u; s) = \bigwedge_{l=1}^{2n} \neg(\epsilon_i \oplus \epsilon_j \oplus s_1 h_1 \oplus \dots \oplus s_{n-k} h_{n-k} \oplus s_{n-k+1} u_1 \oplus \dots \oplus s_{n-k+m} u_m)[l], \quad (36)$$

$$\mathbf{D}(u; s) = \bigwedge_{i=1}^s \bigwedge_{j=i}^s (\mathbf{S}_{i,j}(u) \vee \mathbf{G}_{i,j}(u; s)). \quad (37)$$

VII. PHASE TRANSITIONS IN THE K-CODE DESIGN PROBLEM

The reduction developed in Sec. V establishes that k -CODE DESIGN is NP-complete by polynomial-time reduction from SET COVER, while Sec. VI provides a reduction to mixed 2- and 3-SAT to readily compute a solution. Beyond worst-case complexity, however, it is well-known that NP-complete problems exhibit rich average-case behavior. One of the most striking phenomena is the existence of the *satisfiability phase transition*, where random

instances abruptly change from almost always satisfiable to almost always unsatisfiable as an order parameter is varied. This transition has been extensively studied for random k -SAT and is closely connected to the empirical hardness of complete SAT solvers [52–56]. Here, we demonstrate that the same phenomenon appears in random instances of k -CODE DESIGN.

For random k -SAT, one considers formulas with N Boolean variables and M clauses. The natural order parameter is then the *clause density*

$$\alpha = \frac{M}{N}, \quad (38)$$

which fixes the number of constraints per variable. For fixed N , many random formulas are generated at a given α , and the satisfiability probability

$$P_{\text{SAT}} = \frac{\# \text{ satisfiable formulas}}{\# \text{ total formulas}} \quad (39)$$

is measured. At small clause density, nearly every instance is satisfiable, while at sufficiently large density almost every instance is unsatisfiable. The critical clause density, α_c , at which the crossover occurs is known as the SAT–UNSAT transition [54, 57].

This transition is significant not only because it marks the onset of unsatisfiability, but also because it coincides with the region of greatest computational difficulty. Complete SAT solvers exhibit their largest runtimes near α_c , while formulas that are under- or over-constrained are typically solved quickly. For under-constrained instances, satisfying assignments are abundant and are found with little search. Conversely, for heavily constrained instances, contradictions are rapidly identified, allowing unsatisfiability to be certified efficiently. Near the transition, however, satisfying assignments become exponentially rare, forcing the solver to explore a large fraction of the search tree before reaching a conclusion. Consequently, the phase transition provides an experimentally observable signature of computational hardness in the average-case behavior.

In general, the hardest instances of NP-hard problems occur at the phase transition [58–61]. We therefore investigate whether an analogous transition occurs for random instances of k -CODE DESIGN. Consider a problem instance consisting of n physical qubits, the candidate set $\Sigma = \mathcal{P}_n$, and an error set

$$\mathcal{E} = \{E_1, E_2, \dots, E_m\}. \quad (40)$$

Following Sec. VI, this decision problem is reduced to an instance of mixed 2-3-SAT and solved using a heuristic SAT solver. To study the average-case complexity, we generate random error sets $\mathcal{E}$ by sampling m Pauli errors from the pool of all nontrivial errors of weight at most w , whose size is

$$M_{\leq w}(n) = \sum_{j=1}^w 3^j \binom{n}{j}. \quad (41)$$

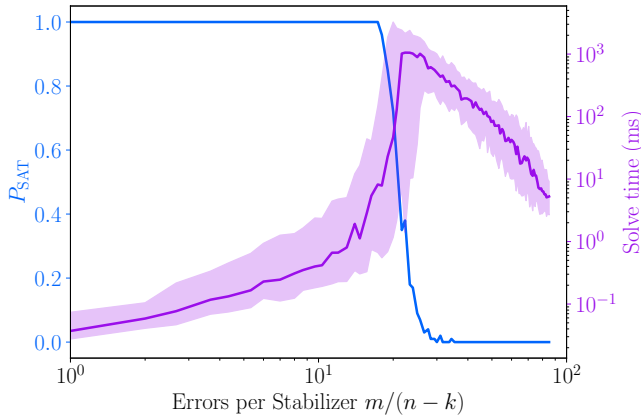


Figure 3. Demonstration of the SAT-UNSAT phase transition which we empirically observe for random instances of k -CODE DESIGN (Def. 3). The order parameter is the number of errors per stabilizer $m/(n-k)$ in the code design problem. In addition to the transition in satisfiability probability P_{SAT} at the critical value, the solve time peaks at this critical value. Additional numerical results and cases are discussed in detail in Appendix D.

For each (n, k, w, m) , many independent instances are generated and the satisfiability probability P_{SAT} is estimated. The number of sampled errors m sets the number of constraints imposed on the candidate code and therefore provides a natural control parameter for the SAT-UNSAT transition. For visualization, it is useful to normalize m by the number of stabilizer generators $n-k$, which measures the number of detected errors per stabilizer. For fixed (n, k, w) , small $m/(n-k)$ corresponds to weakly constrained instances that are typically satisfiable, whereas sufficiently large $m/(n-k)$ produces instances that are typically unsatisfiable. Although the constraint density $m/(n-k)$ is analogous to the clause density in random k -SAT, with both quantities characterizing the balance between available degrees of freedom and imposed constraints, we do not observe a universal critical value of $m/(n-k)$. Rather, the location of the transition depends on the code parameters (n, k, w) , motivating the parameter-dependent critical $m_c(n, k, w)$ developed below.

Figure 3 shows the satisfiability probability and SAT-solver runtime as functions of $m/(n-k)$ for the representative case $(n, k, w) = (4, 1, 4)$. For these fixed parameters, we observe a sharp finite-size SAT-UNSAT crossover near $m_c/(n-k) \approx 20$, accompanied by a peak in the solver runtime. Thus, random instances of k -CODE DESIGN display the characteristic easy-hard-easy behavior associated with satisfiability transitions. Instances far from the crossover are typically resolved quickly, while instances near the transition require substantially greater computational effort. The position of this crossover changes with (n, k, w) — we expand upon this point in Appendix D. So, the transition is characterized by the critical number of errors $m_c(n, k, w)$ rather

than by a constraint density $m/(n-k)$. The coincidence of the satisfiability threshold and the runtime peak therefore demonstrates that random instances of k -CODE DESIGN possess the same characteristic easy-hard-easy behavior observed in other NP-complete problems [54].

We develop a finite-size empirical scaling $\hat{m}_c(n, k, w)$ to determine the critical number of errors m_c at which the problem transitions from SAT to UNSAT. The model and its details are given in Appendix D. We can use the predicted critical value $\hat{m}_c(n, k, w)$ to predict the feasibility of constructing an $[[n, k, d]]$ stabilizer code. A distance- d code must detect every nontrivial Pauli error of weight less than d , so the corresponding k -CODE DESIGN instance contains $M_{\leq d-1}(n)$ errors. The quantum Gilbert-Varshamov (QGV) bound provides a sufficient condition for the existence of an $[[n, k, d]]$ stabilizer code [45, 62]

$$1 + M_{\leq d-1}(n) \leq 2^{n-k}, \quad (42)$$

while the quantum Singleton bound provides a necessary condition [45, 63]

$$n - k \geq 2(d-1). \quad (43)$$

Thus, codes satisfying Eq. (42) are guaranteed to exist, whereas codes violating Eq. (43) cannot exist. Between these rigorous bounds lies a broad region in which neither bound determines code existence. We classify this intermediate region using the empirical model developed in Appendix D.

For a target distance d , we set $w = d-1$ and compare the complete error-set size $M_{\leq d-1}(n)$ with the predicted critical value $\hat{m}_c(n, k, d-1)$. We predict the corresponding k -CODE DESIGN instance to be satisfiable when

$$M_{\leq d-1}(n) < \hat{m}_c(n, k, d-1), \quad (44)$$

and unsatisfiable when the inequality is reversed. Figure 4 shows the resulting phase diagram for distance-three codes. The Gilbert-Varshamov and Singleton bounds rigorously constrain the low- and high-rate regions, respectively, while the fitted SAT-UNSAT transition provides an empirical prediction of code existence in the contested region. The SAT formulation thus provides a rigorous framework to prove the (un)satisfiability of a given set of code parameters.

VIII. PHYSICS-INFORMED CODE DESIGN

We now demonstrate our framework by designing novel quantum codes starting from a set of problem stabilizers. We refer to this application of code extension as physics-inspired code design since the initial stabilizers arise from physical considerations of the problem. This can be seen as a generalization of quantum error detection/symmetry verification [64] as we show how to iteratively build new codes to increase performance as measured by accuracy of computed expectation values. In addition to designing

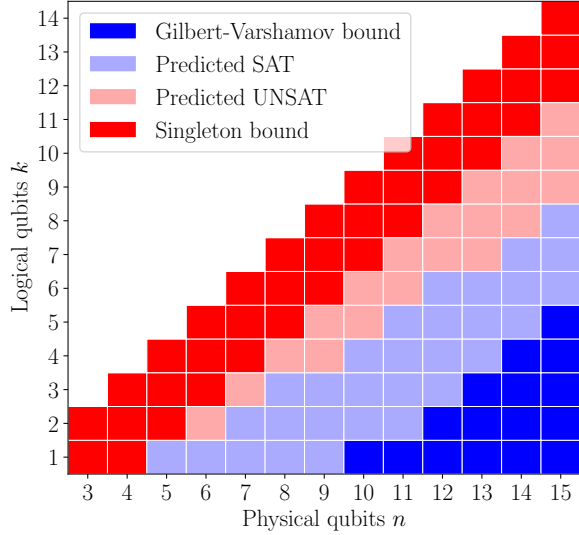


Figure 4. Predicted existence of $[[n, k, 3]]$ stabilizer codes as a function of the number of physical qubits n and logical qubits k . Dark blue denotes parameters for which code existence is guaranteed by the quantum Gilbert–Varshamov bound, while dark red denotes parameters excluded by the quantum Singleton bound. In the intermediate region, the empirical scaling model for the critical error-set size $\hat{m}_c(n, k, w)$, with $w = d - 1 = 2$, predicts satisfiable (light blue) and unsatisfiable (light red) k -CODE DESIGN instances.

new codes, we study their performance relative to the baseline codes.

We consider the Fermi-Hubbard model

$$H = -t \sum_{\langle i, j \rangle} \sum_{\sigma} \left(a_{i, \sigma}^{\dagger} a_{j, \sigma} + a_{j, \sigma}^{\dagger} a_{i, \sigma} \right) + U \sum_i a_{i, \uparrow}^{\dagger} a_{i, \uparrow} a_{i, \downarrow}^{\dagger} a_{i, \downarrow} - \mu \sum_i \sum_{\sigma} a_{i, \sigma}^{\dagger} a_{i, \sigma} - h \sum_i \left(a_{i, \uparrow}^{\dagger} a_{i, \uparrow} - a_{i, \downarrow}^{\dagger} a_{i, \downarrow} \right), \quad (45)$$

where $a_{i, \sigma} (a_{i, \sigma}^{\dagger})$ is the i th site annihilation (creation) operator for spin states $\sigma \in \{\uparrow, \downarrow\}$. The Hamiltonian is further specified by the tunneling amplitude t , while U is the Coulomb potential, μ is the chemical potential, and h is the magnetic field. In this model, the numbers of spin-up and spin-down fermions are separately conserved, providing a set of symmetries for the system. Using the Jordan-Wigner encoding and a staggered fermion ordering on the lattice, this conservation of particle number is enforced by a symmetry group with generators

$$g_{\uparrow} = (I \otimes Z)^{\otimes n/2} \quad \text{and} \quad g_{\downarrow} = (Z \otimes I)^{\otimes n/2}, \quad (46)$$

where n is the number of qubits in the system, which is equal to double the number of sites in the lattice. Since they are Pauli strings, these generators can also be thought of as the generators of a quantum error-detecting code. For an n -qubit system, the code has parameters $[[n, n - 2, 1]]$. We call the stabilizer code with generators given by Eq. (46) the Fermi-Hubbard code.

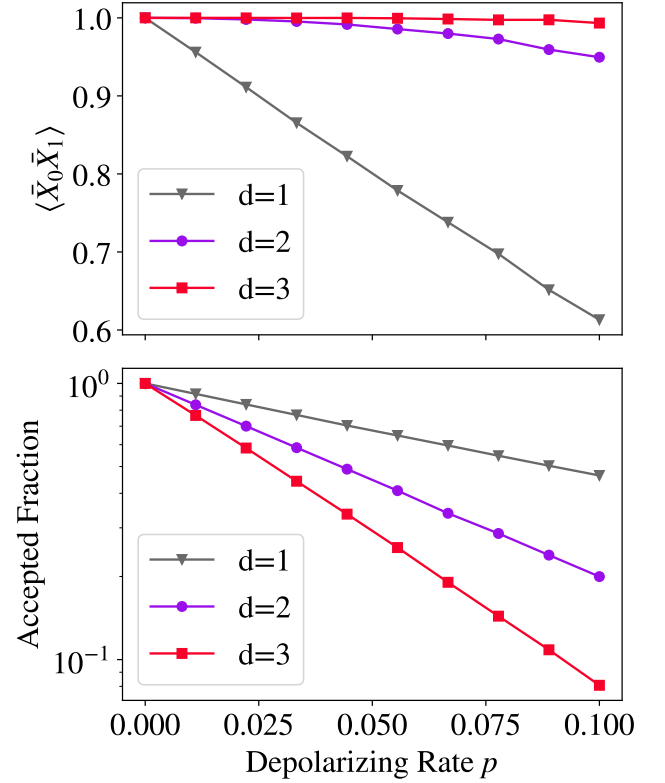


Figure 5. Demonstration of improved accuracy in error-detected expectation values for physics-informed codes. Here we start from the symmetries (46) of an $n = 6$ Fermi-Hubbard Hamiltonian and extend to distance $d = 2$ and $d = 3$ codes. (top) Expectation value of the logical observable $\bar{X}_0 \bar{X}_1$ in a logical Bell state vs. depolarizing noise rate p . Without noise, this expectation value should be one. Here, logical state preparation is assumed to be done fault-tolerantly, and depolarizing noise acts on physical qubits after each logical operation. As can be seen, extending the codes to higher distances significantly improves the accuracy of the expectation value. As discussed in the main text, this can be viewed as a generalization of symmetry verification [64] to improve error-mitigated expectation values. (bottom) The accepted fraction of samples (i.e., the number of sampled codewords) vs. the depolarizing noise rate p . As can be seen, codes extended to higher distances require slightly more samples due to a larger number of physical qubits and therefore larger number of noisy channels. This increase in required shots is significantly less than what would be required for concatenated codes or arbitrary (non physics-inspired) codes. This accepted fraction is present only in error detection where expectation values are computed by filtering in post-processing.

We apply our code design framework to extend the Fermi-Hubbard code and compute the accuracy of error-detected (symmetry verified) expectation values. Fig. 5 shows the expectation value of the logical observable $\bar{X}_0 \bar{X}_1$ versus the noise rate p (defined below). Here, we start from an $n = 6$ Fermi-Hubbard code with distance $d = 1$ and extend to $d = 2$ and $d = 3$. The distance two code has $n = 8$ physical qubits and the distance

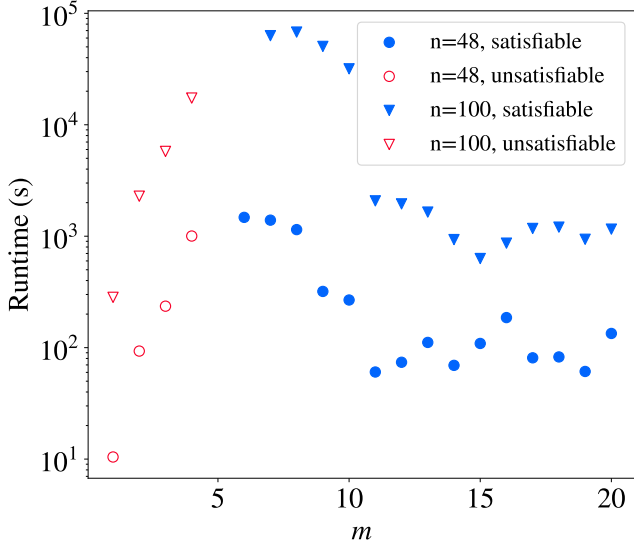


Figure 6. Time to extend the Fermi-Hubbard code (46) from distance $d = 1$ to distance $d = 3$ vs. the number of qubits/stabilizers m added. Timing is shown for both $n = 48$ and $n = 100$ physical qubits. As discussed in Sec. VII and demonstrated in Fig. 3, runtime peaks near the SAT-UNSAT phase transition, which here occurs around $m = 5$. All $n = 48$ code extensions run in at most twenty minutes of CPU time. Most $n = 100$ code extensions run in around 20 minutes, while the hardest cases take around a day.

three code has $n = 12$ qubits. We simulate an error detection experiment in which the logical state $|0\rangle^{\otimes k}$ is prepared, then logical Hadamard and CNOT operations are applied to prepare a Bell state on the first two logical qubits. In the logical Bell state, the ideal expectation value is $\langle \bar{X}_0 \bar{X}_1 \rangle = 1$. The logical state preparation of $|0\rangle^{\otimes k}$ is treated as error free. Between each logical gate, physical qubits are subjected to independent depolarizing noise of rate p . During syndrome extraction, bit-flips occur on the measured qubit with probability p . As the code is extended, the error in the expectation value of the observable decreases dramatically.

These results demonstrate the utility of physics-inspired code design and motivates the design of even larger codes. To demonstrate this, we next extend to $n = 48$ and $n = 100$ Fermi-Hubbard codes from distance one to distance three. The results are shown in Fig. 6. As mentioned in the discussion of prior work in Sec. I, the problem size of $n = 100$ has become somewhat of a target in recent literature — e.g., Ref. [28] presents numerical results for $n = 25$ qubits and a roadmap to $n = 100$ qubits with reinforcement learning, and Ref. [29] uses a multi-agent system to optimize quantum codes and enable discovery of a $[[100, 50, 4]]$ code in one hour. Here, our SAT framework can easily handle the more general problem of physics-informed code design for the $n = 100$ qubit target in just minutes to hours as shown in Fig. 6. Indeed, most code design instances with $m \geq 10$ run in just

minutes, while the hardest instances around $5 \leq m \leq 10$ take up to a day. This is in line with the phase transition in satisfiability/hardness demonstrated in Sec. VII, and we see that instances with $m \leq 5$ turn out to be unsatisfiable. As a more direct comparison to Ref. [29], we also applied our code design formulation to the same $[[100, 50, 4]]$ code *discovery* problem, and found a satisfiable instance in under five minutes of CPU time compared to an hour in Ref. [29]. This example is included under *Code and Data Availability* in Sec. X.

IX. HARDWARE-AWARE CODE DESIGN

The formulation of code extension in terms of a satisfiability problem, outlined in Section VI, is designed to find m new stabilizer generators such that all errors in a given set can be detected. Given some maximal number of new qubits m to be added when extending a code, it is not always possible to make every error in the set detectable. Instead, one would want to be able to detect as many errors as one can with the number of additional qubits (and thus extra stabilizers) allotted. This problem is an instance of maximum satisfiability, where the constraints that a given error is detectable are soft constraints, and the constraints that the group is abelian are treated as hard constraints. Given a hardware error model, a weight can be assigned to the clauses for each error informed by that model. A solution to this maximum satisfiability problem will then give a code that can detect as many of the errors as possible, prioritizing errors that are more likely on a given device.

On a given quantum device, certain Pauli errors will occur more frequently than others. It is common to consider an error model where the rate of Pauli Z errors differs from that of Pauli X and Y errors. If p_X , p_Y , and p_Z are the probabilities of X , Y , and Z errors on any given qubit, respectively, then this model is parameterized by a bias parameter

$$\eta = \frac{p_Z}{p_{XY}} \quad (47)$$

where $p_{XY} = p_X = p_Y$. If, for example, a Z -type error is much more likely than an X - or Y -type error, then the algorithm is likely to produce stabilizers that contain more X -type stabilizers.

A. Hardware-Aware Fermi-Hubbard Code Extensions

To demonstrate tailoring a code to biased noise in this setting, an extension to the Fermi-Hubbard code with stabilizer generators defined in Eq. (46) is constructed, where the extended code either is or is not tailored to the bias described above. Two extra qubits and stabilizers are added to allow for this extension. For some fixed

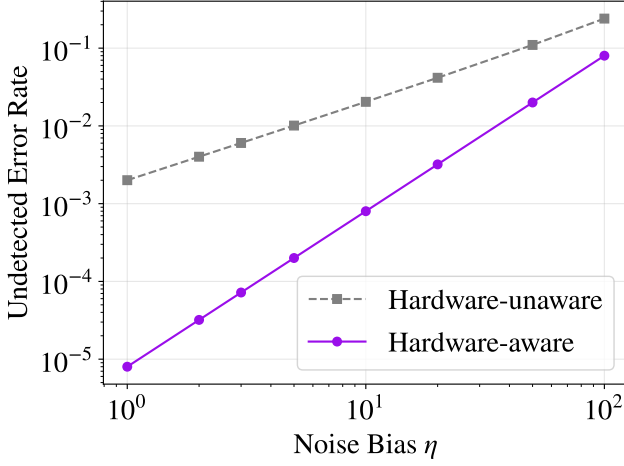


Figure 7. Sum of the rates of all undetectable weight-two errors versus the noise bias parameter η for an extension to the $n = 6$ Fermi-Hubbard code with $m = 2$ added qubits. As shown, the hardware-aware code has a significantly lower undetected error rate than the hardware-unaware code for all noise biases. The overall rate of errors increases as the noise bias increases, so the two codes converge in the limit of large η .

value of $p_X = p_Y$, we sweep over values of η , giving $p_Z = \eta p_X$.

Let $P = \bigotimes_{i=1}^n \sigma_i$ be an n -qubit error, where $\sigma_i \in \{I, X, Y, Z\}$. We compute the rate of the error as

$$r = \prod_{i=1}^n r_i, \quad (48)$$

where $r_i = p_Z$ if $\sigma_i = Z$, $r_i = 1 - p_Z - 2p_X$ if $\sigma_i = I$, and $r_i = p_X$ otherwise. In Fig. 7, we compare the total rate of undetectable weight-two errors as a function of the noise bias η for codes that are tailored and not tailored to the underlying noise. For the hardware-aware code, the weights of the errors are the same as in the biased error model. For the hardware-unaware code, uniform weights are given to each error. As can be seen, the hardware-aware code has a lower undetected error rate, lower by as much as two orders of magnitude for small noise biases. This demonstrates the ability of our code design framework to adapt to given error models to improve error detection/correction under realistic hardware conditions.

B. Hardware-Aware Surface Codes

Next, we consider designing surface codes under biased noise. For a biased noise model, it is known that the XZZX surface code has a higher threshold than the rotated CSS surface code [65]. The XZZX surface code is derived from the CSS surface code by applying Hadamard gates on alternating qubits, which results in stabilizers of

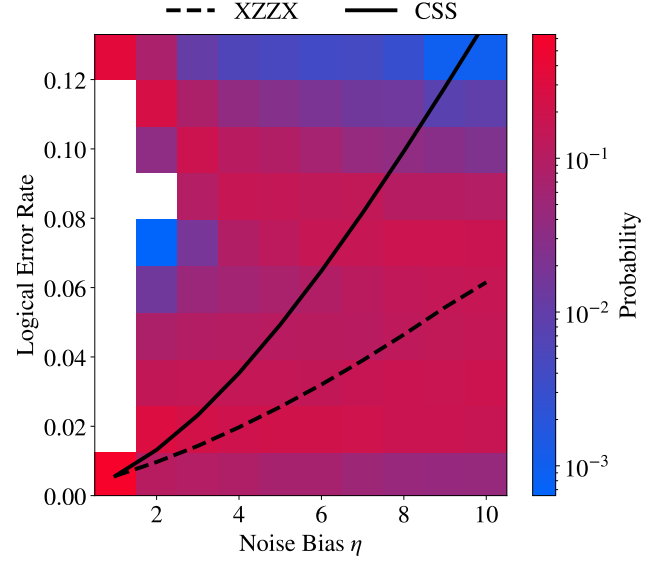


Figure 8. Distribution of logical error rate in designed surface codes vs. the bias parameter $\eta = p_X/p_{YZ}$. The CSS and XZZX codes are shown overlaying the designed codes. Here, 10^5 designed codes are sampled uniformly at random from solutions to the SAT problem, and the colored square show frequencies (probabilities) of codes with the given logical error rate for the given noise bias η . (White squares represent zero.) Each column of the plot represents a probability distribution over the logical error rate for some value of η . As can be seen, many designed codes have a lower logical error rate than the XZZX surface code, which is a current state-of-the-art solution for biased noise.

the form XZZX (or XZ on the boundaries). In general, there are many codes that share the locality properties of the surface code while potentially being more suitable for certain noise models. Here we develop a technique to search for such codes using our SAT formulation.

To discover codes with the same locality as the surface code, each stabilizer is assigned some set of physical qubits where it can act non-trivially. In a rotated surface code, each physical qubit is assigned to a vertex of a graph, and each stabilizer corresponds to a face of the graph. When discovering surface-like codes, each unknown stabilizer generator is assigned to a face of the graph. A constraint is then imposed such that all qubits which do not correspond to vertices on that face of the graph must be assigned as identity in that stabilizer generator. If F_i is the i^{th} face of the graph and $u_i = [u_i^X | u_i^Z]$ is the binary symplectic form of the i^{th} unknown stabilizer, then the constraint function is

$$\bigwedge_{l \notin F_i} \neg(u_{i[l]}^X \vee u_{i[l]}^Z). \quad (49)$$

All other qubits are constrained to be the identity (both the X and Z bits on that qubit are 0).

The constraint in Eq. (49) is imposed along with the usual constraints that the code must be abelian and that

all errors up to the distance of the code must be detectable or in the stabilizer group. Using the SAT formulation, all codes that satisfy the constraints are seen as “equal” in that they can detect all desired errors and have an abelian stabilizer group. However, not all codes will have the same logical error rate under a given error model. Using an algorithm for approximately uniform sampling of SAT solutions, we can characterize the distribution of logical error rates among codes that share the same locality as the surface code but differ in their assignments of Pauli operators to the qubits. The CSS and XZZX surface codes are two codes from this distribution.

All codes discovered by sampling SAT solutions are subjected to a Z -basis memory experiment where the logical $|0\rangle$ state is prepared, the qubits are subjected to noise, and then the stabilizer generators and logical observable $\bar{Z}$ are measured. Decoding the syndromes gives the logical error rate, defined as the rate at which the decoder’s predictions do not match the observed eigenvalue of the logical observable. Because the memory experiment is performed in the Pauli Z basis, we choose to bias errors towards Pauli X . We choose an independent model in which Pauli Y and Z errors have some fixed, equal probability p_{YZ} and the probability p_X of an X error varies. The bias is quantified by the parameter $\eta = p_X/p_{YZ}$. To find the lowest possible logical error rate that the code is capable of, a maximum likelihood decoder is used. The codes discovered by this method will generally not be CSS codes, and as such may not perform well with a minimum-weight perfect matching decoder.

Figure 8 shows the logical error rate as a function of η for the distance three CSS and XZZX surface codes, along with the distribution of the logical error rate at each value of η for codes that were sampled from the SAT problem. At low values of η , the distribution has a large peak near the CSS and XZZX codes, and a smaller peak at a high logical error rate. As η increases, the peaks in the distributions move closer together. As can be seen, many codes perform better than the XZZX code, a current state-of-the-art surface code for biased noise. Interestingly, we find codes with logical error rates between the CSS and XZZX codes. These codes represent novel surface codes which, in a way, interpolate between the CSS and XZZX surface codes. We note that while these codes have the same locality as the surface code, they may not guarantee efficient decoding. Nonetheless, codes with lower logical error rates than the XZZX code are particularly notable, and advances in efficient maximum likelihood decoding may make the codes discovered here a practical choice for implementing error correction on devices with constrained connectivity between physical qubits.

X. CONCLUSION

We have developed a general, flexible, and performant framework for quantum code design, analyzed its worst-case and average-case theoretical complexity, and demonstrated its utility by designing novel codes at scale with improved performance over state-of-the-art methods. In particular, we designed physics-inspired codes which incorporate symmetries of the Fermi-Hubbard model and demonstrated improved accuracy of observable expectation values in quantum error detection experiments. Additionally, we have designed hardware-aware surface codes which have a lower logical error rate than the XZZX surface code for biased noise. Beyond these examples, we present the framework to design additional quantum codes “from scratch” (code discovery) or from initial specifications (code extension) with the ability to incorporate realistic hardware noise. This ability will be crucial for experimentally demonstrating impactful applications of quantum computers.

In this work, we restricted to abelian stabilizer codes in our examples, but our framework is extensible to alternative classes as well. Future work in designing entanglement-assisted codes [66], subsystem codes [67, 68], and non-abelian codes will likely lead to additional impact. While we did not impose conditions on logical operators in this work, this can also be incorporated in future work. Current quantum error correction experiments are mostly memory experiments, where a state is prepared and then preserved through multiple rounds of error correction, but in practical applications it is essential to perform logical operations with encoded qubits. Extending (or in general designing) codes with conditions on logical operations — for example, transversal operations — is the subject of future work. Finally, while we used our SAT formulation to design codes with improved logical error rates, this is an “emergent” property of the code that we did not incorporate into the design, but rather tested after. It is unlikely that one can easily incorporate the logical error rate into the SAT formulation or any formulation for code design since this is a property of the full code that must be measured by a separate (numerical) experiment. We leave as an open problem the possibility to directly incorporate logical error rate into the code design problem in future work.

We have proved general theoretical results that apply to all code design frameworks — in particular, that the code design problem is NP-complete — and we have demonstrated that our SAT formulation is capable of designing novel codes with advantages over current state of the art. While there has been tremendous progress in quantum hardware, a large-scale impactful application of an error corrected quantum computer remains a long sought-after milestone in the field. Designing quantum codes tailored to both the problem and the hardware can accelerate the timeline to this goal. Our work has already discovered novel advantageous codes, and our framework can continue to do so when applied to future problems.

Code and data availability – Our SAT formulation for code design is open source and freely available at [69]. This repository contains all examples and data to reproduce the results of this work.

Acknowledgments – This work was supported in part through computational resources and services provided

by the Institute for Cyber-Enabled Research at Michigan State University. B.D., W.M.W., M.L.L. and G.Q. acknowledge funding by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research in Quantum Computing under Award No. DE-SC0025509.

-
- [1] A. R. Calderbank and P. W. Shor, *Phys. Rev. A* **54**, 1098 (1996).
 - [2] A. Cross, G. Smith, J. A. Smolin, and B. Zeng, *IEEE Transactions on Information Theory* **55**, 433 (2009).
 - [3] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, *Nature* **627**, 778 (2024).
 - [4] A. Kitaev, A. Shen, and M. Vyalyi, *Classical and Quantum Computation*, Graduate Studies in Mathematics, Vol. 47 (American Mathematical Society, Providence, Rhode Island, 2002).
 - [5] M. B. Hastings and J. Haah, *Quantum* **5**, 564 (2021).
 - [6] C. Gidney, “How to factor 2048 bit RSA integers with less than a million noisy qubits,” (2025), arXiv:2505.15917 [quant-ph].
 - [7] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Phys. Rev. A* **86**, 032324 (2012).
 - [8] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, *Phys. Rev. X* **2**, 031007 (2012).
 - [9] J. O’Gorman and E. T. Campbell, *Phys. Rev. A* **95**, 032338 (2017).
 - [10] V. Gheorghiu and M. Mosca, “Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes,” (2019), arXiv:1902.02332 [quant-ph].
 - [11] C. Gidney and M. Ekerå, *Quantum* **5**, 433 (2021).
 - [12] P.-L. Dallaire-Demers, W. Doyle, and T. Foo, “Brace for impact: ECDLP challenges for quantum cryptanalysis,” (2026), arXiv:2508.14011 [quant-ph].
 - [13] H. Zhou, C. Duckering, C. Zhao, D. Bluvstein, M. Cain, A. Kubica, S.-T. Wang, and M. D. Lukin, in *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, ISCA ’25 (Association for Computing Machinery, New York, NY, USA, 2025) p. 1432a–1448.
 - [14] M. Cain, Q. Xu, R. King, L. R. B. Picard, H. Levine, M. Endres, J. Preskill, H.-Y. Huang, and D. Bluvstein, “Shor’s algorithm is possible with as few as 10,000 reconfigurable atomic qubits,” (2026), arXiv:2603.28627 [quant-ph].
 - [15] *Nature* **614**, 676 (2023).
 - [16] Z. Ni, S. Li, X. Deng, Y. Cai, L. Zhang, W. Wang, Z.-B. Yang, H. Yu, F. Yan, S. Liu, *et al.*, *Nature* **616**, 56 (2023).
 - [17] R. Acharya *et al.*, *Nature* **638**, 920 (2025).
 - [18] T. He, W. Lin, R. Wang, Y. Li, J. Bei, J. Cai, S. Cao, D. Chen, K. Chen, X. Chen, *et al.*, *Phys. Rev. Lett.* **135**, 260601 (2025).
 - [19] B. L. Brock, S. Singh, A. Eickbusch, V. V. Sivak, A. Z. Ding, L. Frunzio, S. M. Girvin, and M. H. Devoret, *Nature* **641**, 612 (2025).
 - [20] S. J. Evered, D. Bluvstein, M. Kalinowski, S. Ebadi, T. Manovitz, H. Zhou, S. H. Li, A. A. Geim, T. T. Wang, N. Maskara, H. Levine, G. Semeghini, M. Greiner, V. Vuletić, and M. D. Lukin, *Nature* **622**, 268 (2023).
 - [21] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter, J. P. Bonilla Ataides, N. Maskara, I. Cong, X. Gao, P. Sales Rodriguez, T. Karolyshyn, G. Semeghini, M. J. Gullans, M. Greiner, V. Vuletić, and M. D. Lukin, *Nature* **626**, 58 (2024).
 - [22] C. N. Self, M. Benedetti, and D. Amaro, *Nature Physics* **20**, 219 (2024).
 - [23] C. N. Self, M. Benedetti, D. Amaro, Z. He, R. Shaydulin, and M. Pistoia, *Communications Physics* **8**, 217 (2025).
 - [24] A. Paetznick, B. W. Reichardt, M. P. d. Silva, C. Ryan-Anderson, D. Aasen, J. M. Bello-Rivas, J. P. Campora, R. Chao, A. Chernoguzov, W. van Dam, J. M. Dreiling, C. Foltz, F. Frachon, J. P. Gaebler, T. M. Gatterman, L. Grans-Samuelsson, D. Gresh, D. Hayes, N. Hewitt, C. Holliman, C. V. Horst, J. Johansen, D. Lucchetti, Y. Matsuoka, M. Mills, S. A. Moses, B. Neyenhuis, A. Paz, J. Pino, P. Siegfried, A. Sundaram, D. Tom, Z. Wang, S. J. Wernli, M. Zanner, R. P. Stutz, and K. M. Svore, *Nature* **654**, 349 (2026).
 - [25] Y. Jin, Z. He, T. Hao, S. Omanakuttan, D. Amaro, S. Tannu, R. Shaydulin, and M. Pistoia, “Iceberg beyond the tip: Co-compilation of a quantum error detection code and a quantum algorithm,” (2026), arXiv:2504.21172 [quant-ph].
 - [26] A. Vezvaei, V. Tripathi, M. Morford-Oberst, F. Butt, V. Kasatkin, and D. A. Lidar, *Nature Communications* **17**, 3281 (2026).
 - [27] U. Kapshikar and S. Kundu, *IEEE Transactions on Information Theory* **69**, 6293 (2023).
 - [28] J. Olle, R. Zen, M. Puviani, and F. Marquardt, *npj Quantum Information* **10**, 126 (2024).
 - [29] R. D. Guerrero, “Game-theoretic discovery of quantum error-correcting codes through nash equilibria,” (2026), arXiv:2510.15223 [quant-ph].
 - [30] J. M. Koh, A. Gong, A. C. Diaconu, D. B. Tan, A. A. Geim, M. J. Gullans, N. Y. Yao, M. D. Lukin, and S. Majidi, “Entangling logical qubits without physical operations,” (2026), arXiv:2601.20927 [quant-ph].
 - [31] T. Peham, L. Schmid, L. Berent, M. Mäijler, and R. Wille, *PRX Quantum* **6**, 020330 (2025).
 - [32] M. Ehatamm, Y. Lee, X. Wu, and R. Tao, “End-to-end formalization of quantum error correction,” (2026), arXiv:2605.16523 [quant-ph].
 - [33] R. M. Karp, in *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, edited by R. E. Miller, J. W. Thatcher, and

- J. D. Bohlinger (Springer US, Boston, MA, 1972) pp. 85–103.
- [34] P. Iyer and D. Poulin, *IEEE Transactions on Information Theory* **61**, 5209–5223 (2015).
- [35] A. Ignatiev, A. Morgado, and J. Marques-Silva, in *SAT* (2018) pp. 428–437.
- [36] A. Ignatiev, Z. L. Tan, and C. Karamanos, in *SAT* (2024) pp. 4:1–4:11.
- [37] E. Knill and R. Laflamme, “Concatenated quantum codes,” (1996), arXiv:quant-ph/9608012 [quant-ph].
- [38] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge Series on Information and the Natural Sciences (Cambridge University Press, 2000).
- [39] R. Hill, *Designs, Codes and Cryptography* **17**, 151 (1999).
- [40] R. Hill and P. Lizak, in *Proceedings of 1995 IEEE International Symposium on Information Theory* (1995) pp. 345–.
- [41] M. Grassl, in *2007 IEEE International Symposium on Information Theory* (IEEE, 2007) pp. 476–480.
- [42] Y. Yanay, “Learning better error correction codes with hybrid quantum-assisted machine learning,” (2026), arXiv:2601.08014 [quant-ph].
- [43] C. Cao and B. Lackey, “Growing sparse quantum codes from a seed,” (2025), arXiv:2507.13496 [quant-ph].
- [44] G. M. Sommers, D. A. Huse, and M. J. Gullans, *Phys. Rev. Res.* **7**, 023086 (2025).
- [45] D. Gottesman, (1997), PhD Dissertation.
- [46] N. Lacroix *et al.*, *Nature* **645**, 614 (2025).
- [47] K. Wang, Z. Lu, C. Zhang, G. Liu, J. Chen, Y. Wang, Y. Wu, S. Xu, X. Zhu, F. Jin, Y. Gao, Z. Tan, Z. Cui, N. Wang, Y. Zou, A. Zhang, T. Li, F. Shen, J. Zhong, Z. Bao, Z. Zhu, Y. Han, Y. He, J. Shen, H. Wang, J.-N. Yang, Z. Song, J. Deng, H. Dong, Z.-Z. Sun, W. Li, Q. Ye, S. Jiang, Y. Ma, P.-X. Shen, P. Zhang, H. Li, Q. Guo, Z. Wang, C. Song, H. Wang, and D.-L. Deng, *Nature Physics* **22**, 308 (2026).
- [48] F. v. Iv, M. Leib, and F. R. F. Pereira, *Physical Review Research* **6**, 043123 (2024).
- [49] I. Chuang, A. Cross, G. Smith, J. Smolin, and B. Zeng, *Journal of Mathematical Physics* **50** (2009), 10.1063/1.3086833.
- [50] S. Shamshiri and K.-T. Cheng, in *2010 IEEE International Test Conference* (2010) pp. 1–10.
- [51] T. J. Schaefer, in *Proceedings of the tenth annual ACM symposium on Theory of computing - STOC '78* (ACM Press, San Diego, California, United States, 1978) pp. 216–226.
- [52] S. Kirkpatrick and B. Selman, *Science* **264**, 1297 (1994), <https://www.science.org/doi/pdf/10.1126/science.264.5163.1297>.
- [53] I. P. Gent and T. Walsh, in *Proceedings of the 11th European Conference on Artificial Intelligence, ECAI'94* (John Wiley & Sons, Inc., USA, 1994) p. 105–109.
- [54] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, *Nature* **400**, 133 (1999).
- [55] W. Fernandez de la Vega, *Theoretical Computer Science* **265**, 131 (2001).
- [56] T. Friedrich, A. Krohmer, R. Rothenberger, and A. Sutton, in *31st AAAI Conference on Artificial Intelligence, AAAI 2017* (AAAI press, 2017) pp. 3893–3899, 31st AAAI Conference on Artificial Intelligence, AAAI 2017 ; Conference date: 04-02-2017 Through 10-02-2017.
- [57] S. Franz, G. Parisi, M. Sevelev, P. Urbani, and F. Zamponi, *SciPost Phys.* **2**, 019 (2017).
- [58] W. Zhang and R. E. Korf, *Artificial Intelligence* **81**, 223 (1996), frontiers in Problem Solving: Phase Transitions and Complexity.
- [59] T. Hogg, B. A. Huberman, and C. P. Williams, *Artificial Intelligence* **81**, 1 (1996), frontiers in Problem Solving: Phase Transitions and Complexity.
- [60] D. Coppersmith, D. Gamarnik, M. Hajiaghayi, and G. B. Sorkin, *Random Structures & Algorithms* **24**, 502 (2004), <https://onlinelibrary.wiley.com/doi/pdf/10.1002/rsa.20015>.
- [61] S. Santra, G. Quiroz, G. Ver Steeg, and D. A. Lidar, *New Journal of Physics* **16**, 045006 (2014).
- [62] A. Ekert and C. Macchiavello, *Phys. Rev. Lett.* **77**, 2585 (1996).
- [63] A. Ketkar, A. Klappenecker, S. Kumar, and P. K. Sarvepalli, “Nonbinary stabilizer codes over finite fields,” (2005), arXiv:quant-ph/0508070 [quant-ph].
- [64] Z. Cai, *Quantum* **5**, 548 (2021), arXiv:2101.03151 [quant-ph].
- [65] J. P. Bonilla Ataides, D. K. Tuckett, S. D. Bartlett, S. T. Flammia, and B. J. Brown, *Nature Communications* **12**, 2172 (2021).
- [66] T. Brun, I. Devetak, and M.-H. Hsieh, *Science* **314**, 436 (2006), arXiv:quant-ph/0610092.
- [67] P. Zanardi and S. Lloyd, *Phys. Rev. A* **69**, 022313 (2004).
- [68] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli, “Subsystem codes,” (2006), arXiv:quant-ph/0610153 [quant-ph].
- [69] B. DalFavero, V. Russo, and E. Egger, “Satisfying codes GitHub,” .
- [70] F. Gaitan, “Quantum stabilizer codes,” in *Quantum Error Correction and Fault Tolerant Quantum Computing* (CRC Press, 2013).
- [71] N. Delfosse and B. W. Reichardt, “Short shor-style syndrome sequences,” (2020), arXiv:2008.05051 [quant-ph].
- [72] M. Grassl, “Bounds on the minimum distance of linear codes and quantum codes,” Online available at <http://www.codetables.de> (2007), accessed on 2026-07-20.
- [73] M. Grassl, in *Discovering Mathematics with Magma — Reducing the Abstract to the Concrete*, Algorithms and Computation in Mathematics, Vol. 19, edited by W. Bosma and J. Cannon (Springer, Heidelberg, 2006) pp. 287–313.
- [74] W. Bosma, J. Cannon, and C. Playoust, *Journal of Symbolic Computation* **24**, 235 (1997).
- [75] A. E. Brouwer, in *Handbook of Coding Theory*, edited by V. S. Pless and W. Huffman (Elsevier, Amsterdam, 1998) Chap. 4, pp. 295–461.

Appendix A: Subroutine to find the generators of the normalizer

Finding the generators of the normalizer of a subgroup of Pauli group can be done in polynomial time. Take $\Sigma = \mathcal{N}(S)$, where

$$\mathcal{N}(S) = \{Q \in \mathcal{P}_n \mid [Q, S] = 0, \forall S \in \mathcal{S}\}. \quad (\text{A1})$$

Use the binary symplectic representation, so a Pauli string S maps to the vector $s \in \mathbb{F}_2^{2n}$. Then commutation is

$$a\Lambda b^T = 0, \quad (\text{A2})$$

where

$$\Lambda = \begin{pmatrix} 0 & I_n \\ I_n & 0 \end{pmatrix} \quad (\text{A3})$$

Let

$$S = \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{pmatrix} \quad (\text{A4})$$

be the matrix representation the stabilizer set. Then

$$S\Lambda q^T = 0 \quad (\text{A5})$$

implies Q is in the normalizer of $\mathcal{S}$. The space is

$$\mathcal{N}(\mathcal{S}) \cong \ker(S\Lambda) \quad (\text{A6})$$

The matrix $M\Lambda$ has dimension $m \times 2n$. Compute the rank r of $M\Lambda$. Then $\dim \mathcal{N}(S) = 2n - r$. A basis can be found in polynomial time by Gaussian elimination, $\mathcal{O}(mn^2) \in \mathcal{O}(n^3)$, since the number of stabilizers is always less than the number of physical qubits, else the code space is trivial. Given the generators of the normalizer, products of them can be taken to generate the whole set of candidate Pauli strings.

Appendix B: Hardness of finding code extensions which preserve the number of logical qubits

Theorem 2 guarantees a distance-increasing extension; however, k will increase under this procedure. As a minor corollary, we show that finding a concatenation that maintains the number of logical qubits reduces to finding a quantum code with the added constraint that the code must encode exactly k logical qubits.

Corollary 1. *Determining if there exists a concatenation of an $[[n, k, d]]$ code C_1 with an outer code C_2 such that the resulting concatenated code $C_{1,2}$ is an $[[n+m, k, d']]$ code with $d' > d$ and k remaining the same reduces to solving an exact instance of k -CODE-DESIGN for the outer code.*

Proof. For this proof, we will define a specific procedure to create a concatenated code $C_{1,2}$ by enforcing certain properties on the outer code. Consider an $[[n, k, d]]$ inner code C_1 and an outer code C_2 that is an $[[n_2, k_2, d_2]]$ code. We require that $n_2 \bmod k = 0$ such that we can define a parameter $r := \frac{n_2}{k}$.

We define the concatenation procedure to begin by encoding rn input registers via the C_1 code. This will result in rk logical qubits of the C_1 code. Let the outer code, C_2 act on these logical qubits, such that $n_2 = rk$. Since the final number of logical qubits must remain the same, we require that this second code encode $k_2 = k$ qubits.

Finally, to ensure that the distance increases for the final code, we use the following relation for the distance

d' of a concatenated code constructed in this manner [70, Chp. 3]

$$d' = \frac{d_1 d_2}{k_2} = \frac{d d_2}{k}. \quad (\text{B1})$$

Since we require $d' > d$, we require that $d_2 > k$.

Thus, C_2 must be a $[[rk, k, d_2]]$ code with $d_2 > k$. Note that for the simplest case where $d_2 = k + 1$, the Singleton bound is satisfied as long as $r > 1$. In general, the Singleton bound is satisfied for all $d_2 = k + q$ where $q \leq (r - 2)k + 1$.

Thus, the problem of extending an $[[n, k, d]]$ code via concatenation such that its distance increases but the number of logical qubits remains constant can be solved by solving an exact instance of the k -CODE-DESIGN problem for an $[[rk, k, d_2]]$ code with $(r - 1)k + 1 \geq d_2 > k$. $\square$

Such extensions always exist for $k = 1$. For $k > 1$, the question is a bit trickier. For the $[[4, 2, 2]]$ code, $r = 2$ suffices with C_2 being the $[[8, 2, 3]]$ Hermitian code [71]. If C_1 is the $[[5, 3, 2]]$ code, $r = 2$ would require a $[[10, 3, 4]]$, which does not exist [72–75]; however, for $r = 3$, there exists a $[[15, 3, 5]]$ code that would suffice as a choice for C_2 . Indeed, for a sufficiently large r , such an extension becomes increasingly likely. In fact, by the quantum Gilbert-Varshamov bound [45], we can guarantee the existence of this extension by choosing r satisfying

$$\sum_{j=0}^k \binom{rk}{j} 3^j \leq 2^{(r-1)k}. \quad (\text{B2})$$

Appendix C: Proof of code extension without mutation

Here we prove Theorem 3. First we review the structure of bivariate bicycle (BB) codes as given in Ref. [3]. Let S_ℓ be the $\ell \times \ell$ shift matrix with entries $[S_\ell]_{ij} = \delta_{i, (j+1) \bmod \ell}$, where δ is the Kronecker delta. S_ℓ represents the cyclic permutation $(1, 2, \dots, \ell)$. Let I_ℓ be the $\ell \times \ell$ identity matrix. For some positive integers ℓ and m , define $x = S_\ell \otimes I_m$ and $y = I_\ell \otimes S_m$. Let A_1, A_2, A_3, B_1, B_2 , and B_3 each be a monomial in either x or y . Then $A = A_1 + A_2 + A_3$ and $B = B_1 + B_2 + B_3$ are polynomials in x and y . The BB code defined by the quadruple (ℓ, m, A, B) is the CSS code with X and Z check matrices $H^X = [A|B]$ and $H^Z = [B^T|A^T]$.

Let $x = S_\ell \otimes I_m$, $y = I_\ell \otimes S_m$, $x' = S_\ell \otimes I_{tm}$, and $y' = I_\ell \otimes S_{tm}$. Note $x, y \in \mathbb{F}_2^{\ell m \times \ell m}$ and $x', y' \in \mathbb{F}_2^{t\ell m \times t\ell m}$. For some integer exponent p , $x^p = S_\ell^p \otimes I_m$ and $y^p = I_\ell \otimes S_m^p$. Similarly, $(x')^p = S_\ell^p \otimes I_{tm}$ and $(y')^p = I_\ell \otimes S_{tm}^p$.

Our first goal is to construct a matrix $\widetilde{(x^p)} \in \mathbb{F}_2^{\ell m \times t\ell m}$ from x^p s.t. every row of $\widetilde{(x^p)}$ also occurs as a row of $(x')^p$. For every element $(x^p)_{ij}$, set $\widetilde{(x^p)}_{i,tj} = (x^p)_{ij}$. All other elements of $\widetilde{(x^p)}$ are 0. This will be called the

“padding” of the matrix. By the definition of the tensor product,

$$x^p = S_\ell^p \otimes I_m = \begin{bmatrix} [S_\ell^p]_{11} \mathbb{I}_m & \cdots & [S_\ell^p]_{1\ell} \mathbb{I}_m \\ \vdots & \ddots & \vdots \\ [S_\ell^p]_{\ell 1} \mathbb{I}_m & \cdots & [S_\ell^p]_{\ell\ell} \mathbb{I}_m \end{bmatrix} \quad (\text{C1})$$

and

$$(x')^p = S_\ell^p \otimes I_{tm} = \begin{bmatrix} [S_\ell^p]_{11} \mathbb{I}_{tm} & \cdots & [S_\ell^p]_{1\ell} \mathbb{I}_{tm} \\ \vdots & \ddots & \vdots \\ [S_\ell^p]_{\ell 1} \mathbb{I}_{tm} & \cdots & [S_\ell^p]_{\ell\ell} \mathbb{I}_{tm} \end{bmatrix}. \quad (\text{C2})$$

Since all rows and columns of S_ℓ^p have Hamming weight 1, all rows and columns of x^p and $(x')^p$ also have Hamming weight one. By construction, all rows of $\widetilde{(x^p)}$ have Hamming weight one and all columns have Hamming weight either one or zero. The rows of all of these matrices are unique. For a row of (x^p) to equal a row of $(x')^p$, it suffices to check that they have their 1 in the same column. If $[S_\ell^p]_{ij} = 1$, then for any $k = 1, \dots, m$,

$$[x^p]_{(i-1)m+k, (j-1)m+k} = 1 \quad (\text{C3})$$

and

$$[\widetilde{(x^p)}]_{(i-1)m+k, t((j-1)m+k)} = 1. \quad (\text{C4})$$

For any $k = 1, \dots, tm$,

$$[(x')^p]_{(i-1)tm+k, (j-1)tm+k} = 1. \quad (\text{C5})$$

For some k between 1 and m , the row of $\widetilde{(x^p)}$ with a 1 in column $t((j-1)m+k)$ is equal to a row of $(x')^p$ with column index $(j-1)tm+k'$ for some k' with $1 \leq k' \leq tm$. Equating the two column indices, $k' = tk$. Thus the row of $\widetilde{(x^p)}$ with index $(i-1)m+k$ is equal to the row of $(x')^p$ with index $(i-1)tm+k' = t((i-1)m+k)$. In words, for any row i of $\widetilde{(x^p)}$, row ti of $(x')^p$ is equal to it.

Next construct $\widetilde{(y^p)}$ from y^p in the same manner as for $\widetilde{(x^p)}$. It must be shown that every row of $\widetilde{(y^p)}$ is also a row of $(y')^{tp}$. By definition,

$$y^p = I_\ell \otimes S_m^p = \begin{bmatrix} S_m^p & & \\ & \ddots & \\ & & S_m^p \end{bmatrix} \quad (\text{C6})$$

and

$$(y')^{tp} = I_\ell \otimes S_{tm}^{tp} = \begin{bmatrix} S_{tm}^{tp} & & \\ & \ddots & \\ & & S_{tm}^{tp} \end{bmatrix}. \quad (\text{C7})$$

Suppose $[y^p]_{ij} = 1$. Then $[\widetilde{(y^p)}]_{i,tj} = 1$. By the definition of the shift matrices, $i = j - p \bmod \ell$. If $[(y')^{tp}]_{i',j'} = 1$, then similarly $i' = j' - p \bmod \ell$. Setting $j' = tj$, $i' =$

$tj - p \bmod \ell$ and $i' = ti$. Thus the i^{th} rows of $\widetilde{(x^p)}$ and $\widetilde{(y^p)}$ equal the ti^{th} rows of $(x')^p$ and $(y')^{tp}$.

Let $A(x, y) = x^{p_1}y^{q_1} + x^{p_2}y^{q_2} + x^{p_3}y^{q_3}$. The terms in A must be monomials, so $q_i = 0$ if $p_i \neq 0$ and vice versa. Let $B(x, y) = x^{r_1}y^{s_1} + x^{r_2}y^{s_2} + x^{r_3}y^{s_3}$. For every monomial $x^{p_i}y^{q_i}$ in A , the i^{th} row of padding of the monomial corresponds to the ti^{th} row of the monomial $(x')^{p_i}(y')^{q_i}$ in A' . The same condition applies to B and B' . Since this correspondence does not depend on the exponents p_i or q_i (r_i or s_i), the corresponding rows between the two matrices can be summed over all three monomials and the i^{th} row of A (B) corresponds to the ti^{th} row of A' (B'). The X -type check matrices of C and C' are $H^X = [A|B]$ and $(H^X)' = [A'|B']$, respectively. The i^{th} row of H^X corresponds to the ti^{th} row of $(H^X)'$.

Now consider the Z -type check matrix $H^Z = [B^T|A^T]$ for the original BB code along with its counterpart $(H^Z)' = [(B')^T|(A')^T]$ for the larger BB code. We must show that for every row of B^T (A^T) its padding is also a row of $(B')^T$ ($(A')^T$).

Let $\widetilde{((x^p)^T)}$ be the padding of $(x^p)^T$. Every row and column of $(x^p)^T$ has weight one, as do the rows and columns of $\widetilde{((x^p)^T)}$. Since the rows and columns of $(x^p)^T$ have weight one, so do the rows and columns of $\widetilde{((x^p)^T)}$. Analogous to Eqs. (C3) through (C5), for $k = 1, \dots, m$

$$[(x^p)^T]_{(j-1)m+k, (i-1)m+k} = 1 \quad (\text{C8})$$

and

$$[\widetilde{((x^p)^T)}]_{(j-1)m+k, t((i-1)m+k)} = 1. \quad (\text{C9})$$

For any $k = 1, \dots, tm$,

$$[(x')^p]_{(j-1)tm+k, (i-1)tm+k} = 1. \quad (\text{C10})$$

Since the rows and columns of $\widetilde{((x^p)^T)}$ have weight one as do the rows and columns of $((x')^p)^T$, a row of $\widetilde{((x^p)^T)}$ is equal to a row of $((x')^p)^T$ if they have a one in the same location. For any $k = 1, \dots, m$ there must exist some $k' = 1, \dots, tm$ s.t.

$$t((i-1)m+k) = (i-1)tm+k',$$

which is satisfied by $k' = tk$. Thus for every row of $(x^p)^T$, its padding is also a row of $((x')^p)^T$.

Let $\widetilde{((y^p)^T)}$ be the padding of $(y^p)^T$. Recall that $(y^p)^T = I_\ell \otimes (S_m^p)^T$ and $((y')^{tp})^T = I_\ell \otimes (S_{tm}^{tp})^T$. Written out explicitly,

$$(y^p)^T = I_\ell \otimes S_m^p = \begin{bmatrix} (S_m^p)^T & & \\ & \ddots & \\ & & (S_m^p)^T \end{bmatrix} \quad (\text{C11})$$

and

$$(y')^{tp} = I_\ell \otimes S_{tm}^{tp} = \begin{bmatrix} (S_{tm}^{tp})^T & & \\ & \ddots & \\ & & (S_{tm}^{tp})^T \end{bmatrix}. \quad (\text{C12})$$

Suppose $[(y^p)^T]_{ij} = 1$. Then $[(\widetilde{(y^p)^T})]_{i,tj} = 1$. By the definition of the shift matrices and the transpose that was performed, $j = i - p \bmod \ell$. If for some (i', j') $[(y')^{tp}]_{i',j'} = 1$, then similarly $j' = j' - p \bmod \ell$. Setting $j' = tj$, $i' = tj - p \bmod \ell$ and $i' = ti$. Thus the i^{th} rows of $((x^p)^T)$ and $((y^p)^T)$ equal the ti^{th} rows of $((x')^p)^T$ and $((y')^{tp})^T$. Since the transpose in A^T and B^T distributes over the monomial terms, the rest of the proof proceeds as for H^X above, and for every row of H^Z we find that its padding is also a row of $(H^Z)'$.

This construction requires that columns and rows of the original check matrix become spaced out by an interval t when going from H^X to $(H^X)'$. To restore the standard form of the check matrix for an extended code, permute the rows and columns of $(H^X)'$ s.t. H^X is in its upper-left block. Permuting the rows does nothing to the code, and permuting the columns (*i.e.* physical qubits) does not affect the distance of the code.

Appendix D: More Details on the SAT–UNSAT Phase Transition

To characterize the finite-size SAT–UNSAT transition systematically, we generate random k -CODE DESIGN instances for fixed (n, k, w) , where w is the maximum Pauli weight included in the error pool, and vary the number of errors samples, m , from this pool. For each parameter point, we generate 100 independent instances and estimate the satisfiability probability P_{SAT} . Figure 9 illustrates the relation between the resulting SAT instances, their clause density, and the natural code-design density $m/(n - k)$ for $n = 4$, $k = 1$, and $w = 3$. The dashed line at $\alpha \simeq 4.267$ denotes the critical clause density of random 3-SAT and is included only as a reference, since the formulas generated by k -CODE DESIGN are structured mixed 2–3-SAT instances.

Figure 10 shows the same transition over a broader set of parameters for $n = 4$, with rows corresponding to $k = 1, 2, 3$ and columns to $w = 1, 2, 3, 4$. Figure 10(d) is reported in the main text as Fig. 3 to demonstrate the spike in runtime complexity at the P_{SAT} phase transition. The blue curves report the satisfiability probability P_{SAT} , while the purple curves show the median SAT-solver runtime and the shaded region illustrates the 90% confidence interval. The location of the transition varies appreciably with (n, k, w) , demonstrating that $m/(n - k)$ alone does not define a universal critical density.

It is clear from Fig. 10, that while illustrative, the errors per stabilizer parameter $[m/(n - k)]$ does not describe the order parameter since the critical value increases with increasing $n - k$. Therefore, we extract a parameter-dependent critical point $m_c(n, k, w)$ by fitting each transition to

$$P_{\text{SAT}}(m) = \frac{1}{1 + \exp\left(\frac{m - m_c}{a}\right)}, \quad (\text{D1})$$

where m_c is the 50% satisfiability crossing and a determines the transition width, as illustrated in Fig. 11. Of the 131 sampled (n, k, w) curves, 46 include the SAT–UNSAT transition within the sampled range and are retained for the scaling analysis. Defining the number of stabilizers $r = n - k$ and the number of Pauli errors of weight at most w as

$$M_{\leq w}(n) = \sum_{j=1}^w 3^j \binom{n}{j}, \quad (\text{D2})$$

a model comparison with the Bayesian information criterion (BIC) favors the empirical scaling form

$$m_c = c_0 + An^a r^b [\log M_{\leq w}(n)]^c k^d w^e. \quad (\text{D3})$$

After model selection and refitting to all identifiable critical points, we obtain the fit parameters: $c_0 = 3.29$, $a = 1.01$, $b = 1.78$, $c = 1.85$, $d = -0.34$, and $e = -0.44$. Note that the scaling law indicates a strong dependence on both r and the number of Pauli errors.

The fit is performed in conjunction with cross-validation procedures. In particular, the critical m_c dataset is split into training and validation sets. A fit using only the training points gives a held-out RMSE of 0.096 in $\log m_c$, corresponding to an RMS relative error of approximately 9.7%, while five-fold cross-validation gives 0.105 ± 0.040 in $\log m_c$. Figure 12 compares the predicted and fitted critical points, and Fig. 13 shows the resulting collapse of the transition under the rescaled order parameter $m/\widehat{m}_c(n, k, w)$. We emphasize that this relation is an empirical finite-size scaling ansatz over the sampled range rather than an asymptotic scaling law.

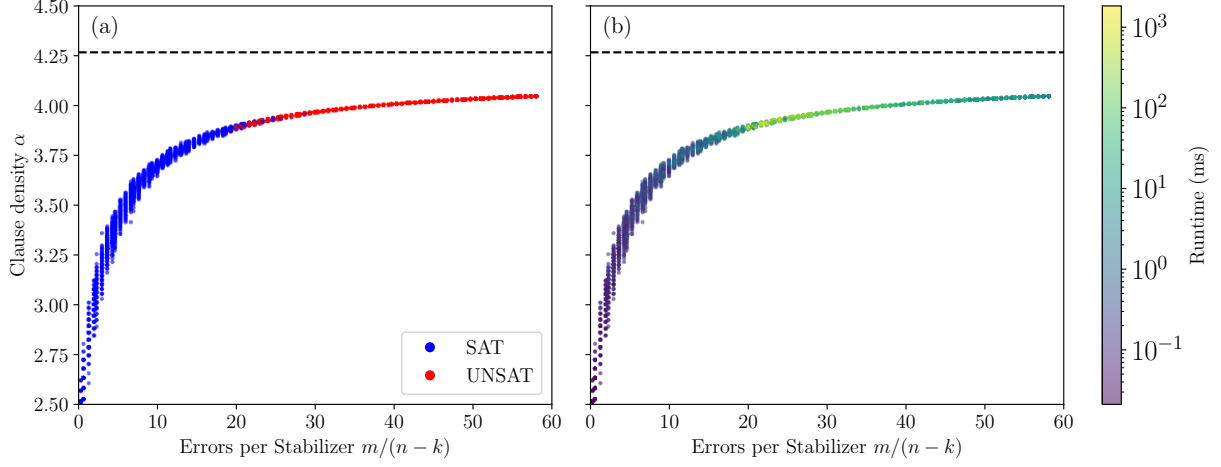


Figure 9. Relationship between the natural code-design density $m/(n-k)$ and the clause density α of the corresponding SAT instance for $n = 4$, $k = 1$, and $w = 3$. The panel (a) indicates whether each random instance is satisfiable (SAT) or unsatisfiable (UNSAT), while the panel (b) shows the corresponding SAT-solver runtime. The horizontal dashed line at $\alpha \approx 4.267$ denotes the critical clause density of random 3-SAT given as a reference, since the code-design instances produce structured mixed 2-3-SAT formulas.

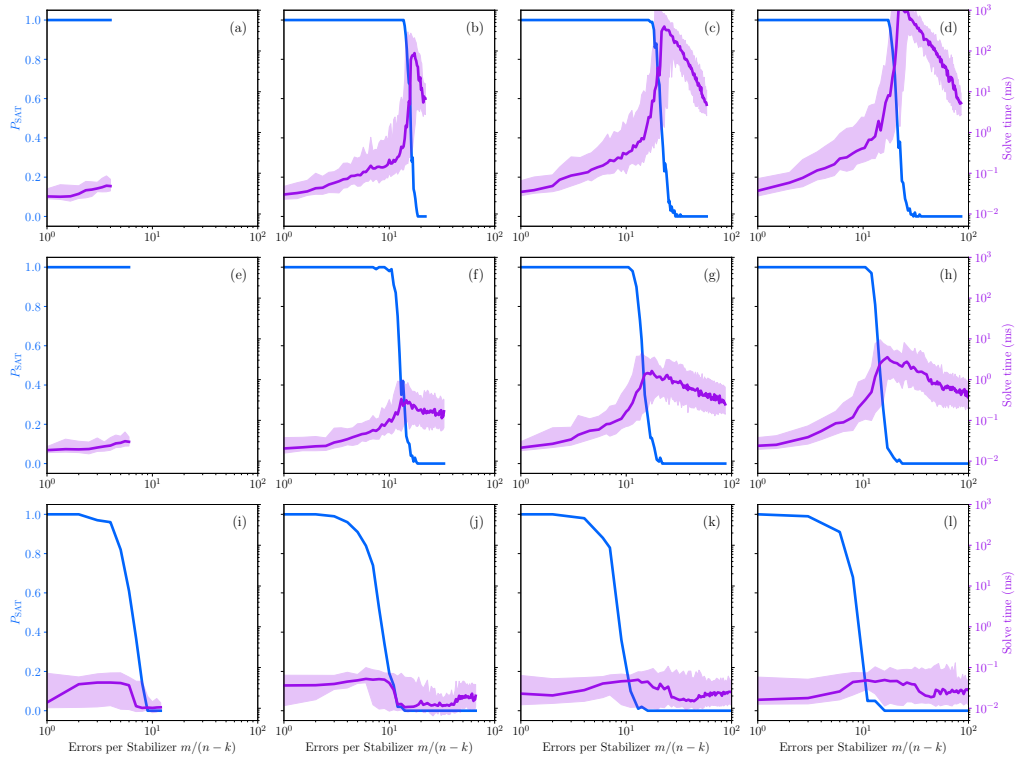


Figure 10. Finite-size SAT-UNSAT transitions for random code-design instances with $n = 4$. Rows correspond to increasing numbers of logical qubits k , and columns correspond to increasing maximum error weights w . The blue curves show the empirical satisfiability probability P_{SAT} as a function of the code-design density $m/(n-k)$. The purple curves show the median SAT-solver runtime, with the shaded region indicating the 90% confidence interval. The characteristic increase in runtime near the SAT-UNSAT crossover illustrates the easy-hard-easy behavior associated with NP-hard problems.

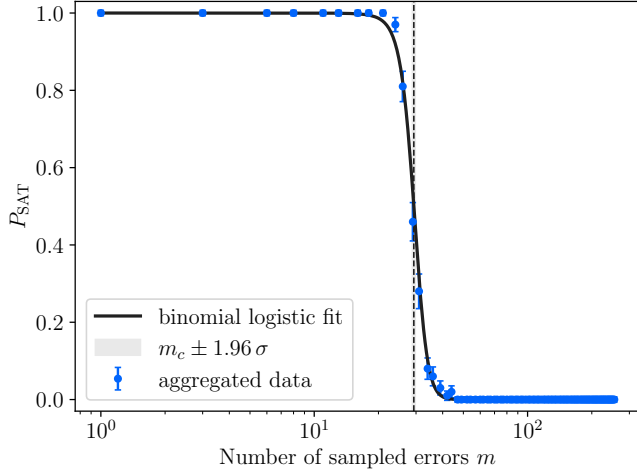


Figure 11. Example of random code-design instances fitted to extract the critical m_c of the finite-size SAT–UNSAT transition. The empirical satisfiability probability $P_{\text{SAT}}(m)$ is fit to a logistic function, with m_c defined by the $P_{\text{SAT}} = 1/2$ crossing. The fitted width parameter characterizes the sharpness of the finite-size transition.

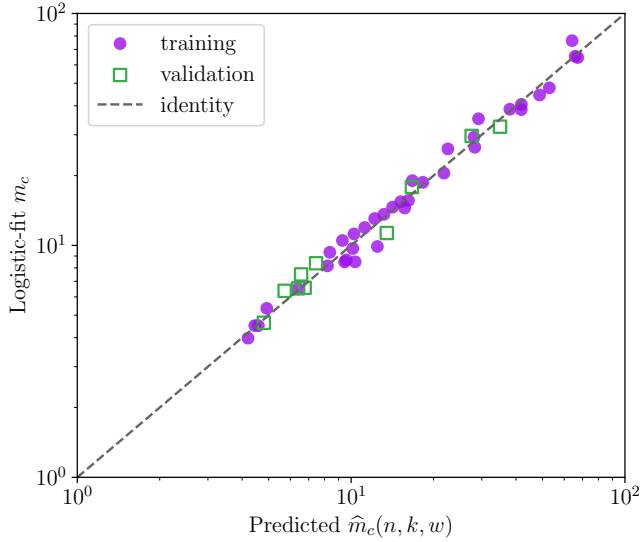


Figure 12. Comparison of the observed critical m_c , obtained from individual logistic fits, with the values predicted by the empirical finite-size scaling model. Agreement with the dashed $y = x$ line indicates accurate prediction of the location of the SAT–UNSAT crossover across different (n, k, w) for training and validation sets.

Appendix E: Additional Numerical Results and Discussion for Physics-Informed Code Design

With the simple noise model detailed in Section VIII, the result for the distance-3 code is almost perfect (see Fig. 5). Fig. 14 shows a similar simulation under a circuit-level noise model, where physical qubits experience noise after each physical gate in the gadgets imple-

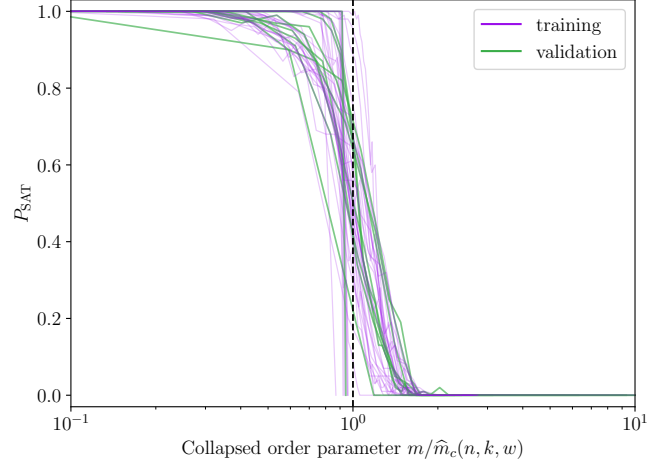


Figure 13. Collapse of the satisfiability transitions under rescaling by the predicted critical point $\hat{m}_c(n, k, w)$. The collapse demonstrates that the empirical scaling model captures much of the parameter dependence of the finite-size transition.

menting the logical circuits. While the improvement in expectation values in this case is, as expected, less emphatic than in Fig. 5, we still see that the extended codes perform better even when state preparation is noisy.

Because the SAT formulation only guarantees that errors are detectable, with few other constraints on the code, there is no guarantee that the logical operations of the resulting codes will have fault-tolerant implementations. This is the source of the failure of the code extension seen in Fig. 14. Extending a quantum code in such a way that all fault-tolerant operations remain fault-tolerant in the extended code will be the subject of future work. Indeed, there must be some sequences of extended codes at successively greater distances where operations remain transversal.

On fault-tolerant quantum hardware, QEC codes (*e.g.* the surface code or QLDPC codes) will be implemented in order to perform computations with logical qubits. Even with very performant decoders, logical errors can still occur during the course of a computation. At the cost of a few extra logical qubits, our construction could be used to detect these logical errors in addition to detecting violations in the symmetries of the physical system (or other problem) being studied. Under these conditions, logical errors might be sufficiently rare that there is an advantage to employing the codes constructed here in order to enhance existing symmetry verification procedures with the ability to detect logical errors.

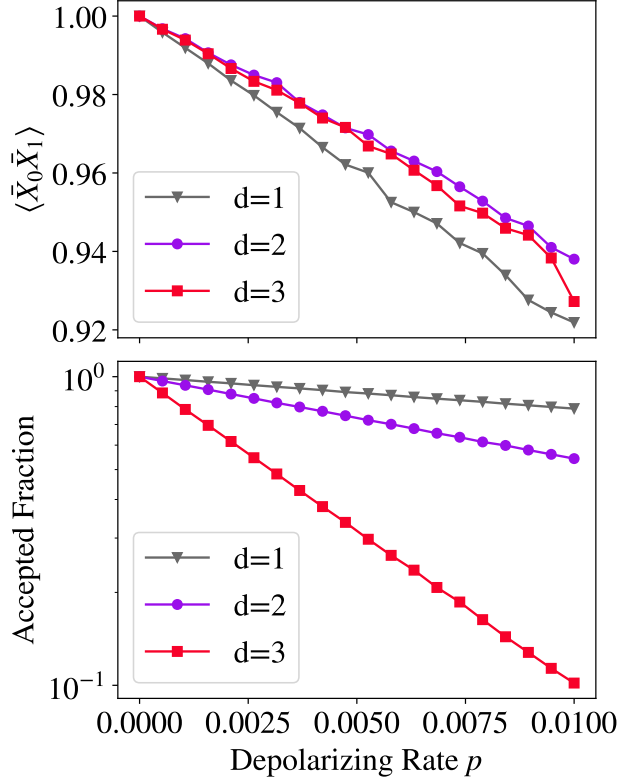


Figure 14. (top) Expectation value of logical observable $\bar{X}_0\bar{X}_1$ in a logical Bell state vs. the depolarizing noise rate p acting on the qubits for Fermi-Hubbard codes with distances 1, 2, and 3. The qubits are subject to circuit-level noise during the execution of each logical gate. (bottom) The fraction of shots that are accepted at each code distance vs. the depolarizing noise rate p . Points with no accepted shots are not shown.

To evaluate the performance of code extension, we consider the problem of starting with the original FH code with generators given in Eq. (46) and extending the code to have distance three. The original code has stabilizer generators that consist of only Pauli Z operators, and so only bit flip errors can be detected. A distance-three extension is capable of detecting any Pauli operators up to weight three. Fig. 15 shows the time needed to solve this extension problem (or prove it is unsatisfiable) when adding m new stabilizers to an n -qubit FH code. The black line shows the minimum value of m that is necessary to increase the code to distance three. Many instances are seen to be easy, completing in around a minute. The hard instances, which take several minutes, lie along the satisfiable/unsatisfiable boundary for $n \geq 15$. This spike in the solution time is reminiscent of the phase transition seen in Sec. VII: when m is small, the problem is easily proven unsatisfiable and when m is large there are many degrees of freedom with which to create a code and the problem is easily satisfiable. The problems along the satisfiable/unsatisfiable boundary are the ones where the runtime required by the SAT solver may significantly increase. Similar results for larger systems can be seen in Fig. 6 of the main text.

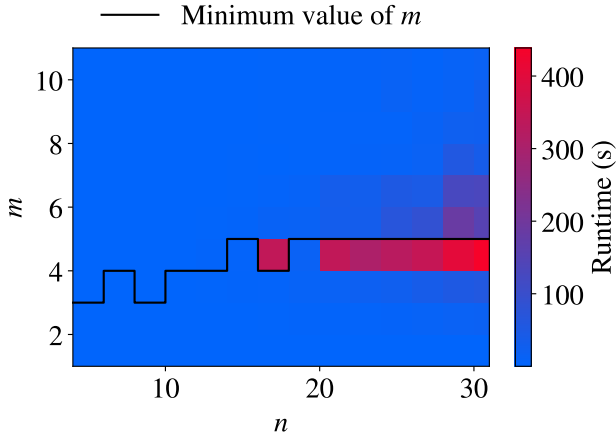


Figure 15. Time to use SAT to increase the FH code from distance 1 to distance 3 as a function of the number of qubits and stabilizers added to the code (m) and the number of qubits in the original system (n). The runtime indicates the time to either prove the problem is unsatisfiable for the given m or find a distance-three extension. The black line shows the minimal value m above which the problem is satisfiable.