Hypergraph product codes: a bridge to scalable quantum computers

Armanda O. Quintavalle

Submitted for the degree of Doctor of Philosophy
October 2022
Department of Physics and Astronomy
The University of Sheffield

How to build a bridge

A bridge needs good foundations. Mine had Earl Campbell. Earl taught me how to *do science*. Blame me for whatever is not science in this thesis. Not him. Thank you for all the afternoons at the whiteboard and the occasional tequila.

This bridge also had Pieter Kok. A pro-bono mentor and on-paper supervisor. His optimism has helped both physics and life.

Joshka Roffe has been 24/7 emergency rescue for any build-related issues. While I learnt a good chunk of quantum information stuff from him, he has hopefully learnt to tolerate a pedantic mathematician.

Elena Callus has been on call too. She is a brilliant physicist who taught me that my favourite qubit is the logical qubit, and hence forever saved me from physics.

A good bit of this work has been done thanks to the fine architect advice of Mike Vasmer - we had fun doing that.

A bridge does not only have towers but arches too. They are strong and help control the load. Yingkai Ouyang, Paul Webster, Zixin Huang, and, of course, the rest of the Sidney crew - Felix Thomsen and Mark Webster. Arthur Pesah, aka the 3D guy, a great QEC person and mindful promoter of this bridge. The unlucky experimentalist and inhabitants of E13a who bared with many disparate and desperate questions.

17 November 2022
Thank you to Dan Browne and David Whittaker, who took the time to discuss this work with me. Thank you to Dan Browne and David Whittaker, who took the time to discuss this work with me. I gained a lot of insights from our conversation.

Abstract

A physical machine for storage and manipulation of information, being physical, will always be subject to noise and failure. For this reason, the design of fault-tolerant architectures is of prime importance for building a working quantum computer. Quantum error correction codes offer a possible elegant framework for fault-tolerance when provided with methods to operate qubits without corrupting the information stored therein. This work specialises in hypergraph product (HGP) codes and seeks to lay the groundwork for a quantum computer architecture based on them.

The leading approach to fault-tolerant quantum computation is, today, based on the planar code. A planar-code-based quantum computer, however, would require dramatic qubit overhead and we believe that *good* low-density parity-check (LDPC) codes are necessary to attain the full potential of quantum computing. The HGP codes, of which the planar code is an instance, are not, strictly speaking, good LDPC codes. Still, they are an efficient alternative. On the one hand, the best HGP codes improve upon the planar code as they can store multiple logical qubits. On the other, they are not considered good because their noise robustness is sub-optimal. Nonetheless, we see the design of a HGP-based quantum computer as a bridge between the currently-favoured planar code design and the gold standard of good LDPC codes. A HGP-based architecture would inform our knowledge on how to design fault-tolerant protocols when a code stores multiple logical qubits, which is, to a large extent, still an open question.

Our first original contribution is a decoding algorithm for all families of two-fold HGP codes. Second, we exhibit a constructive method to implement some logical encoded operations, given HGP codes with particular symmetries. Last, we propose the concept of confinement as an essential characteristic for a code family to be robust against syndrome measurement errors. Importantly, we show that both expander and three-dimensional HGP codes have the desired confinement property.

Contents

Acronyms

CSS code	Calderbank-Shor-Steane code	12
LDPC	low-density parity-check	8
ML decoder	maximum-likelihood decoder	9
qLDPC	quantum low-density parity-check	10
qML decoder	quantum maximum-likelihood decoder	11
qMW decoder	quantum minimum-weight decoder	11

Chapter 1

Context and notation - Introduction

Reliable exchange of information needs redundancy. Natural language is redundant and in fact, we could write a sentence, remove all the vowels –'th dg s n th grdn' – or make a few typos – 'teh dog is in the graden' – but still have a good chance that our message goes through. These are just two sides of the same coin: we can encode information to compress it (source coding) or we can encode information to protect it from errors (channel coding). Here we deal with the latter when the information is processed by a quantum computer. Before turning to the quantum side, we briefly go over the key features of channel coding for classical information processing in Section 1.1. We introduce the corresponding concepts for quantum information processing in the remaining Sections 1.2 to 1.6.

1.1 Bits

The basic unit of information is the bit, a two-state system, $\mathbb{F}_2$, whose possible values are $\{0, 1\}$ [1, 2]. If a bit b has an equal probability of being in each state, the amount of information content in b is 1 shannon. Given a bit b in an unknown state, we can flip its value:

$$\begin{aligned} \text{flip}(b) &= \begin{cases} 0 & \text{if } b = 1 \\ 1 & \text{if } b = 0 \end{cases}, \\ &= \quad b + 1 \text{ in } \mathbb{F}_2. \end{aligned}$$

The flip operation maps valid states of b into valid ones: it is a *logical* operation on the bit state. In contrast, the operation $\text{vespa}(b) =$ 🛵 is not a logical operation since the symbol 🛵 does not belong to $\mathbb{F}_2$ and it is therefore not a valid state for the bit b.

Because a bit only has two possible valid states, the flip is the only reversible error a bit is subject to. Nonetheless, if b undergoes a flip error with some probability $p \in (0, 1)$, there is no way to infer whether b has been erroneously flipped or not. Being a logical operation, the flip preserves valid states and thus cannot be *detected*. With only one bit at our disposal, the knowledge that an error has occurred with probability p is not enough: we need to add redundancy.

One way of adding redundancy is to use the classical repetition code. It encodes the *logical* bit b into a register of $n > 1$ *physical* bits. For $n = 3$, the bit b is mapped into $c = (c_1\, c_2\, c_3) \in \mathbb{F}_2^3$ via the

linear map:

$$\begin{aligned} G: \quad \mathbb{F}_2 &\implies \mathbb{F}_2^3 \\ b &\longmapsto \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot b. \end{aligned} \tag{1.1}$$

The valid states for the register c are the ones in the image[1] of G. The image of G is

$$\mathrm{im}(G) = \{(0\,0\,0), (1\,1\,1)\}.$$

We call the vector space $\mathrm{im}(G)$ the codespace. The elements of the codespace are the codewords. The logical flip operation on the codespace requires three single bit-flips:

$$\mathrm{flip}(c) = \begin{pmatrix} \mathrm{flip}(c_1) \\ \mathrm{flip}(c_2) \\ \mathrm{flip}(c_3) \end{pmatrix} = c + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in \mathbb{F}_2^3.$$

One flip was previously enough to corrupt the information stored in b, but as many as three physical flips are now necessary to corrupt an encoded state in $\mathrm{im}(G)$ and for the event to go undetected. If the register c is found in a state with $c_i \neq c_j$ for some $i \neq j$ then at least an error has occurred: we have detected that c belongs to the complement[2] $\mathbb{F}_2^3 \setminus \mathrm{im}(G)$ of the codespace. Crucially, the codespace $\mathrm{im}(G)$ is isomorphic to $\mathbb{F}_2$, the vector space describing the state space of a single bit. Despite that, the logical flip in the codespace 'costs' three times the cost of a single bit-flip. We have added redundancy and lowered the probability of undetectable errors – from p to p^3 in this example.

The repetition code offers a working, albeit simple, example of classical code. However, its construction does not carry over to qubits as we illustrate in the following Sections. In Section 1.2, we review some basic features of qubit systems. Our presentation, far from exhaustive, aims to highlight the primary issues at the core of robust quantum information processing. The quantum repetition code is presented in Section 1.2.1, and its shortcomings are pointed out. In Section 1.3 we briefly run through classical linear codes as it is needed to understand their quantum counterpart, the stabiliser codes. We outline the construction of the planar code – the quantum working equivalent of the repetition code – in Section 1.4. We conclude this Chapter with Sections 1.5 and 1.6, where we discuss two concurrent issues to quantum code construction for the design of a fault-tolerant computer architecture: logical operations and noisy measurements.

1.2 Qubits

A qubit is a two-dimensional Hilbert space, $\mathbb{C}^2$, whose state is described by a unit vector [3, 4]. We write $|0\rangle$ and $|1\rangle$ to indicate the standard basis of $\mathbb{C}^2$, so that the state of a qubit q is written as $|\varphi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where the vector of coefficients $(\alpha, \beta)^T \in \mathbb{C}^2$ has unit norm, $|\alpha|^2 + |\beta|^2 = 1$.

[1] The image of a function $f : A \longrightarrow B$, denoted $\mathrm{im}(f)$, is the set of all elements $f(a) \in B$, as a varies in the domain A of f.

[2] Given two sets A and B, the complement of B in A is the set of all elements of A that do not belong to B. The complement is written $A \setminus B$.

The possible reversible operations on a qubit are unitary maps: $U \in \mathbb{C}^{2\times 2}$ such that $UU^\dagger = \mathbb{1}$. An extensively used basis for the linear operators in $\mathbb{C}^{2\times 2}$ is the Pauli basis:

$$\mathbb{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The Pauli operators X, Y, Z, together with $\pm i\mathbb{1}$, generate a multiplicative group which obeys the product and commutativity relations:

$$X^2 = Y^2 = Z^2 = \mathbb{1}, \quad XY = iZ, \quad YZ = iX, \quad ZX = iY,$$

and

$$XY = -YX, \quad YZ = -YZ, \quad ZX = -XZ.$$

An n-qubit register has states that are vectors in $(\mathbb{C}^2)^{\otimes n}$ with standard basis $\{|v\rangle \text{ s.t } v \in \mathbb{F}_2^n\}$. The Pauli group on n qubits, $\mathcal{P}_n$, is the group generated by n-fold tensor products of Pauli operators. Given a Pauli operator $P \in \mathcal{P}_1 \subseteq \mathbb{C}^{2\times 2}$, we write P_i for the operator in $\mathcal{P}_n \subseteq \mathbb{C}^{2^n \times 2^n}$ which is the tensor product of P at position i and the identity elsewhere. We define the support of a Pauli operator $\mathcal{P}_n$ as the set:

$$\text{supp}(P) = \{i \text{ such that } P = P_1 \otimes \ldots \otimes P_n \text{ and } P_i \neq \mathbb{1}\},$$

and its weight $|P|$ as $|P| = |\text{supp}(P)|$. On $(\mathbb{C}^2)^{\otimes n}$, the Pauli basis is the set:

$$\{\mathbb{1}_n, X_i, Z_i, Y_i \quad \text{s.t. } 1 \leq i \leq n\},$$

where $\mathbb{1}_n \in (\mathbb{C})^{2^n \times 2^n}$ is the identity matrix of size $2^n \times 2^n$. Similarly to the one-qubit Pauli basis operators in $\mathcal{P}_1$, Pauli basis operators in $\mathcal{P}_n$ square to the identity and any two of them either commute or anti-commute.

Information on the qubits' state is acquired via the measurement of observables, which are Hermitian operators $A = A^\dagger \in \mathbb{C}^{2^n \times 2^n}$. Measuring an observable collapses the state of the qubits onto one of the eigenspaces of the measured observable. The possible outcomes are the eigenvalues of the observable, observed with probability equal to the squared norm of the inner product[3] between the pre-measurement state of the qubits and the corresponding eigenvector. Note that outcomes and post-measurement states for an observable are well defined. By the Spectral Theorem, every Hermitian operator has an orthonormal basis of eigenvectors [5]. Furthermore, if A and B are both Hermitian and commute, they have a common basis of eigenvectors and hence the measurement of one does not alter the probability distribution of the outcome of the other. Commuting observables are compatible.

Importantly, if we knew that a qubit register is in one of two orthogonal states, we could measure the corresponding observable – any Hermitian operator whose eigenvectors are the two orthogonal states – and distinguish between the two orthogonal options with certainty. However, if we do not have such a priori knowledge, measuring a qubit register would collapse its wave function and corrupt the information stored therein. A possible solution to gather information on the state of a register whilst preserving it is to 'write' that information on an auxiliary register of qubits, as we will now

[3]The complex inner product is a map $\langle \cdot, \cdot \rangle : \mathbb{C}^{\otimes n} \times \mathbb{C}^{\otimes n} \longrightarrow \mathbb{C}$ defined as $\langle v, w \rangle \coloneqq v^\dagger w$, where $v^\dagger$ is the conjugate transpose of v.

explain. We refer to the qubit register we want to gather information about as the *data* register, and to the auxiliary register as *auxilia* register[4]. Information on the data qubits can be transferred to the auxilia register via the application of some entangling unitary operators on the composite data-auxilia system. Via the appropriate choice of the entangling operation, we can assume that the desired information can be acquired from the destructive measurement results of the auxilia qubits. In the following presentation, we will shortly refer to measurement outcomes of data qubits. By this, we always imply, if not otherwise specified, that suitable auxilia qubits are freshly prepared for the task, appropriate entangling operations between data and auxilia qubits are performed, the auxilia qubits are measured and then reinitialized in a standard state, ready to be used again.

To model the manipulation of information stored in qubit registers, we use the quantum circuit model of computation [3]. The quantum circuit model is based on the assumption that computation on a qubit register can be reduced to:

(i) Register preparation in a initial state, conventionally $|0\rangle^{\otimes n}$.

(ii) Active computation via the application of unitary operators on the register for a finite amount of time.

(iii) Results acquisition via the measurement of qubits in the Pauli Z basis.

Some observations are immediate. First, any realization of a quantum computer, a physical machine that runs quantum circuits, will be able to perform only a finite set of operations in a finite number of time steps. Second, if any unitary is a possible operation, then it is also a possible error on the qubit register. Third, every component – state preparation, gates, measurements – is subject to noise.

The Solovay-Kitaev theorem [7, 3] resolves the first obstacle: even if unitaries on $(\mathbb{C}^2)^{\otimes n}$ are a continuous set, a finite number of gates is sufficient to efficiently implement any unitary to arbitrary precision. More precisely Solovay-Kitaev states the following.

> Consider a norm $\|\cdot\|$ over the unitaries on $\mathbb{C}^2$. Let $\mathcal{G}$ be a finite set of unitaries, closed under multiplicative inverse. Further assume that the multiplicative group $\langle\mathcal{G}\rangle$ generated by $\mathcal{G}$ is dense in the unitaries, so that for every unitary U and $\varepsilon > 0$ in $\mathbb{R}$, there exists $G \in \langle\mathcal{G}\rangle$ such that $\|U - G\| < \varepsilon$. Fix a precision $\varepsilon > 0$. Then every unitary can be approximated within distance ε via a sequence of at most n_ϵ gates in $\mathcal{G}$, where $n_\varepsilon \propto \log^c(\frac{1}{\varepsilon})$ and $1 < c < 4$.

We remark the importance of the poly-logarithmic scaling in the inverse-precision $\frac{1}{\varepsilon}$ in Solovay-Kitaev's result. In fact, assume we want to implement m arbitrary unitaries, in m separate timesteps, on a qubit and we only have access to the gates in $\mathcal{G}$. We aim at precision $\varepsilon > 0$. Roughly, we will have

[4]The most common term in the literature for *auxilia* qubits is *ancilla* qubits. However, the Latin term ancilla means maidservant and is therefore intrinsically misogynist. For this reason, the community started using alternative terms such as *measurement* or *syndrome* qubits. I believe that the term 'measurement qubits' is biased towards the circuit model of computation as opposed to the measurement-based model [6]; whilst the term 'syndrome qubits' fails to cover the different usages of ancilla qubits, which go beyond the syndrome extraction function. As such, I have looked for a gender-neutral term. *Auxilium* in Latin means help, aid. I warmly invite the reader to adopt the phrasing *auxilia qubits*, as I firmly believe that changing the words we use is the first step towards shaping our thought, raising awareness and overcoming (more or less) implicit biases.

to implement each of the m unitaries to precision m/ε. By the Solovay-Kitaev's theorem, the total time cost scales as $m \cdot \log^c(\frac{m}{\varepsilon})$, with a total scale factor only poly-logarithmic. Keeping time overhead under control is paramount for the realizations of faithful quantum computations and Solovay-Kitav's theorem ensures that this is possible for single-qubit unitary operations. Conveniently, it can be shown that single-qubit unitaries and controlled-NOT, CNOT, gates

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

between arbitrary pairs of qubits are universal – meaning that an arbitrary unitary on $(\mathbb{C}^2)^{\otimes n}$ can be written as a product of CNOTs and single-qubit unitaries [3]. From Solovay-Kitaev it follows that there exists finite sets that can efficiently approximate any unitary operation up to arbitrary precision. We call any such finite set *universal*. Infinitely many universal gate sets exist, our preferred one is the Clifford+T set. The Clifford group $\mathcal{C}_n$ on $(\mathbb{C}^2)^{\otimes n}$ is generated by the tensor products of Hadamard, H, phase gate, S, and CNOT:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad \text{CNOT}.$$

The T gate is the square root of the phase gate, $T = \sqrt{S} \notin \mathcal{C}_n$, explicitly:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}.$$

The T gate grants universality to the set Clifford+T and therefore we have solved the first issue: a quantum computer that implements Clifford+T gates is computationally universal.

Before turning to the second issue, namely that possible errors are a continuum, we remark some key feature of the Clifford group. The Clifford group is not universal because the set generated by $\{S, H\}$ is not dense in the set of single-qubit unitaries – whilst $\{T, H\}$ generates a dense set. Furthermore, as per the Gottesman-Knill theorem, a circuit made of state preparation, Clifford operations only and measurements, can be simulated in polynomial time on a probabilistic classical computer [8]. Conjugation via Clifford permutes the Pauli and, in fact, the Clifford group can be alternatively defined as:

$$\mathcal{C}_n = \{U \text{ unitary s.t. } UPU^\dagger \in \mathcal{P}_n \text{ for all } P \in \mathcal{P}_n\}.$$

Every unitary operator can be written as a linear combination of the Pauli and therefore any operator can be characterized, up to global phase, by tracking its action on the Pauli. Hoping that the quantum circuit model strictly contains the classical circuit model, Clifford and Pauli are not sufficient to fully express the quantum circuit model of computation – still, we can use them to define a self-contained theory of error correction, as the discussion below and Section 1.3 explain.

Since any unitary is a possible operation, it also constitutes a possible error. Conveniently though we can write a generic unitary U on $\mathbb{C}^2$ as:

$$U = a_{\mathbb{1}}\mathbb{1} + a_x X + a_z Z + a_y Y,$$

for some complex coefficients $a_{\mathbb{1}}, a_x, a_y, a_z$. Consider a n qubit register, $|\varphi, \alpha\rangle$ where $|\varphi\rangle \in \mathbb{C}^2$ represent the state of the data qubit and $|\alpha\rangle \in (\mathbb{C}^2)^{\otimes n-1}$ represent the state of a $n-1$ auxilia register. Let $P = \mathbb{1}, X_1, Z_1, Y_1$ be a Pauli operator with support on the data qubit and suppose there exists a unitary operator O on $(\mathbb{C}^2)^{\otimes n}$ such that, for some initial state $|\alpha_0\rangle$ of the auxilia register,

$$O\left(P\,|\varphi, \alpha_0\rangle\right) = |P\varphi, \alpha_P\rangle\,, \tag{1.2}$$

where $|\alpha_P\rangle = |\alpha_{\mathbb{1}}\rangle$, $|\alpha_{X_1}\rangle$, $|\alpha_{Z_1}\rangle$, $|\alpha_{Y_1}\rangle$ are orthogonal in $(\mathbb{C}^2)^{\otimes n-1}$. Since orthogonal states can be distinguished via appropriate measurements, we can identify in which of the product states on the right-hand side of Eq. (1.2) the data-auxilia qubit register is in. By linearity, if U acts on the data qubit, Eq. (1.2) yields:

$$O(U\,|\varphi, \alpha_0\rangle) = a_{\mathbb{1}}\,|\varphi, \alpha_{\mathbb{1}}\rangle + a_x\,|X_1\varphi, \alpha_{X_1}\rangle + a_z\,|Z_1\varphi, \alpha_{Z_1}\rangle + a_y\,|Y_1\varphi, \alpha_{Y_1}\rangle\,.$$

Provided that such a unitary operator O does exist, and upon appropriate measurement of the auxilia qubits, the post-measurement state of the n-qubit register is $|P\varphi, \alpha_P\rangle$ with probability $|a_p|^2$. The original state of the data qubit $|\varphi\rangle$ can thus be restored by applying the correction operator P to the register. Ergo, if we were able to detect and correct for Z and X errors we could correct for all errors – Y errors corresponding to the concurrent event of an X and a Z error. We refer to this phenomenon as *error discretization* [3].

To showcase error discretization, we have made a very special assumption: namely that, given a n qubit register, errors could occur only on the first qubit of the register. Nonetheless, our analysis serves well to demonstrate:

(i) Even if measurements destroy superposition, not every superposition stores quantum information: the state of the data qubit $|\varphi\rangle$ is preserved when measuring the auxilia register $|\alpha_P\rangle$.

(ii) Even if the amount of information gained in a measurement is finite, measurements can be used to correct a continuous set of possible errors.

Error discretization is a key idea in quantum error correction and it remains a faithful assumption when errors are local and occur independently on each qubit of a register i.e. *local stochastic noise model*, see Eq. (1.8) in Section 1.3. Under such a premise, we can model errors occurring at any stage of the circuit model of computation and therefore also solve the third of our issues. Specifically, each gate of a circuit could fail and propagate errors (circuit level noise) and measurements result could be faulty too (measurement errors). We refer the reader to Sections 1.5 to 1.6 and [3] for a more detailed discussion on these matters and conclude this Section by exhibiting a toy example of quantum code.

1.2.1 The quantum repetition code

Mirroring the classical repetition code, we encode the logical qubit q in a 3-qubit register q_r via the unitary maps that on the standard basis act as:

$$\begin{aligned} G_r : \mathbb{C}^2 &\longrightarrow (\mathbb{C}^2)^{\otimes 3} \\ |0\rangle &\longmapsto |000\rangle\,, \\ |1\rangle &\longmapsto |111\rangle\,. \end{aligned} \tag{1.3}$$

With such encoding, the generic state in the quantum repetition codespace $\mathrm{im}(G_r) \simeq \mathbb{C}^2$ spanned by $\{|000\rangle, |111\rangle\}$ is:

$$|\psi\rangle_r = \alpha\,|000\rangle + \beta\,|111\rangle\,. \tag{1.4}$$

A valid choice[5] for the Pauli operators on the codespace is described by the group homomorphism γ induced by:

$$\begin{aligned} \mathcal{N}_r &\xrightarrow{\gamma} \mathcal{P}_1 \\ Z_1Z_2 &\longmapsto \mathbb{1}, \\ Z_2Z_3 &\longmapsto \mathbb{1}, \\ X_1X_2X_3 &\longmapsto X, \\ Z_1 &\longmapsto Z. \end{aligned} \tag{1.5}$$

where $\mathcal{N}_r$ is the group generated by:

$$\{i\mathbb{1}, Z_1Z_2, Z_2Z_3, X_1X_2X_3, Z_1\}.$$

The homomorphism γ is well-defined: its kernel[6],

$$\mathcal{S}_r \coloneqq \ker(\gamma) = \langle\, Z_1Z_2, Z_2Z_3 \,\rangle,$$

is a normal subgroup of $\mathcal{N}_r$ and the pre-images[7] of X and Z anti-commute.

We have added degeneracy: all the elements in $\mathcal{S}_r$ act as the identity on the generic encoded state $|\psi\rangle_r \in (\mathbb{C}^2)^{\otimes 3}$ e.g. $Z_1Z_2\,|\psi\rangle_r = |\psi\rangle_r$. Accordingly, every Pauli in the pre-image $\gamma^{-1}(X)$ of X has the same action on the encoded state $|\psi\rangle_r$ as the Pauli X has on the generic one-qubit state $|\varphi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. Similarly for $\gamma^{-1}(Z)$. For example, the logical Pauli Z on $|\psi\rangle_r$ can equivalently be implemented as $Z_1\,|\psi\rangle_r$ or $Z_3\,|\psi\rangle_r$. So, if for some reason we do not have access to the first qubit of the register q_r, we could still perform a Z operation on the logical qubit of the quantum repetition code applying Z_3. Yet, degeneracy is not the same as redundancy. On the one hand, the Pauli X on q is mapped to a weight three Pauli operator on q_r. On the other, the minimum weight of an encoded Z is not increased. Because of this imbalance, the quantum repetition code is resilient to X errors but sensitive to Z errors, as we now explain.

The operators in $\mathcal{S}_r$ have trivial action on the codespace and any error-free state $|\psi\rangle_r$ is their common $+1$ eigenstate. If the data register q_3 yields the result -1 upon measurement of Z_1Z_2 or Z_2Z_3, then at least one error has occurred and the register q_3 is in a state in $\mathbb{C}^2 \setminus \mathrm{im}(G_r)$. However, measuring Z_1Z_2 and Z_2Z_3 detects Pauli X errors of weight at most two, but cannot discern any difference between Z errors and valid states in the codespace. Because of degeneracy, all Pauli Z operators in $\mathcal{P}_3$ of weight one or three act as the logical Z operator on the encoded state $|\psi\rangle_r$ and all the weight two have trivial action. In other words, the action of Z operators is either a valid logical operation or trivial and therefore undetectable. If errors on each qubit are independent and one error occurs with

[5]The more expert reader will recognise Z_1Z_2, Z_2Z_3 and $X_1X_2X_3$, Z_1 as the stabilisers and the logical Pauli operators of the quantum repetition code. We decided to postpone the introduction of such terminology to Section 1.3 to illustrate to the less expert reader the group-theoretic essence of code construction.

[6]The kernel of a map $f : A \longrightarrow B$, denoted $\ker(f)$, is the set of all elements $a \in A$ such that $f(a) = 0 \in B$.

[7]The pre-image of a map $f : A \to B$, denoted $f^{-1}(b)$ for $b \in B$, is the set of all elements $a \in A$ such that $f(a) = b$.

probability p, encoding via the quantum repetition code decreases the probability of undetectable X errors from p to p^3, but increases the probability of undetectable Z error from p to $3p(1-p)^2 + p^3$ – so we have worsened it by a leading factor of three in p.

Universal computation and discretization of errors are expressions of the dual nature of the Pauli operators since, being both unitary and Hermitian, they serve as operations as well as observables to be measured. In addition to this, Pauli operators can define a self-contained theory of error correction. The need to protect from both X- and Z-type errors, and to aggregate equivalent (i.e. degenerate) ones, shows that a simple translation of what has been done for bits is not enough to protect a logical qubit from noise. Yet, degeneracy and redundancy are fundamental attributes of robust qubit encoding – as Section 1.3 and Section 1.4 specify.

1.3 Classical linear codes and their quantum counterpart: stabiliser and CSS codes

A classical linear code is a vector subspace of $\mathbb{F}_2^n$ defined as the kernel of a linear map $\sigma : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$. We call the subspace codespace and its elements codewords. With this terminology, the defining features of the syndrome map is that it yields the all-zeroes vector only when it acts on a codeword. Given a matrix representation $H \in \mathbb{F}_2^{m\times n}$ of σ, we say that the associated code has length n, dimension k and distance d if:

$$k = \dim(\ker H) = n - \mathrm{rank}(H) \quad \text{and} \quad \min_{\substack{v\in\ker H,\\ v\neq 0}} |v| = d,$$

where $|v|$ is the Hamming weight of the binary vector v:

$$|v| = |\{v_i \neq 0, \text{ where } v = (v_1, \ldots, v_n) \in \mathbb{F}_2^n\}|.$$

We say that H is a parity-check matrix for the code and call its rows checks. A code is a (r, c) low-density parity-check (LDPC) code if the rows and columns of H have weight at most r and c respectively. We will shortly say that the code is (r, c)-LDPC and has parameters $[n, k, d]$.

For instance, a parity check matrix for the classical repetition code in Section 1.1 is:

$$H_3 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

The parity-check matrix H_3 defines a $(2, 2)$-LDPC code of parameters $[3, 1, 3]$. See Fig. 1.

Figure 1: The classical repetition code of parameters $[7, 1, 7]$. Bits are placed on edges and checks are on circles. Each circle/check evaluates that the two edges incident to it have the same parity. Wiggly lines represent flip errors and red circles highlight the checks whose incident edges have different parity i.e. red circles represent the support of the syndrome. We note how flip errors on consecutive edges yield a weight two syndrome, with checks located at the boundaries of the error.

Given a parity check matrix $H \in \mathbb{F}_2^{m\times n}$ of a $[n,k,d]$ code, any error $e \in \mathbb{F}_2^n$ of weight strictly less than d can be detected. Namely, by construction, $H(c+e) \neq 0 \in \mathbb{F}_2^m$ for any codeword $c \in \ker H$. In particular, the value of the syndrome map is determined by the error vector e and it is independent of the codeword c. The optimal decoder for a code of distance d corrects all errors of weight at most $t = \lfloor \frac{d-1}{2} \rfloor$ because for every $c \in \ker H$ the ball of radius t centred in it does not contain any other vectors in $\ker H$.

In the following, we consider a local stochastic noise model of parameter $p \in (0,1)$, where the probability of an error e to occur is:

$$\mathbb{P}(e) = p^{|e|}(1-p)^{n-|e|}. \tag{1.6}$$

Eq. (1.6) signifies that single bit-flips are independent and the probability of an error occurring depends only on its weight. Since $\mathbb{P}(e)$ is monotonically decreasing for noise rates $p < \frac{1}{2}$, minimum weight errors are the most likely in this regime. The maximum-likelihood decoder (ML decoder) for $p < 1/2$ and $\mathbb{P}(e)$ as in Eq. (1.6) is $\mathcal{D}$:

$$\begin{aligned} \mathcal{D} : \mathrm{im}(\sigma) &\longrightarrow \mathbb{F}_2^n, \\ s &\longmapsto \arg\max_e \mathbb{P}\left(e \,|\, \sigma(e) = s\right). \end{aligned} \tag{1.7}$$

The decoder $\mathcal{D}$ outputs the most likely error so that, for a bit register in the state $m \in \mathbb{F}_2^n$, the recovered state is $m + \mathcal{D}(\sigma(m))$. For example, the maximum likelihood error recovering scheme for the repetition code under local stochastic noise is based on a majority vote strategy: if for some $m \in \mathbb{F}_2^3$ (or $\mathbb{F}_2^n$ in the general case) a non-zero syndrome is detected, flipping the minority bit (or bits) restores the parity of the register. In general, assuming local stochastic noise, the ML decoder is optimal.

The quantum counterpart of linear codes are *stabiliser codes*, first introduced in [9]. A stabiliser code is a subspace of $(\mathbb{C}^2)^{\otimes n}$ defined as the $+1$ common eigenspace of the stabiliser group, an Abelian subgroup $\mathcal{S}$ of the Pauli operators in $\mathcal{P}_n$ such that $-\mathbb{1} \notin \mathcal{S}$. As for classical codes, we call this subspace codespace and its elements the codewords. We denote by $\mathcal{C}(\mathcal{S})$ the codespace defined by the stabiliser group $\mathcal{S}$,

$$\mathcal{C}(\mathcal{S}) = \{|\psi\rangle \text{ s.t. } S|\psi\rangle = |\psi\rangle \quad \forall S \in \mathcal{S}\}.$$

Concretely, the group $\mathcal{S}_r$ introduced in Section 1.2.1 is a stabiliser group and $\mathcal{C}(\mathcal{S}_r)$ is the subspace generated by $|000\rangle$ and $|111\rangle$ in $(\mathbb{C}^2)^{\otimes 3}$. Because $\mathrm{Tr}(X) = \mathrm{Tr}(Y) = \mathrm{Tr}(Z) = 0$, for $P = P_1 \otimes \dots P_n \in \mathcal{P}_n$, it holds:

$$\mathrm{Tr}(P) = \prod_i \mathrm{Tr}(P_i) = 0.$$

Combining $P^2 = \mathbb{1}$, $\mathrm{Tr}(P) = 0$ and the Spectral Theorem, we find that each Pauli $P \in \mathcal{P}_n$ has exactly 2^{n-1} independent $+1$ eigenvectors, 2^{n-1} independent -1 eigenvectors, and that the dimension of the common $+1$ eigenspace of m independent Pauli operators in $\mathcal{P}_n$ is $k = n - m$. For the quantum repetition code, $\{Z_1Z_2, Z_2Z_3\}$ is a minimal set of generators for the stabiliser group and $k = 1$.

Given a set $S_1, \dots, S_m \in \mathcal{S}$ of generators for the stabiliser group, they uniquely define a syndrome map:

$$\begin{aligned} \sigma : \mathcal{P}_n &\longrightarrow \mathbb{F}_2^m \\ E &\longmapsto (s_1, \dots, s_m), \end{aligned}$$

where $s_i = 0$ if and only if $ES_i = S_iE$. The kernel of the syndrome map is the normaliser of $\mathcal{S}$ in $\mathcal{P}_n$, denoted $\mathcal{N}(\mathcal{S})$. The normaliser is the set of Pauli operators that commutes with all the stabilisers. Because $\mathcal{S}$ is normal in $\mathcal{N}(\mathcal{S})$, the quotient[8]:

$$\mathcal{L} := {}^{\mathcal{N}(\mathcal{S})}\!/_{\mathcal{S}},$$

is well-defined. The Pauli subgroup $\mathcal{L}$ is referred to as the logical Pauli group. If the stabiliser group has m generators, $\mathcal{L}$ is isomorphic to $\mathcal{P}_k$, where $k = n - m$ and the codespace $\mathcal{C}(\mathcal{S})$ is isomorphic to $(\mathbb{C}^2)^{\otimes k}$. The normaliser for the quantum repetition code is the group $\mathcal{N}_r$ introduced in Eq. (1.5) and generated by $\{\mathbb{1}, Z_1Z_2, Z_2Z_3, X_1X_2X_3, Z_1\}$. With a little abuse of notation, we write $L \in \mathcal{L}$ to indicate any representative L in its equivalence class $[L]$. Explicitly:

$$L \in [L] = \{LS \text{ such that } S \in \mathcal{S}\}.$$

For each $[L] \neq [\mathbb{1}]$ in $\mathcal{L}$, there exists by construction at least one state $|\psi\rangle \in \mathcal{C}(\mathcal{S})$ such that $L|\psi\rangle \neq |\psi\rangle$. Most importantly, elements in the same equivalence class in the logical Pauli group yield the same action on the codespace:

$$L|\psi\rangle = L'|\psi\rangle\,, \text{ for each } L, L' \in [L] \text{ and } |\psi\rangle \in \mathcal{C}(\mathcal{S}).$$

The logical Pauli group of the classical repetition code is generated by $\{Z_1, X_1X_2X_3\}$.

The distance d of $\mathcal{C}(S)$ is the minimum weight of any non-trivial logical operator in $\mathcal{L}$. Briefly we say that the code $\mathcal{C}(\mathcal{S})$ is $[\![n, k, d]\!]$. If the stabiliser generators have all weight at most r and every qubit in $(\mathbb{C}^2)^{\otimes n}$ is involved in at most c stabiliser measurements we say that the code is a (r, c) quantum LDPC code (qLDPC). The quantum repetition code is $(2, 2)$-qLDPC code of parameters $[\![3, 1, 1]\!]$.

The measurement of the stabiliser operators detects all errors in $\mathcal{P}_n \setminus \mathcal{N}(\mathcal{S})$. In particular all errors of weight less than d but the one that have trivial action on the codespace, namely Pauli operators in the stabiliser group. The optimal decoder for a distance d code can correct non-trivial Pauli errors up to weight $t = \lfloor \frac{d-1}{2} \rfloor$.

The definition of a quantum analogue of a maximum-likelihood decoder is subtle, because of the degeneracy of the syndrome map. Let $|\psi\rangle_{\mathcal{S}} \in \mathcal{C}(\mathcal{S})$ be an encoded state for some non-trivial stabiliser code with stabiliser group $\mathcal{S}$ and let $E \in \mathcal{P}_n \setminus \mathcal{N}(\mathcal{S})$ be a correctable error. Suppose that the encoded state $|\psi\rangle_{\mathcal{S}}$ undergoes the error E. Consider a decoder that, on input $\sigma(E)$, outputs F with $\sigma(F) = \sigma(E)$. Upon recovery, the state $FE|\psi\rangle_{\mathcal{S}} \in \mathcal{C}(\mathcal{S})$ is restored. The following holds:

$$\ker(\sigma) = \mathcal{N}(\mathcal{S}) \quad \Longrightarrow \quad F = EL, \text{ for some } L \in \mathcal{N}(\mathcal{S}).$$

If $L \in \mathcal{S}$, then L is the identity on the codespace and the restored state is equivalent to the original one. If instead $L \in \mathcal{N}(\mathcal{S}) \setminus \mathcal{S}$, then L is a non-trivial Pauli operator on the encoded state $|\psi\rangle_{\mathcal{S}}$ and in particular $|\psi\rangle_{\mathcal{S}} \neq L|\psi\rangle_{\mathcal{S}}$ in $\mathcal{C}(\mathcal{S})$. Ultimately, the original state is restored if the decoder outputs any of the Pauli operators equivalent to the original error E, namely operators in the set Σ_E:

$$\Sigma_E = \{SE \text{ such that } S \in \mathcal{S}\}.$$

[8] A quotient group ${}^{A}\!/_{B}$ is well defined if $B \subseteq A$ is normal in A, meaning that $aba^{-1} \in B$ for every a in A and b in B. Given a quotient group $({}^{A}\!/_{B}, \cdot)$, we indicate by $[a]$ its elements. The elements of a quotient group are equivalence classes: $[a] = \{ab \text{ such that } b \in B\}$.

Shortly:

$$\sigma(F) = \sigma(EL) = \sigma(E)$$

but

$$[L] \neq [\mathbb{1}] \quad \text{in} \quad \mathcal{N}(\mathcal{S})/\mathcal{S}$$

and the decoder fails. Under local stochastic noise, when the Pauli error E is sampled with probability:

$$\mathbb{P}(E) = p^{|E|}(1-p)^{n-|E|} \tag{1.8}$$

the quantum maximum-likelihood decoder (qML decoder) is:

$$\begin{aligned} \mathcal{D}_q : \mathrm{im}(\sigma) &\longrightarrow \mathcal{P}_n \\ s &\longmapsto \arg\max_E \mathbb{P}(\Sigma_E \,|\, \sigma(E) = s), \end{aligned}$$

where:

$$\mathbb{P}(\Sigma_E) = \sum_{E' \in \Sigma_E} \mathbb{P}(E'). \tag{1.9}$$

In other words, when Pauli errors on each qubit are independent and identically distributed so that low-weight errors are more likely, the code's degeneracy gives rise to a tension between the most probable error – namely the minimum weight one that agrees with the syndrome – and the number of different error configurations equivalent to it. An equivalence class that happens to contain multiple low-weight errors could end up having a greater overall probability than the equivalence class of the minimum weight error.

The qML decoder is optimal but the summation in Eq. (1.9) contains exponentially many terms: 2^{n-k} for a $[\![n, k, d]\!]$ code. A computationally easier algorithm is the quantum minimum-weight decoder (qMW decoder):

$$\begin{aligned} \mathcal{D}_{\mathrm{MW}} : \mathrm{im}(\sigma) &\longrightarrow \mathcal{P}_n \\ s &\longmapsto \arg\max_E \mathbb{P}(E \,|\, \sigma(E) = s). \end{aligned}$$

In general, qML and qMW decoders could yield different answers but for sufficiently low physical error probability p, it can be shown that they are equivalent [10]. The classical ML decoder instead is a minimum-weight decoder: degeneracy is exquisitely quantum.

Stabiliser codes can be readily understood in terms of binary arithmetic. Since $ZX = iY$, up to a phase, any Pauli $P \in \mathcal{P}_n$, can be written as

$$P \propto X(v) \cdot Z(w),$$

where

$$X(v) \cdot Z(w) \coloneqq X^{v_1} \otimes \ldots X^{v_n} \cdot Z^{w_1} \otimes \ldots Z^{w_n}, \quad v, w \in \mathbb{F}_2^n. \tag{1.10}$$

More formally, up to phases, the n-qubit Pauli group with multiplication is isomorphic to the additive group of the vector space $\mathbb{F}_2^{2n}$:

$$\mathcal{P}_n/_{\langle i\mathbb{1}\rangle} \longrightarrow \mathbb{F}_2^{2n} \tag{1.11}$$

$$X(v)Z(w) \cdot X(v')Z(w') \longmapsto (v + v', w + w'). \tag{1.12}$$

Two operators $X(v)Z(w)$ and $X(v')Z(w')$ commute if and only if

$$\langle v, w' \rangle + \langle v', w \rangle = 0 \in \mathbb{F}_2,$$

where

$$\begin{aligned} \langle \cdot, \cdot \rangle : \mathbb{F}_2^n \times \mathbb{F}_2^n &\longrightarrow \mathbb{F}_2 \\ v, w &\longmapsto v^T w. \end{aligned}$$

Any Pauli operator P in $\mathcal{P}_n$ can be faithfully and completely described, up to a phase, by its symplectic representation $(v, w) \in \mathbb{F}_2^{2n}$ as in Eq. (1.10). Via the symplectic representation a stabiliser code can be described via its syndrome map represented by a parity check matrix $H = (A|B)$ such that for any $P \in \mathcal{P}_n$,

$$\sigma(P) = \left(\begin{array}{c|c} A & B \end{array} \right) \left(\begin{array}{c|c} 0_n & \mathbb{1}_n \\ \hline \mathbb{1}_n & 0_n \end{array} \right) \left(\begin{array}{c} v \\ \hline w \end{array} \right) = Aw + Bv \in \mathbb{F}_2^m,$$

where $A, B \in \mathbb{F}_2^{m \times n}$, and 0_n and $\mathbb{1}_n$ are the zero and identity matrix on $\mathbb{F}_2^{n \times n}$.

A stabiliser code is a Calderbank-Shor-Steane code (CSS code) whenever the stabiliser group has a set of generators that can be partitioned in a set $\mathcal{S}_x$ of X operators and a set $\mathcal{S}_z$ of Z operators [11, 12]. In the following we describe CSS codes borrowing from the language of algebraic topology and using chain complexes, see [13]. We believe that the chain complex perspective can not only enlighten our understanding of CSS codes, but also inspires new code constructions, as it has been demonstrated several times already [14].

We can associate a length-2 chain complex over $\mathbb{F}_2$ to any CSS code and, vice-versa, to any chain complex of length at least two we can associate a CSS code. A length-ℓ chain complex is an object described by a sequence of $\ell + 1$ vector spaces $\{C_i\}_i$ over $\mathbb{F}_2$ and ℓ linear operators $\delta_i : C_i \to C_{i+1}$ such that, for each i, $\delta_{i+1}\delta_i = 0$. Given a chain complex:

$$C_0 \xrightarrow{\delta_0} C_1 \xrightarrow{\delta_1} C_2$$

where C_i has dimension n_i, we can define a CSS code as explained below.

With slight abuse of notation, let δ_0, δ_1 indicate the matrix representations of the corresponding binary maps for some preferred bases of the spaces C_0, C_1 and C_2. The X stabiliser group $\mathcal{S}_x$ is generated by the set of the $X(v)$, as v varies among the rows of δ_0^T. The Z stabiliser group $\mathcal{S}_z$ is generated by $Z(w)$, as w varies among the rows of δ_1. Because $\delta_1\delta_0 = 0$, the group $\mathcal{S}_{x\cup z} = \langle \mathcal{S}_x \cup \mathcal{S}_z \rangle$ generated by $\mathcal{S}_x \cup \mathcal{S}_z$ is Abelian and therefore defines a stabiliser code. The dimension of the code so defined is:

$$k = \dim(\ker \delta_1) - \dim(\operatorname{im} \delta_0) = \dim(\ker \delta_0^T) - \dim(\operatorname{im} \delta_1^T),$$

and equates the dimension of the first homology group, $\left(\ker \delta_1 / \operatorname{im} \delta_0, +\right)$, of the chain complex, or equivalently the dimension of the zeroth cohomology group, $\left(\ker \delta_0^T / \operatorname{im} \delta_1^T, +\right)$. We call elements in the first homology group and the zeroth cohomology group cycles and co-cycles respectively. The homology and cohomology groups are additive quotient groups and therefore their elements are equivalence classes.

Explicitly, $[v] \in {}^{\ker \delta_1}/_{\text{im}\, \delta_0}$ and $[w] \in {}^{\ker \delta_0^T}/_{\text{im}\, \delta_1^T}$ are defined as:

$$[v] = \{v + a_0 \text{ such that } a_0 \in \text{im}\, \delta_0\}, \quad \text{for } v \in \ker \delta_1,$$
$$[w] = \{w + a_1' \text{ such that } a_1' \in \text{im}\, \delta_0\}, \quad \text{for w} \in \ker \delta_0^T.$$

For a CSS code, we define the logical Pauli X group, $\mathcal{L}_x$, and the logical Pauli Z group, $\mathcal{L}_z$:

$$\mathcal{L}_x = \left\{X(v) \text{ s.t } v \in [v] \in {}^{\ker \delta_1}/_{\text{im}\, \delta_0}\right\},$$
$$\mathcal{L}_z = \left\{Z(w) \text{ s.t } w \in [w] \in {}^{\ker \delta_0^T}/_{\text{im}\, \delta_1^T}\right\}.$$

The Pauli operators in $\mathcal{L}_x$ and $\mathcal{L}_z$ are the operators whose symplectic representation are cycles and co-cycles. The logical Pauli group of the code $\mathcal{L}_{x\cup z} = \langle \mathcal{L}_x \cup \mathcal{L}_z \rangle$ is the group generated by $\mathcal{L}_x \cup \mathcal{L}_z$. Since CSS codes are stabiliser codes, what has been said for the logical Pauli group of stabiliser codes still holds:

$$\mathcal{L}_{x\cup z} = {}^{\mathcal{N}(\mathcal{S}_{x\cup z})}/_{\mathcal{S}_{x\cup z}}.$$

In particular, the equivalence classes of cycles and co-cycles respect the equivalence classes of logical operators on the codespace and two logical Pauli operators have the same actions on the codespace if and only if their symplectic representations define the same homology and cohomology class:

$$X(v)Z(w) = X(v')Z(w') \quad \text{in} \quad \mathcal{L}_{x\cup z},$$

if and only if

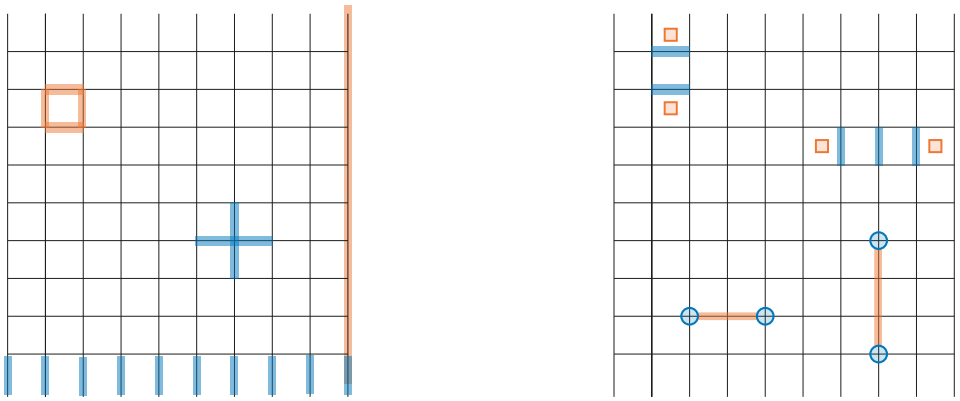
$$[v] = [v'] \quad \text{in} \quad {}^{\ker \delta_1}/_{\text{im}\, \delta_0} \qquad \text{and} \qquad [w] = [w'] \quad \text{in} \quad {}^{\ker \delta_0^T}/_{\text{im}\, \delta_1^T}.$$

For CSS codes, we define the X and Z distances, d_x and d_z, as the minimum weight of a representative of a non-trivial cycle and co-cycle. The distance of the code is the minimum between the two. For instance, the quantum repetition code is a CSS codes with $d_x = 3$ and $d_z = 1$ and hence it has distance 1. We show in Section 1.4 how to build a CSS code with both $d_x > 1$ and $d_z > 1$.

1.4 The planar code

In Section 1.3 we introduced the quantum repetition code and pointed out how it fails to protect the encoded logical qubit from arbitrary errors. In this Section, we briefly review the planar code, a quantum code which represents the proper quantum version of the classical repetition code. The planar code was first introduced in [15, 16] as an adaption to the two-dimensional planar geometry of Kitaev's seminal work on the toric code [7, 17]. Not only is the planar code the currently leading candidate in quantum error correction [18] but it is also an *hypergraph product code* – the main object of our studies in Chapters 3 to 5.

We imagine qubits placed on the edges of a $\ell \times (\ell - 1)$ square lattice $\mathfrak{L}$ and refer to this n-qubit register, where $n = \ell^2 + (\ell-1)^2 = 2\ell^2 - 2\ell + 1$, as q_ℓ. On q_ℓ, we define the operator Z_{vert} as any of the Z Pauli operators whose support spans an entire column of vertical edges. Similarly, the operator X_{hor} is defined as any X Pauli operator whose support spans an entire row of vertical edges. See Fig. 2a. The operators Z_{vert} and X_{hor} have both weight ℓ and they always anti-commute since their supports



(a) (b)

Figure 2: Planar code of distance $d = 10$. Qubits are placed on the edges. Pauli Z operators have support on orange edges, while Pauli X operators have support on blue edges. In (2a), we can see one face operator, the orange square, and one vertex operator, the blue cross. The horizontal blue line represents the support of X_{hor} and the vertical orange one represents the support of Z_{vert}. As expected, X_{hor} and Z_{vert} have odd overlap – namely the overlap on the one qubit located on the bottom-right-corner edge. In (2b) colored edges represent the support of Pauli errors: blue for X errors and orange for Z errors. Orange squares and blue circles correspond to the locations of the stabilisers which yield non-trivial syndrome. Orange squares at the boundary of blue errors are face stabilisers that anti-commute with the blue errors while blue circles are vertex stabilisers that anti-commute with the orange errors. The error locations and shapes have been chosen to highlight the similarity between the classical repetition code and the planar code, see also Fig. 1.

overlap on exactly one physical qubit. Since they respect the commutation relations of Pauli X and Pauli Z in $\mathcal{P}_1$, they suitably define logical Pauli X and Z operators on a one-dimensional subspace of $(\mathbb{C}^2)^{\otimes n}$. For every square face $\square_f$ and every vertex $+_v$ of the lattice $\mathfrak{L}$, we define the operators:

$$\square_f = \prod_{i \sim f} Z_i,$$
$$+_v = \prod_{i \sim v} X_i,$$

where i represents a qubit location. We write $i \sim f$ if the edge i belongs to the face f, and $i \sim v$ if the edge i is incident to the vertex v. We define the groups $\mathcal{S}_\square$ and $\mathcal{S}_+$ as the subgroups of $\mathcal{P}_n$ generated by the faces and vertices operators respectively:

$$\mathcal{S}_\square = \langle\, \square_f \text{ s.t. } f \text{ is a face of } \mathfrak{L} \,\rangle,$$
$$\mathcal{S}_+ = \langle\, +_v \text{ s.t. } v \text{ is a vertex of } \mathfrak{L} \,\rangle.$$

The group $\mathcal{S} = \langle \mathcal{S}_\square, \mathcal{S}_+ \rangle$ is Abelian: on a square lattice, neighbouring faces and vertices shares two edges, so the Z operator $\square_f$ and the X operator $+_v$, either have disjoint support or they overlap on

exactly two edges. Hence,

$$\mathcal{N}_\ell \xrightarrow{\gamma_\ell} \mathcal{P}_1$$
$$\mathcal{S} \to \mathbb{1},$$
$$Z_{\rm vert} \longmapsto Z,$$
$$X_{\rm hor} \longmapsto X,$$

where

$$\mathcal{N}_\ell = \langle\, i\mathbb{1},\, \mathcal{S},\, Z_{\rm vert},\, X_{\rm hor}\,\rangle$$

is a well-defined group homomorphism that fully characterizes the codespace $\mathcal{C}_\ell \simeq \mathbb{C}^2$ of the planar code:

$$\mathcal{C}_\ell = \{|\psi\rangle_\ell \in (\mathbb{C}^2)^{\otimes n} \text{ s.t. } S\,|\psi\rangle_\ell = |\psi\rangle_\ell \text{ for all } S \in \mathcal{S}\}.$$

The chain complex associated with the planar code is:

$$\mathbb{F}_2^{n-1} \xrightarrow{\sigma_+^T} \mathbb{F}_2^n \xrightarrow{\sigma_\square} \mathbb{F}_2^{n-1},$$

where σ_+ and $\sigma_\square$ are the two syndrome maps that measure every vertex and face operator respectively. Namely, for $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$, the vth coordinate of $\sigma_+(|\psi\rangle) \in \mathbb{F}_2^{n-1}$ is 1 if and only if the measurement of the observable $+_v$ on the state $|\psi\rangle \in (\mathbb{C}^2)^{\otimes n}$ yields outcome -1, and likewise for $\sigma_\square(\cdot)$. By construction, $\sigma_+(P) = \sigma_\square(P) = 0$, for every $P \in \mathcal{N}_\ell$. Shortly, the planar code has parameters $n = \ell^2 + (\ell-1)^2$, $k = 1$, $d_x = d_z = \ell$.

The planar code is the quantum twin of the classical repetition code, as we now outline. Suppose that a Z error Z_i occurs on a qubit placed on the vertical edge i. Upon measurement of face operators, the state $Z_i\,|\psi\rangle_\ell$ yield outcomes $+1$ because $\square_f$ and Z_i commute for every face $f \in \mathfrak{L}$. The situation is different when we measure vertex X-operators. Operators $+_v$ defined by vertices v that are not on the same column as the edge i, commute with Z_i. However, if we restrict to vertices on the same column as i, we find an instance of the repetition code of length ℓ: only the vertices at the north and the south of the edge i anti-commute with Z_i. Furthermore, if we expand the error pattern Z_i and consider a Z error $Z_{(w)}$ with support on $w < \ell$ consecutive vertical edges, then, again as for the repetition code, the only vertex operators that possibly anti-commute with it are the ones that reside at its north and south boundary. A similar argument works for horizontal edges, east and west boundaries and X errors, Fig. 2b. The observation that the encoding induced on columns by vertex operators is a repetition code is key in understanding that operators of the type $Z_{\rm vert}$ are the minimum weight Z-operators that can not be detected – and similarly for $X_{\rm hor}$ type operators. As in the classical repetition code, where the parallel flip operation on all the bits of the register was a valid logical operation on the encoded bit, here $Z_{\rm vert}$ and $X_{\rm hor}$ are non-trivial logical operations on the codespace, therefore undetectable. Hence, for $\ell \geq 2$, all single-qubit errors can be detected and, using $n = 2\ell^2 - 2\ell + 1$ physical qubits to protect one logical qubit, we can decrease the probability of undetectable Pauli errors from p to p^ℓ.

The planar code is a 'working' quantum error correcting code, not only it is resilient to both X and Z errors (and therefore all errors) but, at the cost of considering codes of increasing size, it can

grant protection against an arbitrary number of errors. Most importantly for our scope, it is a hypergraph product code, see Chapter 3.

In Sections 1.5 and 1.6, we formulate two central problems in the theory of quantum information processing, namely logical operations and syndrome errors. Our brief overview aims to familiarise the reader to our works in Chapters 4 and 5.

1.5 On logical operations

A code, per se, is a static object. Nonetheless, we want to process and manipulate the information stored and protected in its codespace. To this end it is necessary to develop a theory of fault-tolerant computation and seek ways to implement logical operations on the protected logical qubits. Suppose that $\mathcal{C}(\mathcal{S})$ is a $[\![n, k, d]\!]$ code, with encoding map $\hat{G} : (\mathbb{C}^2)^{\otimes k} \longrightarrow (\mathbb{C}^2)^{\otimes n}$. The obvious strategy is:

(i) Un-encode the logical qubit state $|\psi\rangle_{\text{log}} \in \mathcal{C}(\mathcal{S}) \subseteq (\mathbb{C}^2)^{\otimes n}$. Namely apply a left-inverse of $\tilde{G}$ to the n-qubit register $|\psi\rangle_{\text{log}}$, and obtain $|\tilde{\psi}\rangle_{\text{phy}} \in (\mathbb{C}^2)^{\otimes k}$.

(ii) Perform the desired quantum gate on the physical qubit register $(\mathbb{C}^2)^{\otimes k}$ in the state $|\tilde{\psi}\rangle_{\text{phy}}$, and obtain $|\tilde{\psi}'\rangle_{\text{phy}} \in (\mathbb{C}^2)^{\otimes k}$.

(iii) Encode the state $|\tilde{\psi}'\rangle_{\text{phy}}$ via the application of the encoding map $\hat{G}$, and obtain the state $|\psi'\rangle_{\text{log}} \in (\mathbb{C}^2)^{\otimes n}$.

Clearly though, during step (ii) the physical qubits are not protected from noise.

A possible solution is to avoid the un-encode step and perform the operation directly on the codespace via *transversal* operators. A unitary operator U on a n-qubit register is said to be transversal if it can be written as the tensor product of single-qubit unitaries:

$$U = \bigotimes_{i=1}^{n} U_i,$$

where U_i is unitary operator on $\mathbb{C}^2$. A transversal operator is naturally fault-tolerant. A Pauli error on one of the qubits on the register is only propagated to one single qubit, and failure of one of its product components similarly only affects one qubit. Unfortunately, by the Eastin-Knill theorem, no code allows a universal set of transversal gates [19, 20]. The Eastin–Knill theorem, however, does not rule out the existence of other fault-tolerant protocols for universal quantum computation and in fact, there exist several proposals for universal fault-tolerant computation such as the addition of measurements and classical feed-forward [21], magic state distillation [22], code concatenation [23] and pieceable fault tolerance [24].

1.6 On syndrome measurement errors

We have so far illustrated a framework where information is protected by a code, errors are detected via *perfect* syndrome measurements and the appropriate correction is found by a decoder. However, the syndrome can be noisy too and conventionally syndrome errors are dealt with by repeating multiple rounds of measurements to increase the confidence in the result. In this Section instead we discuss the single-shot scenario, where only one round of syndrome measurements is performed.

Our presentation starts with the construction of a two-dimensional classical repetition code. Not only does the two-dimensional repetition code protect one single bit from flip errors in single-shot mode but, more importantly, it constitutes a simple but sound illustration of the confinement property and related single-shot error correction features [25, 26] that are the main object of Chapter 5. Single-shot error correction has historically been understood as a by-product of self-correction phenomena [27, 28], and in this same spirit we then introduce the two-dimensional Ising model as a self-correcting classical memory. Macroscopic confinement for the two-dimensional repetition code corresponds to the existence of a macroscopic energy barrier in the two-dimensional Ising model, which ensures its self-correcting properties as a classical memory. We review these concepts and their relations in Section 1.6.1.

Given a syndrome map $\sigma : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$, we assume that the *observed* syndrome $\tilde{s}$ is noisy:

$$\tilde{s} = \sigma(e) + \eta \in \mathbb{F}_2^m,$$

where $e \in \mathbb{F}_2^n$ is the bit error vector and $\eta \in \mathbb{F}_2^m$ is the syndrome measurement error. Loosely, given a $[n, k, d]$ code with syndrome map σ, we say that a pair syndrome map-decoder $(\sigma, \mathcal{D}_{\text{ss}})$ is single-shot if

> The decoder $\mathcal{D}_{\text{ss}}$, on input a noisy syndrome $\sigma(e) + \eta$, for $|e| < t$ and $|\eta| < d_{\text{ss}}$ for some integer $d_{ss} > 1$ outputs a recovery operator e_r such that, the residual error on the system after correction is correctable:
>
> $$|e + e_r| < t,$$
>
> where $t = \lfloor {}^{d-1}/_{2} \rfloor$ and d is the distance of the code defined by the syndrome map σ.

In what follows, we refer to the classical repetition code as defined in Section 1.3 as the one-dimensional repetition code. We claim that the one-dimensional repetition code and a ML decoder are not single-shot. We illustrate our claim with an example. Consider the $[7, 1, 7]$ repetition code with syndrome map $H \in \mathbb{F}_2^{6\times 7}$ such that $H_{i,i} = 1$, $H_{i,i+1} = 1$, for $i = 1, \ldots, 6$ and $H_{i,j} = 0$ elsewhere, see Fig. 1. Let e be the bit error and η the syndrome error defined as:

$$e = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}^T, \quad \eta = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}^T.$$

The observed syndrome $\tilde{s}$ is:

$$\tilde{s} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}^T.$$

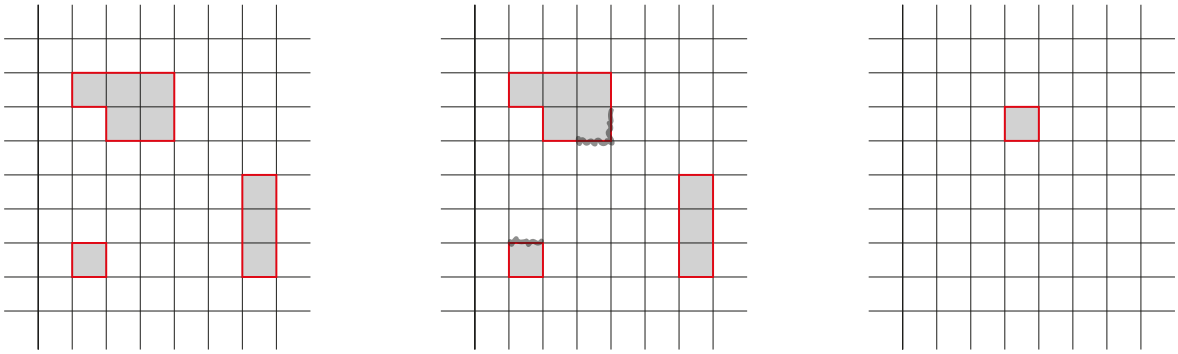
On input $\tilde{s}$, the ML decoder $\mathcal{D}$ defined in Eq. (1.7), outputs the recovery operator e_r:

$$e_r = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}^T$$

yielding the non-correctable weight-5 residual error $e + e_r$:

$$e + e_r = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}^T.$$

Although an example does not constitute a proof, it is enough to reveal why the one-dimensional repetition code does not have single-shot properties. The reason is that the one-dimensional repetition code does not have *confinement*. We can think of the syndrome weight as a cost function. In the



(a) (b) (c)

Figure 3: Two-dimensional $[90, 1, 90]$ repetition code. Bits are placed on faces and checks on edges. Grey faces represent the support of flip errors on bits. The red edges represent the support of the corresponding syndrome. In (3a) the support of the error e is highlighted in grey. The syndrome $\sigma(e)$ has support on the red edges which form the perimeter of the error area. In (3b) the error e and the observed syndrome $\sigma(e) + \eta$. The support of the syndrome error η is represented by the shaded red edges. The observed syndrome $\sigma(e) + \eta$ has support on the red edges. In (3c) the support of the residual error $e + e_r$ is highlighted in grey. Its syndrome $\sigma(e + e_r)$ has support on the red edges at its boundary. The residual error is produced after the recovery e_r is found by a ML decoder that minimises the weight of the residual syndrome $\sigma(e + e_r)$.

one-dimensional repetition code, errors on adjacent edges always trigger at most two checks, the ones at the boundaries of the error chain. Hence, starting from an error configuration where only one edge is flipped, we could lengthen the error chain at one of the boundaries without increasing the weight of its syndrome. Crucially, we could do so till we have flipped more than half of the edges, and the error chain is not correctable anymore. The key point is that the syndrome weight is not always proportional to the error weight and therefore small syndrome errors could produce high-weight residual errors: the syndrome map is not confined.

Alternatively, we can build a two-dimensional repetition code and ensure that the syndrome's weight grows with the weight of the error. In the two-dimensional repetition code we consider a square lattice where bits are placed on faces and checks on edges. Each edge assesses the parity of the faces it belongs to: the faces at its north and south for horizontal edges, the faces at its east and west for vertical edges. In particular the syndrome is local with respect to the error support. Moreover, the syndrome weight cost function grows as the perimeter of the error area: the syndrome is confined. Because the syndrome weight is proportional to the error weight, small syndrome errors will produce small residual error, see Fig. 3. This intuition is justified when the syndrome weight is mapped into an energy cost of a suitably defined statistical mechanical model. As we explain in Section 1.6.1, for our purposes the repetition code corresponds to the Ising model.

1.6.1 Statistical mechanics models for error correction

The bit (and qubit) encoding proposed rely on a common strategy that we can summarise as follows.

(i) Find an abstraction of the elementary system considered, the vector space $\mathbb{F}_2$.

(ii) Identify its basic dynamics, the flip operator.

(iii) Find a description for composite systems, the direct sum of vector spaces over $\mathbb{F}_2$.

(iv) Find an embedding of the elementary system into a composite one such that the elementary system is there described by global features, the syndrome map.

(v) Determine a suitable description of the dynamics of the embedded elementary system, logical flip.

(vi) Verify that the chosen embedding mitigates the impact of errors, lower probability of undetected errors.

Essential to successful error protection in this paradigm is the embedding of the dynamics of the elementary system, the bit, into macroscopic properties of a large system of interacting elementary units, a multi-bit register. The prototypical example of such embedding in statistical mechanics is the Ising model of ferromagnetism and its physical realization is the basis of magnetic classical data storage [29]. The fact that the two-dimensional (2D) Ising model has a phase transition ensures the existence of a self-correcting bit memory – a macroscopic physical system that enables robust storage and manipulation of classical information without the need for active error correction [27, 30].

In this Section, we think of a bit as a spin-$\frac{1}{2}$ particle whose possible states are the $\uparrow$ and the $\downarrow$ state, corresponding to 0 and 1 respectively. In the D-dimensional Ising model, a spin-$\frac{1}{2}$ is placed at each site of a D-dimensional lattice and each spin is coupled to its neighbouring spins [29]. We here consider only square lattices and indicate by Ω the configuration space for n spins on a D-dimensional square lattice, $|\Omega| = 2^n$. For $\omega \in \Omega$ with the assignment $\uparrow$ to $+1$ and $\downarrow$ to -1, the total energy of the system reads:

$$H(\omega) = -J \sum_{i \sim j} \omega_i \omega_j, \tag{1.13}$$

where $i \sim j$ if the lattice site i and j are neighbours. On a square lattice in D-dimension each spin has $2D$ neighbours e.g. in 1 dimension each spin in the bulk is coupled to its left and right neighbours and in 2 dimensions it is coupled to its north, east, west and south ones. We consider the ferromagnetic Ising model, obtained for $J > 0$ in Eq. (1.13). A ferromagnetic system has two degenerate ground states: the one with all spins aligned in the $\uparrow$ state, $\omega^{\uparrow}$, and the one with all spins aligned in the $\downarrow$ state, $\omega^{\downarrow}$. We define the magnetization of a configuration $\omega \in \Omega$ as:

$$m(\omega) = \frac{1}{n} \sum_{\omega_i \in \omega} \omega_i.$$

The magnetization $m(\omega)$ is a real value in $[-1, 1]$. If we consider the interaction of the spin system with the environment, its statistical behaviour is described by the Boltzmann distribution. The probability of the system of being in the configuration ω is

$$\mathbb{P}(\omega) = \frac{e^{-\beta H(\omega)}}{Z_\beta}, \quad \omega \in \Omega,$$

where $\beta = 1/T$ is the inverse temperature (with appropriate choice of scale i.e. T is the inverse thermal energy $T = kT'$ where T' is the temperature in absolute degrees and k is the Boltzmann's constant)

and Z_β is the partition function:

$$Z_\beta = \sum_{\omega \in \Omega} e^{-\beta H(\omega)}.$$

The average magnetization of the system is defined as:

$$\langle m \rangle = \sum_{\omega \in \Omega} m(\omega) \mathbb{P}(\omega). \tag{1.14}$$

For all finite positive temperatures, the average magnetization of the system is 0. In fact, if $-\omega$ denotes the configuration where all spins are flipped with respect to ω, we have $H(\omega) = H(-\omega)$ and hence $\mathbb{P}(\omega) = \mathbb{P}(-\omega)$. Because $m(\omega) = -m(-\omega)$, the non-zero terms in Eq. (1.14) cancel pairwise and $\langle m \rangle = 0$. The system however has a temperature dependency. In the limit $\beta \to 0$ the Boltzmann distribution on Ω converges to the uniform distribution: for each $\omega \in \Omega$,

$$\lim_{\beta \to 0} \mathbb{P}(\omega) = \frac{1}{|\Omega|}.$$

For $\beta \to \infty$ instead the Boltzmann distribution concentrates on the configuration $\omega^\uparrow$ and $\omega^\downarrow$ that minimise the energy in Eq. (1.13), the ground states. Therefore, for $T \to 0$ and $\beta \to \infty$:

$$\lim_{\beta \to \infty} \mathbb{P}(\omega) = \begin{cases} \frac{1}{2}, & \text{for } \omega = \omega^\uparrow, \omega^\downarrow, \\ 0, & \text{otherwise.} \end{cases} \tag{1.15}$$

Equation (1.15) says that, in the limit of small temperatures, a system initialised in one of the two ground states is stable under sufficiently small thermal fluctuations. The two limit cases of infinite and zero temperature reveal the existence of very different possible scenarios. At high temperatures, we expect the typical configuration to have an equal proportion of spins in the $\uparrow$ and $\downarrow$ states. At zero temperature, we expect most of the spins to be in the same state. Nonetheless, these two limit cases do not inform us on the typical behaviour of the system for intermediate values of the inverse temperature $0 < \beta < \infty$. For our purposes, we want to determine if it is possible to preserve the system in one of the two ground states at finite values of $\beta < \infty$, at the price of considering systems of growing size. In other words, if above we have fixed the size n of the configuration space and studied the behaviour as $\beta \to \infty$, now we want to study the behaviour of the system when β is fixed and $n \to \infty$.

The problem is to assess whether or not there exist arbitrarily low but positive values of the temperature for which the magnetization of the system is observed as a global order parameter. More precisely, we want to understand whether D-dimensional lattice spin systems of increasingly large size n exhibits global magnetization, for some arbitrary $\beta < \infty$. A counting argument shows that the probability for the system to be in one of the two ground states decreases, even at low temperatures, as the number of spins increases. In fact, the number of excited states that differs from one of the two ground states on w sites is proportional to $\binom{n}{w} \sim \frac{n^w}{w!}$. Hence, even if the ground states have larger individual probability at low temperatures, the number of different possible configurations of excited states grows with the size of the system n. The tension between these two phenomena, the energy and the entropy cost of excitations, determines the existence of a phase transition, a discontinuous change in the global magnetization of the system.

In the Ising model, the existence of a phase transition is dimensionally-dependent: the 1D Ising

model has no phase transition whilst the 2D Ising model, or actually the Ising model in any dimension D$\geq$ 2, does have a phase transition. To understand how this is the case we consider the energy cost, or energy barrier, of a spin flip. Let us consider a 1D and a 2D lattice where most of the spins are aligned in the $\uparrow$ state but there are some 'droplets' of flipped neighbouring spins in the $\downarrow$ state. Eq. (1.13) imposes an energy cost of $2J$ for each pair of anti-aligned neighbouring spins, so that the energy cost for flipping a droplet of spins is proportional to its boundary. Because in 1D boundaries of consecutive segments of spins are zero-dimensional, flipping one spin has an energy penalty of $4J$, but flipping any of its neighbours has no additional energy cost. Thus, starting from the ground state configuration $\omega^{\uparrow}$, we could reverse the magnetization of the system at constant energy cost – $4J$. In 2D instead, the boundary of a droplet grows proportionally to its area and therefore the energy cost for reversing the magnetization of the system is proportional to the system size. As such, we expect systems of increasing size to be increasingly more robust to thermal fluctuations. Peierls argument [31] that the 2D Ising model has a phase transition at some critical temperature $T_c > 0$ is based on this same intuition and in particular entails that the macroscopic behaviour of the 2D Ising model depends on the temperature. At high temperature the typical configuration has magnetization close to 0, whilst below T_c the absolute value of the magnetization gets closer to 1 as the temperature decreases, see Fig. 4.

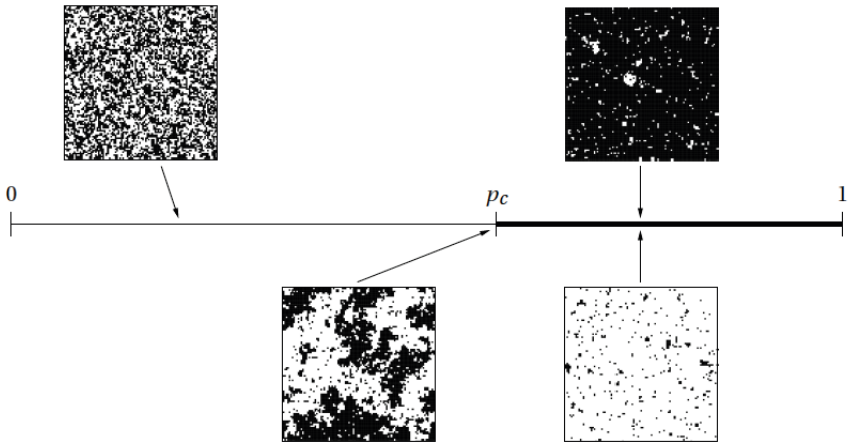


Figure 4: Figure from [29]. Monte Carlo simulations of the 2D Ising model for $n = 100$ as $p = 1 - e^{-2\beta} \in [0, 1)$ varies. White dots represent $\uparrow$ spins and black dots represent $\downarrow$ spins, $p_c = 1 - e^{-2\frac{1}{T_c}}$ corresponds to the value at the critical temperature T_c. For high temperatures, p is close to 0 and the $\uparrow$ and $\downarrow$ states roughly appear in the same proportions. At low temperatures and above p_c, a typical configuration is a ground state with a small number of excitations and hence would give magnetization close to either $+1$ or -1.

The correspondence of the Ising model to a self-correcting memory is understood via the identification of thermal fluctuations with random flip errors . The states $\omega^{\uparrow}$ and $\omega^{\downarrow}$ can be naturally thought of as the length n repetition code encoding of a bit or a qubit, as in Eq. (1.1) and Eq. (1.3). The magnetization is a global property of the system and it is also a measure of the information there stored: $m(\omega^{\uparrow}) = 1$, $m(\omega^{\downarrow}) = -1$ and $m(\omega) \neq \pm 1$ otherwise. If we find $m > 0$, a majority of the spins is in the $\uparrow$ state and a ML decoder would interpret the encoded bit as being in the 0 state; vice-versa, $m < 0$ indicates the 1 state. The existence of a phase transition therefore determines the ability of the

spin system to robustly encode a bit: if the magnetization of the system is preserved at some non-zero temperature, the logical bit can undergo thermal excitations but still preserve its information content. In conclusion, a bit encoded in the magnetization of the 2D Ising model where flip errors correspond to thermal fluctuations exhibits self-correction below a critical temperature T_c. Below T_c, the thermal fluctuations will correct errors at a higher pace than they are introduced, by favouring lower energy configurations in which the error droplets narrow.

Self-correction and single-shot error correction are complementary problems [25]. In both settings, the goal is not to remove all entropy – all errors – in the system but only to keep it under control. By ensuring that the residual error left on the system is small enough, we can be confident that the information there stored is not irreversibly damaged. Self-correction is intrinsically syndrome-agnostic and therefore could explain single-shot error correction, where syndrome information has to be treated as maybe inaccurate. We refer to Chapter 5 for an in-depth analysis of single-shot error correction.

References

[1] Jacobus Hendricus Van Lint. *Introduction to coding theory*. Vol. 86. Springer Science & Business Media, 2012.

[2] Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge university press, 2008.

[3] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[4] Daniel A Lidar and Todd A Brun. *Quantum error correction*. Cambridge university press, 2013.

[5] Lang Serge. *Linear Algebra*. Springer New York, NY, 1987.

[6] Robert Raussendorf and Hans J Briegel. "A one-way quantum computer". In: *Physical review letters* 86.22 (2001), p. 5188.

[7] A Yu Kitaev. "Quantum computations: algorithms and error correction". In: *Russian Mathematical Surveys* 52.6 (1997), p. 1191.

[8] Daniel Gottesman. "Fault-tolerant quantum computation with constant overhead". In: *arXiv preprint arXiv:1310.2984* (2013).

[9] Daniel Gottesman. *Stabilizer codes and quantum error correction*. California Institute of Technology, 1997.

[10] Pavithran Iyer and David Poulin. "Hardness of decoding quantum stabilizer codes". In: *IEEE Transactions on Information Theory* 61.9 (2015), pp. 5209–5223.

[11] A Robert Calderbank and Peter W Shor. "Good quantum error-correcting codes exist". In: *Physical Review A* 54.2 (1996), p. 1098.

[12] Andrew Steane. "Multiple-particle interference and quantum error correction". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 452.1954 (1996), pp. 2551–2577.

[13] Allen Hatcher. "Algebraic topology". In: (2001).

[14] Nikolas P Breuckmann and Jens Niklas Eberhardt. "Quantum low-density parity-check codes". In: *PRX Quantum* 2.4 (2021), p. 040101.

[15] Sergey B Bravyi and A Yu Kitaev. "Quantum codes on a lattice with boundary". In: *arXiv preprint quant-ph/9811052* (1998).

[16] Michael H Freedman and David A Meyer. "Projective plane and planar quantum codes". In: *Foundations of Computational Mathematics* 1.3 (2001), pp. 325–332.

[17] A Yu Kitaev. "Fault-tolerant quantum computation by anyons". In: *Annals of Physics* 303.1 (2003), pp. 2–30.

[18] Austin G Fowler et al. "Surface codes: Towards practical large-scale quantum computation". In: *Physical Review A* 86.3 (2012), p. 032324.

[19] Xie Chen et al. "Subsystem stabilizer codes cannot have a universal set of transversal gates for even one encoded qudit". In: *Physical Review A* 78.1 (2008), p. 012353.

[20] Bryan Eastin and Emanuel Knill. "Restrictions on transversal encoded quantum gate sets". In: *Physical review letters* 102.11 (2009), p. 110502.

[21] Xinlan Zhou, Debbie W Leung, and Isaac L Chuang. "Methodology for quantum logic gate construction". In: *Physical Review A* 62.5 (2000), p. 052316.

[22] Sergey Bravyi and Alexei Kitaev. "Universal quantum computation with ideal Clifford gates and noisy ancillas". In: *Physical Review A* 71.2 (2005), p. 022316.

[23] Tomas Jochym-O'Connor and Raymond Laflamme. "Using concatenated quantum codes for universal fault-tolerant quantum gates". In: *Physical review letters* 112.1 (2014), p. 010505.

[24] Theodore J Yoder, Ryuji Takagi, and Isaac L Chuang. "Universal fault-tolerant gates on concatenated stabilizer codes". In: *Physical Review X* 6.3 (2016), p. 031039.

[25] Hector Bombin. "Single-shot fault-tolerant quantum error correction". In: *Physical Review X* 5.3 (2015), p. 031043.

[26] Earl T Campbell. "A theory of single-shot error correction for adversarial noise". In: *Quantum Science and Technology* 4.2 (2019), p. 025006.

[27] Barbara M Terhal. "The fragility of quantum information?" In: *International Conference on Theory and Practice of Natural Computing*. Springer. 2012, pp. 47–56.

[28] Benjamin J Brown et al. "Quantum memories at finite temperature". In: *Reviews of Modern Physics* 88.4 (2016), p. 045005.

[29] Sacha Friedli and Yvan Velenik. *Statistical mechanics of lattice systems: a concrete mathematical introduction*. Cambridge University Press, 2017.

[30] Francesca Valery Day and Sean Barrett. "The Ising ferromagnet as a self-correcting physical memory: a Monte-Carlo study". In: *arXiv preprint arXiv:1201.0390* (2012).

[31] Rudolf Peierls. "On Ising's model of ferromagnetism". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 32. 3. Cambridge University Press. 1936, pp. 477–481.

Chapter 2

The need for *good* codes - Overview

The two main challenges to overcome in order to build a usable and useful quantum computer are noise and scalability. Careful quantum code design could provide the solution to both. The planar code is today the leading approach for fault-tolerant quantum computing. Crucially, it is an LDPC code for which the syndrome extraction circuit is not only low-depth but also local[1]. Locality is what makes planar code experimental realizations practical [1, 2, 3, 4]. However, the planar code family parameters are not *good*.

The term good is borrowed from classical coding theory: a $[n, k, d]$ code is said good if both k and d scale linearly with n – the same definition applies to a $[\![n, k, d]\!]$ quantum code. A code family has rate $r \in \mathbb{R}$ if the ratio k/n tend to r, as we consider codes of increasing size n. In particular, codes with dimension linear in their length will have positive rate $r > 0$. A family of codes has a threshold $p_{\text{th}} > 0$ if, for error rates below p_{th} in the examined noise model, there exists a decoder whose probability of failure decays exponentially in the system size [5]. Loosely, having a threshold means that the code is robust to typical errors and hence, by considering codes of increasing size and distance, we can increase the system's resilience to faults.

The planar code has distance scaling as $\sqrt{n}$ and zero rate. Even if the distance scaling is sub-linear, the planar code family has a threshold [6, 7]. Having a threshold, the planar code solves the problem of noise but fails in addressing scalability. Thirteen physical qubits[2] are needed to protect one logical qubit from one error, and a reasonably protected logical qubit would need between 10^3 and 10^4 physical qubits, depending on the application and the noise in the device [9, 10, 11, 12]. This contrasts with the best achievable parameters for classical LDPC codes: as first shown by Gallager in 1963 [13], good classical LDPC codes do exist. Furthermore, these codes can be efficiently decoded [14, 15]. In classical coding theory the LDPC property is necessary for the existence of efficient decoders, but, for quantum codes, it is even more relevant. At a high level, a scalable fault-tolerant quantum computation needs, of course, codes with a good rate, good distance and an efficient decoding algorithm. At a lower level, it needs to be resilient to errors occurring at any step in the circuit model. We need fault-tolerant state preparation, fault-tolerant measurements for syndrome extraction and fault-tolerant gate constructions. The LDPC assumption is here essential in the design of quantum

[1] As seen in Section 1.4, the planar code is a $(4, 4)$-qLDPC code i.e. each stabiliser only involves at most 4 qubits, and at most 4 stabilisers act on each qubit.

[2] More precisely: 13 data qubits and 12 auxilia qubits, for a total of 25 physical qubits. This figure is reduced to 17 physical qubits – 9 data and 8 auxilia qubits – for the rotated version of the planar code, see [8].

codes because the number of gates, and hence potentially faulty locations, required to perform error correction scales with the stabilizer weight [16, 17, 18].

Gallager's construction is random: with high probability, a random low-density parity check matrix will define a classical code with constant rate and linear distance. However, to build a quantum LDPC code, some structure is needed. On the one hand, a $[\![n, k, d]\!]$ stabilizer code can be mapped onto a $[\![4n, 2k, 2d]\!]$ CSS code, so that parameters are the same up to constant factors [19]. On the other, we cannot employ random binary code constructions to define CSS codes via their symplectic representation. A random matrix for the X stabilizers would define a high-distance classical code, as such the resulting Z stabilizer matrix will not be sparse. In the literature, structure to ensure the commutativity constraints whilst preserving the sparsity of the check matrices has either been given by geometry or via randomness aided by product constructions. The earliest examples of the former, obtained via hyperbolic surfaces, can be found in [20, 21, 22] or the review [23]. Our work focuses on the first product construction, the hypergraph product, originally proposed by Tillich and Zémor in [24]. The planar code is a special instance of hypergraph product codes which, in general, use classical random matrices as seeds [24, 25, 26, 27]. When good LDPC code families are used as seeds, the corresponding hypergraph product codes have constant rate and distance scaling as $\sqrt{n}$. Even if hypergraph product codes do not have great distance scaling, as happens for the planar code, they tolerate typical errors [5]. Furthermore, they can be efficiently decoded [28, 29, 30, 31, 32, 33, 34, 35] and some work has also been done in terms of fault-tolerant computation [36, 37, 38], and syndrome extraction circuits [39, 40].

When we started working on hypergraph product codes they were the best LDPC code family known, with $k \sim n$ and $d \sim \sqrt{n}$. This is no longer the case and good qLDPC codes are now known to exist. We here report a brief history of the good qLDPC discovery and refer the reader to [23] for a –already outdated– review. In 1997 Kitaev introduces the toric code – a hypergraph product code with parameters $[\![2n^2, 2, n]\!]$. In 2002, Freedman, Meyer and Luo used hyperbolic geometry to build a family of codes with distance scaling as $\sqrt{n}\sqrt[4]{\log n}$ but again with zero rate [20]. No progress was made until 2020, when the first code constructions with $d \sim \sqrt{n}\,\mathrm{polylog}\, n$ were proposed [41, 42] and then finally $d > \sqrt{n}\,\mathrm{polylog}\, n$ [43, 44, 45]. Some of these constructions yield a constant rate in the asymptotic limit but still do not obey the definition of good codes. Panteleev and Kalachev eventually exhibited the first construction of asymptotically good quantum LDPC codes in 2021 [46], closely followed by Leverrier and Zémor [47], Dinur et al. [48], Lin and Hsieh [49]. The existence of (asymptotically) good quantum LDPC codes is definitely a breakthrough for our field but it is far from being the end of the story.

Good LDPC codes could solve scalability in theory, but much more goes into the architecture of a fault-tolerant quantum computer – at a minimum: experimentally feasible circuit design, practical efficient decoders and fault-tolerant encoded operations are needed. These issues have been studied at depth for the surface code (see, for instance, the review [9]) and less so for codes with $k > 1$ [23]. Gottesman's foundational Threshold Theorem [50] ensures that fault-tolerant quantum computation is possible with constant overhead if positive rate codes are available. The overhead is there defined as the ratio of physical qubits over logical qubits used in a fault-tolerant algorithm, and Gottesman shows that it scales as the inverse rate of the code family, in the limit of very large computations. Nonetheless, the Threshold Theorem establishes that fault-tolerant quantum computing architectures are viable as long as we use positive rate codes.

Hypergraph product codes satisfy all necessary conditions of Gottesman's result – roughly, they are LDPC, have constant rate, efficient decoding algorithms and a threshold [5, 29, 30]. Hypergraph product codes could thus serve as a bridge between a surface-code architecture and a constant-overhead one. Their study could inform the exploration and design of new techniques for syndrome extraction, decoding and fault-tolerant gates implementation that could be later generalised to other code constructions. As per the Threshold Theorem, they could also serve as the cardinal building block of a scalable quantum computer.

The three pieces of original work we present in Chapter 3, Chapter 4 and Chapter 5 aim to aid and foster the design of a full fault-tolerant architecture with hypergraph product codes. In the effort of giving practical answers to the Threshold Theorem requirements, we address the issues of decoding, fault-tolerant gates and measurement errors during syndrome extraction. In Chapter 3 we detail the code construction and present a novel qMW decoder for hypergraph product codes. We propose a fault-tolerant method to implement logical gates in Chapter 4. In Chapter 5 we exhibit a foundational result on code design for faulty syndromes. We conclude in Chapter 6, discussing some open questions.

References

[1] James R Wootton et al. "Proposal for a minimal surface code experiment". In: *Physical Review A* 96.3 (2017), p. 032338.

[2] Sergey Bravyi et al. "Correcting coherent errors with surface codes". In: *npj Quantum Information* 4.1 (2018), pp. 1–6.

[3] Sebastian Krinner et al. "Realizing repeated quantum error correction in a distance-three surface code". In: *Nature* 605.7911 (2022), pp. 669–674.

[4] Rajeev Acharya et al. "Suppressing quantum errors by scaling a surface code logical qubit". In: *arXiv preprint arXiv:2207.06431* (2022).

[5] Alexey A Kovalev and Leonid P Pryadko. "Fault-Tolerance of" Bad" Quantum Low-Density Parity Check Codes". In: *arXiv preprint arXiv:1208.2317* (2012).

[6] Eric Dennis et al. "Topological quantum memory". In: *Journal of Mathematical Physics* 43.9 (2002), pp. 4452–4505.

[7] Chenyang Wang, Jim Harrington, and John Preskill. "Confinement-Higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory". In: *Annals of Physics* 303.1 (2003), pp. 31–58.

[8] Alexey A Kovalev and Leonid P Pryadko. "Improved quantum hypergraph-product LDPC codes". In: *2012 IEEE International Symposium on Information Theory Proceedings*. IEEE. 2012, pp. 348–352.

[9] Austin G Fowler et al. "Surface codes: Towards practical large-scale quantum computation". In: *Physical Review A* 86.3 (2012), p. 032324.

[10] Earl Campbell, Ankur Khurana, and Ashley Montanaro. "Applying quantum algorithms to constraint satisfaction problems". In: *Quantum* 3 (2019), p. 167.

[11] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". In: *Quantum* 5 (2021), p. 433.

[12] Isaac H Kim et al. "Fault-tolerant resource estimate for quantum chemical simulations: Case study on Li-ion battery electrolyte molecules". In: *Physical Review Research* 4.2 (2022), p. 023019.

[13] Robert Gallager. "Low-density parity-check codes". In: *IRE Transactions on information theory* 8.1 (1962), pp. 21–28.

[14] Michael Sipser and Daniel A Spielman. "Expander codes". In: *IEEE transactions on Information Theory* 42.6 (1996), pp. 1710–1722.

[15] David JC MacKay and Radford M Neal. "Near Shannon limit performance of low density parity check codes". In: *Electronics letters* 33.6 (1997), pp. 457–458.

[16] Peter W Shor. "Fault-tolerant quantum computation". In: *Proceedings of 37th conference on foundations of computer science*. IEEE. 1996, pp. 56–65.

[17] Rui Chao and Ben W Reichardt. "Quantum error correction with only two extra qubits". In: *Physical review letters* 121.5 (2018), p. 050502.

[18] Rui Chao and Ben W Reichardt. "Flag fault-tolerant error correction for any stabilizer code". In: *PRX Quantum* 1.1 (2020), p. 010302.

[19] Sergey Bravyi, Barbara M Terhal, and Bernhard Leemhuis. "Majorana fermion codes". In: *New Journal of Physics* 12.8 (2010), p. 083039.

[20] Michael H Freedman, David A Meyer, and Feng Luo. "Z2-systolic freedom and quantum codes". In: *Mathematics of quantum computation*. Chapman and Hall/CRC, 2002, pp. 303–338.

[21] Larry Guth and Alexander Lubotzky. "Quantum error correcting codes and 4-dimensional arithmetic hyperbolic manifolds". In: *Journal of Mathematical Physics* 55.8 (2014), p. 082202.

[22] Michael H Freedman and Matthew B Hastings. "Quantum systems on non-k-hyperfinite complexes: A generalization of classical statistical mechanics on expander graphs". In: *arXiv preprint arXiv:1301.1363* (2013).

[23] Nikolas P Breuckmann and Jens Niklas Eberhardt. "Quantum low-density parity-check codes". In: *PRX Quantum* 2.4 (2021), p. 040101.

[24] Jean-Pierre Tillich and Gilles Zémor. "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength". In: *IEEE Transactions on Information Theory* 60.2 (2013), pp. 1193–1202.

[25] Sergey Bravyi and Matthew B Hastings. "Homological product codes". In: *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 2014, pp. 273–282.

[26] Anthony Leverrier, Jean-Pierre Tillich, and Gilles Zémor. "Quantum expander codes". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 810–824.

[27] Weilei Zeng and Leonid P Pryadko. "Higher-dimensional quantum hypergraph-product codes with finite rates". In: *Physical review letters* 122.23 (2019), p. 230501.

[28] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. "Efficient decoding of random errors for quantum expander codes". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 521–534.

[29] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. "Constant overhead quantum fault-tolerance with quantum expander codes". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018, pp. 743–754.

[30] Alexey A Kovalev et al. "Numerical and analytical bounds on threshold error rates for hypergraph-product codes". In: *Physical Review A* 97.6 (2018), p. 062320.

[31] Antoine Grospellier and Anirudh Krishna. "Numerical study of hypergraph product codes". In: *arXiv preprint arXiv:1810.03681* (2018).

[32] Pavel Panteleev and Gleb Kalachev. "Degenerate quantum LDPC codes with good finite length performance". In: *Quantum* 5 (2021), p. 585.

[33] Antoine Grospellier et al. "Combining hard and soft decoders for hypergraph product codes". In: *Quantum* 5 (2021), p. 432.

[34] Joschka Roffe et al. "Decoding across the quantum low-density parity-check code landscape". In: *Physical Review Research* 2.4 (2020), p. 043423.

[35] Omar Fawzi, Lucien Grouès, and Anthony Leverrier. "Linear programming decoder for hypergraph product quantum codes". In: *2020 IEEE Information Theory Workshop (ITW)*. IEEE. 2021, pp. 1–5.

[36] Tomas Jochym-O'Connor. "Fault-tolerant gates via homological product codes". In: *Quantum* 3 (2019), p. 120.

[37] Anirudh Krishna and David Poulin. "Fault-tolerant gates on hypergraph product codes". In: *Physical Review X* 11.1 (2021), p. 011023.

[38] Lawrence Z Cohen et al. "Low-overhead fault-tolerant quantum computing using long-range connectivity". In: *Science Advances* 8.20 (2022).

[39] Nicolas Guillaume Delfosse, Maxime Tremblay, and Michael Edward Beverland. *Short-depth syndrome extraction circuits in 2d quantum architectures for hypergraph product codes*. US Patent App. 17/219,331. 2022.

[40] Maxime A Tremblay, Nicolas Delfosse, and Michael E Beverland. "Constant-overhead quantum error correction with thin planar connectivity". In: *Physical Review Letters* 129.5 (2022), p. 050504.

[41] Shai Evra, Tali Kaufman, and Gilles Zémor. "Decodable quantum LDPC codes beyond the square root distance barrier using high dimensional expanders". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2020, pp. 218–227.

[42] Tali Kaufman and Ran J Tessler. "New cosystolic expanders from tensors imply explicit Quantum LDPC codes with $\Omega\sqrt{n}\log^k n)$ distance". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 1317–1329.

[43] Matthew B Hastings, Jeongwan Haah, and Ryan O'Donnell. "Fiber bundle codes: breaking the n 1/2 polylog (n) barrier for quantum ldpc codes". In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 1276–1288.

[44] Pavel Panteleev and Gleb Kalachev. "Quantum LDPC codes with almost linear minimum distance". In: *IEEE Transactions on Information Theory* 68.1 (2021), pp. 213–229.

[45] Nikolas P Breuckmann and Jens N Eberhardt. "Balanced product quantum codes". In: *IEEE Transactions on Information Theory* 67.10 (2021), pp. 6653–6674.

[46] Pavel Panteleev and Gleb Kalachev. "Asymptotically good quantum and locally testable classical LDPC codes". In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. 2022, pp. 375–388.

[47] Anthony Leverrier and Gilles Zémor. "Quantum tanner codes". In: *arXiv preprint arXiv:2202.13641* (2022).

[48] Irit Dinur et al. "Good Quantum LDPC Codes with Linear Time Decoders". In: *arXiv preprint arXiv:2206.07750* (2022).

[49] Ting-Chun Lin and Min-Hsiu Hsieh. "Good quantum LDPC codes with linear time decoder from lossless expanders". In: *arXiv preprint arXiv:2203.03581* (2022).

[50] Daniel Gottesman. "Fault-tolerant quantum computation with constant overhead". In: *arXiv preprint arXiv:1310.2984* (2013).

Chapter 3

Code construction and decoding - ReShape

Context and Results

In [1] we propose the ReShape decoder for hypergraph product codes: we show how to lift a classical maximum-likelihood decoder (ML decoder, see page 9) to a quantum minimum-weight decoder (qMW decoder, see page 11). The adaptation of classical decoders to the quantum setting is, in general, not a trivial task. For instance, belief-propagation [2] can efficiently decode all classical LDPC codes but, because of the degeneracy of quantum codes, it cannot be used out-of-the-box on stabilizer codes. Similarly, the bit-flip decoding algorithm for classical expander codes fails in the quantum setting [3, 4]. Both these algorithms however can be modified to work on quantum codes [5, 6, 7, 8, 9, 10, 11]. Differently, our ReShape decoder uses the classical decoding algorithm as a black box. In a nutshell, hypergraph product codes are built via the tensor product of two classical linear codes; we 'invert' the product structure to reduce the quantum decoding problem to a classical one. This action, an ad hoc version of the Cleaning Lemma [12], informs us of some invariant properties of hypergraph product codes which further clarify the composition of their logical Pauli group.

Limitations

The ReShape decoder is not suitable for use in realistic noise settings as it only works for adversarial noise. An adversarial error model is such that an adversary chooses error locations to damage the encoded information. Under this noise model, and by definition of $[\![n, k, d]\!]$ code, an adversary is always able to find $t = \lfloor \frac{d-1}{2} \rfloor$ locations for which *any* decoder would fail. On the contrary, under local stochastic noise of parameter p, we expect to have about pn errors on n physical qubits, but we also expect the probability of t malicious locations to undergo errors to be low. Kovalev and Pryadko showed that *typical* errors can be corrected by any qLDPC code provided that p is below a certain threshold p_{th} [13]. At low p, typical errors form isolated clusters in a low-degree graph associated with the code. As such, a decoder able to correct isolated and not malicious clusters separately would succeed with high probability. ReShape however is intrinsically non-local on the relevant graph and hence it fails to satisfy Kovalev and Pryadko's assumptions.

Authorship declaration

AOQ derived the proofs, the numerical simulations and wrote the manuscript.

References

[1] Armanda O Quintavalle and Earl T Campbell. "ReShape: a decoder for hypergraph product codes". In: *IEEE Transactions on Information Theory* (2022).

[2] David JC MacKay and Radford M Neal. "Near Shannon limit performance of low density parity check codes". In: *Electronics letters* 33.6 (1997), pp. 457–458.

[3] Daniel A Spielman. "Linear-time encodable and decodable error-correcting codes". In: *IEEE Transactions on Information Theory* 42.6 (1996), pp. 1723–1731.

[4] Michael Sipser and Daniel A Spielman. "Expander codes". In: *IEEE transactions on Information Theory* 42.6 (1996), pp. 1710–1722.

[5] David JC MacKay, Graeme Mitchison, and Paul L McFadden. "Sparse-graph codes for quantum error correction". In: *IEEE Transactions on Information Theory* 50.10 (2004), pp. 2315–2330.

[6] David Poulin and Yeojin Chung. "On the iterative decoding of sparse quantum codes". In: *arXiv preprint arXiv:0801.1241* (2008).

[7] Pavel Panteleev and Gleb Kalachev. "Degenerate quantum LDPC codes with good finite length performance". In: *Quantum* 5 (2021), p. 585.

[8] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. "Constant overhead quantum fault-tolerance with quantum expander codes". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018, pp. 743–754.

[9] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier. "Efficient decoding of random errors for quantum expander codes". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 521–534.

[10] Kao-Yueh Kuo and Ching-Yi Lai. "Refined belief propagation decoding of sparse-graph quantum codes". In: *IEEE Journal on Selected Areas in Information Theory* 1.2 (2020), pp. 487–498.

[11] Kao-Yueh Kuo and Ching-Yi Lai. "Exploiting degeneracy in belief propagation decoding of quantum codes". In: *npj Quantum Information* 8.1 (2022), pp. 1–9.

[12] Sergey Bravyi and Barbara Terhal. "A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes". In: *New Journal of Physics* 11.4 (2009), p. 043029.

[13] Alexey A Kovalev and Leonid P Pryadko. "Fault-Tolerance of" Bad" Quantum Low-Density Parity Check Codes". In: *arXiv preprint arXiv:1208.2317* (2012).

ReShape: A Decoder for Hypergraph Product Codes

Armanda O. Quintavalle and Earl T. Campbell

***Abstract*— The design of decoding algorithms is a significant technological component in the development of fault-tolerant quantum computers. Often design of quantum decoders is inspired by classical decoding algorithms, but there are no general principles for building quantum decoders from classical decoders. Given any pair of classical codes, we can build a quantum code using the hypergraph product, yielding a hypergraph product code. Here we show we can also lift the decoders for these classical codes. That is, given oracle access to a minimum weight decoder for the relevant classical codes, the corresponding $[[n, k, d]]$ quantum code can be efficiently decoded for any error of weight smaller than $(d-1)/2$. The quantum decoder requires only $O(k)$ oracle calls to the classical decoder and $O(n^2)$ classical resources. The lift and the correctness proof of the decoder have a purely algebraic nature that draws on the discovery of some novel homological invariants of the hypergraph product codespace. While the decoder works perfectly for adversarial errors, that is errors of weight up to half the code distance, it is not suitable for more realistic stochastic noise models and therefore can not be used to establish an error correcting threshold.**

***Index Terms*— Quantum error correction, product codes, decoding.**

I. Introduction

The construction of quantum codes often takes classical codes as a starting point. The CSS construction is one method for combining a pair of classical codes into a quantum code. However, the CSS recipe only works when the pair of classical codes are dual to each other. Unfortunately, some of the best known classical code families, such as those based on expander graphs, do not come in convenient dual pairs. The hypergraph product is a different recipe that allows a pair of arbitrary classical codes to form the basis of a quantum code [1]. Crucially, when the hypergraph product uses families of classical low-density parity check (LDPC) codes, it leads to families of quantum-LDPC codes. The quantum-LDPC property eases the experimental difficulty of implementation and, combined with suitably growing distance, ensures the existence of an error correction threshold [2].

Manuscript received 10 August 2021; revised 29 March 2022; accepted 6 June 2022. Date of publication 17 June 2022; date of current version 15 September 2022. This work was supported by the Engineering and Physical Sciences Research Council under Grant EP/M024261/1 (E.T.C). *(Corresponding author: Armanda O. Quintavalle.)*

Armanda O. Quintavalle is with the Department of Physics and Astronomy, The University of Sheffield, Sheffield S3 7RH, U.K. (e-mail: armandaoq@gmail.com).

Earl T. Campbell is with the Department of Physics and Astronomy, The University of Sheffield, Sheffield S3 7RH, U.K., and also with Riverlane, Cambridge CB2 3BZ, U.K.

Communicated by P. Kiran Sarvepalli, Associate Editor for Quantum Information Theory.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2022.3184108.

Digital Object Identifier 10.1109/TIT.2022.3184108

Two of the most widely known quantum codes, the toric and planar surface codes, are hypergraph product codes that use the classical repetition code as their seed classical code. The decoding problem for the surface code can be recast as a minimum-weight perfect-matching problem, which is efficiently solved by the blossom algorithm [3], [4] and the union-find algorithm [5]. Another interesting class of hypergraph product codes uses classical expander codes as their seed, with the resulting offspring called quantum expander codes [6], which are quantum-LDPC codes achieving both constant rate and $\Omega(\sqrt{n})$ distance. The classical expander codes can be decoded by a very simple bit-flip algorithm discovered by Spiser and Spielman [7]. This inspired the small-set flip decoder for quantum expander codes, which follows a similar idea but is slightly modified, and has been shown to correct adversarial errors [6], stochastic errors [8] and also to operate as a single-shot decoder [9]. However, any binary linear code can be used as a seed to build hypergraph product codes. Using classical codes other than repetition and expander codes, for instance the semi-topological codes proposed in [10], yield a broad range of hypergraph product codes for which there is no general propose decoder that is proven to work across the whole code family. For classical LDPC codes, using a belief propagation decoder (BP) works well in practice but it cannot be used out of the box on quantum-LDPC codes. In fact whenever a decoding instance has more than one minimum weight solution, it is degenerate, BP does not converge and yields a decoding failure. Degeneracy is the quintessential feature of quantum codes and therefore some workarounds are needed to use BP on quantum-LDPC codes [11], [12]. The literature offers many examples of BP inspired decoders for quantum-LDPC codes which show an error correcting threshold [10], [13]–[17], however none of them come with a correctness proof. Recently, a union-find like decoder has been proposed to decode quantum-LDPC codes [18]. The authors in [18] prove that their union-find decoder corrects for all errors of weight up to a polynomial in the distance for three classes of quantum-LDPC codes: codes with linear confinement (see [19], [20]), D-dimensional hyperbolic codes and D-dimensional toric codes for $D \geq 3$. The decoder in [18] is therefore provably correct for adversarial noise, nonetheless a comprehensive investigation of its performance under stochastic noise is still missing.

Here we introduce the ReShape decoder for generic hypergraph product codes. Given a $[[n, k, d]]$ hypergraph product code built using classical codes with parity matrices δ_A and δ_B, we assume access to a minimum weight decoder for parity matrices δ_A, δ_B, δ_A^T and δ_B^T. The ReShape decoder calls these classical decoders as blackbox oracles without any modification or knowledge of their internal working,

and furthermore only requires $O(k)$ oracle calls, and only a polynomial amount of additional classical computation. Under these conditions we prove that ReShape works in the adversarial setting, correcting errors (up to stabilisers) of weight less than half the code distance. Therefore, ReShape lifts the classical decoders to the status of a quantum decoder, providing the first general purpose hypergraph product codes decoder proven to correct adversarial errors. Formally we prove:

Theorem 1: Any $[[n, k, d]]$ hypergraph product code constructed from the classical parity check matrices δ_A and δ_B can be successfully decoded from error of weight up to $(d-1)/2$ using $O(k)$ oracle calls to classical decoders for the seed matrices and their transpose plus $O(n^2)$ classical operations.

Theorem 1 though, does not state anything about stochastic noise or error correcting thresholds. Families of n-qubit hypergraph product codes have distance of at most $O(\sqrt{n})$ and so they are bad codes in the sense that the distance is sub-linear. However, given a stochastic noise model with each qubit affected independently with probability p, the typical error size will be pn. Thus, for $n > (d/2p)$, the most likely errors will not necessarily be corrected by ReShape and there is no guarantee that a threshold will be observed. Indeed, we implemented ReShape for several code families and found evidence that ReShape fails to provide a threshold (see Figure 4). A clear open problem is whether there exists a similar general lifting procedure, or modification of ReShape, for which one can prove good performance in the stochastic settings. Hence, if on one hand Theorem 1 provides a solution to the adversarial decoding problem for hypergraph product codes, on the other, a stronger, difficult and much longed-for result is desirable. Namely, the solution of the stochastic decoding problem for hypergraph product codes both on a theoretical level (proof of a threshold) and on a practical one (numerical observation of a high correcting threshold). Even so, ReShape still provides some improvement over state-of the-art BP and union-find like decoders for stochastic noise. First, ReShape comes with a proof of correctness, that BP lacks; second, the proof works for all errors up to the optimal value of $(d-1)/2$, whilst the modification of union-find proposed in [18] is provably correct only for errors of weight up to Ad^{α}, for some $A, \alpha > 0$ and $\alpha < 1$.

II. Preliminaries and Notation

A classical $[n, k, d]$ linear code is compactly described by its parity check matrix H. The matrix H is a binary matrix of size $m \times n$ such that the codespace $\mathcal{C}(H) \subseteq \mathbb{F}_2^n$ is described by:

$$\mathcal{C}(H) = \{v \in \mathbb{F}_2^n : Hv = 0\}. \tag{1}$$

The codespace $\mathcal{C}(H)$ has dimension $k = n - \mathrm{rank}(H)$ and distance d defined as:

$$d = \min\{|v| : v \in \mathcal{C}(H), v \neq 0\},$$

where $|v|$ is the Hamming weight of the binary vector v. Whenever the parity check matrix has columns and rows of small weight we say that it is a low density parity check (LDPC) matrix; when H has constant column and row weight w_c, w_r we shortly say that it is a (w_c, w_r)-matrix.

The classical decoding problem can be stated as: given a *syndrome* vector $s \in \mathbb{F}_2^m$, find the minimum weight solution $e \in \mathbb{F}_2^n$ to the equation

$$He = s. \tag{2}$$

It is easy to show that the optimal decoder for any classical linear code can correct errors of weight up to half the code distance (see, for instance, [21]).

A quantum $[[n, k, d]]$ stabiliser code [22] is a subspace of dimension 2^k of the Hilbert space $(\mathbb{C}^2)^{\otimes n}$. It is described as the common $+1$ eigenspace of its stabiliser group $\mathcal{S}$, an Abelian subgroup of the Pauli group $\mathcal{P}_n$ such that $-\mathbb{1} \notin \mathcal{S}$. The Pauli group on n qubits is the group generated by the n-fold tensor product of single qubit Pauli operators. The weight $|P|$ of a Pauli operator $P \in \mathcal{P}_n$ is the number of its non-identity factors. We indicate by $\mathcal{N}(\mathcal{S})$ the normaliser of $\mathcal{S}$ i.e. the group of Paulis which commute with the stabiliser group $\mathcal{S}$. Because $\mathcal{S} \subseteq \mathcal{N}(\mathcal{S})$, the quotient group

$$\mathcal{L} = \mathcal{N}(\mathcal{S})/\mathcal{S}$$

is well defined and referred to as *homology group*, see Appendix B. Elements $[P]$ of $\mathcal{L}$ are *homology classes*: equivalence classes with respect to the congruence modulo multiplication by stabiliser operators. Explicitly:

$$[P] = \{PS : S \in \mathcal{S}\}, \tag{3}$$

and for any Pauli P, its homology class $[P]$ is uniquely defined via Eq. (3). Importantly, each Pauli P such that $[P] \neq [\mathbb{1}]$ in $\mathcal{L}$ is an operator that preserves the codespace and has non-trivial action on it. We refer to such code operators modulo $\mathcal{S}$ as *logical Pauli operators*; with slight abuse of notation we write $P \in \mathcal{L}$, meaning $[P] \in \mathcal{L}$. Two logical operators P, Q are said to be *homologically equivalent*, or just equivalent, if and only if they belong to the same homology class i.e. by Eq. (3), if and only if $[P] = [Q]$. Importantly, for a code of dimension k, $\mathcal{L} \simeq \mathcal{P}_k$. The distance d of the code is the minimum weight of any non-trivial logical operator in $\mathcal{L}$. Any generating set of the stabiliser group $\mathcal{S}$ induces a syndrome map σ. Namely, if $\mathcal{S} = \langle S_1, \ldots, S_m \rangle$, the associated syndrome function σ maps any Pauli $P \in \mathcal{P}_n$ in a binary vector $s = (s_1, \ldots, s_m)^T \in \mathbb{F}_2^m$ such that $s_i = 0$ if and only if P commutes with S_i and 1 otherwise. We refer to the vector s as the *syndrome*. Conventionally, when considering a stabiliser code, it is always intended that a generating set $\{S_1, \ldots, S_m\}$ for the stabiliser group is chosen and with it a syndrome map. We say that a stabiliser code is LDPC if each S_i has low weight and each qubit is in the support of only a few generators.

The decoding problem for stabiliser codes can be stated as: given a syndrome vector $s \in \mathbb{F}_2^m$, find an operator $E_r \in \mathcal{P}_n$ such that (i) $\sigma(E_r) = s$ and (ii) $[E_r] = [E_{\min}]$, where E_{min} is a minimum weight operator with syndrome s. We call any operator that satisfies (i) a *valid* solution of the syndrome equation and operators for which both (i) and (ii) are true, *correct* solutions.

Pauli operators can be put into a one-to-one correspondence with binary vectors, if we discard the phase factor $\pm i$. In fact, any Pauli P can be written as:

$$P \propto X[v] \cdot Z[w], \\ = X^{v_1} \otimes \ldots \otimes X^{v_n} \cdot Z^{w_1} \otimes \ldots \otimes Z^{w_n}, \quad v, w \in \mathbb{F}_2^n$$

from which it follows:

$$(X[v]Z[w])(X[v']Z[w']) = \pm X[v+v']Z[w+w'], \tag{4}$$

and two operators commute if and only if

$$\langle v, w' \rangle + \langle v', w \rangle = 0 \mod 2 \tag{5}$$

and anti-commute otherwise. This correspondence between binary vectors and Pauli operators is particularly handy when dealing with CSS codes [23], [24]. CSS codes are stabiliser codes for which the stabiliser group can be generated by two disjoint sets $\mathcal{S}_x$ and $\mathcal{S}_z$ of X and Z type operators respectively. If $\mathcal{S}_x = \{X[v_1], \ldots X[v_{m_x}]\}$, $\mathcal{S}_z = \{Z[w_1], \ldots, Z[w_{m_z}]\}$ and we define H_X and H_Z as the matrices whose rows are the v_is and the w_is respectively, then the commutation relation on the stabilisers generators translate in to the binary constraint $H_X H_Z^T = 0$. Using Eq. (5), it is easy to show that the syndrome for a Pauli error $E = X[e_x]Z[e_z]$ is described by the two binary vectors $s_z = H_Z e_x$ and $s_x = H_X e_z$. Since these two linear equations are independent, we can treat the X-part and Z-part of the error separately. For CSS codes, we define the X-distance d_x as the minimum weight of an operator $X[v]$ which commutes with all the stabilisers in $\mathcal{S}_z$ but does not belong to the group generated by $\mathcal{S}_x$. Note that the weight of an operator $X[v]$ equates the Hamming weight $|v|$ of the vector v. Therefore, combining Eq. (4), Eq. (5) and the definition of d_x, we shortly say that d_x is the minimum weight of a vector v in $\ker H_Z$ which does not belong to the row span of H_X, i.e.

$$d_x := \min\{|v| : H_Z v = 0, v \notin \operatorname{Im} H_X^T\} \tag{6}$$

Similarly, d_z is the minimum weight of a vector in $\ker H_X$ not in $\operatorname{Im} H_Z^T$.

The Z-error decoding problem for CSS code can be stated as: given a syndrome vector $s \in \mathbb{F}_2^{m_x}$, find a valid and correct solution $e \in \mathbb{F}_2^n$ to the equation:

$$H_X e = s, \tag{7}$$

where e_r is valid if and only if $H_X e_r = s$ and it is correct if and only if it belongs to the homology class of the minimum weight operator with syndrome s. Because for an operator $Z[e]$ its weight equates the Hamming weight of the vector e, the Z-decoding problem for CSS codes can be reformulated exactly as done for the classical decoding problem in Eq. (2). Explicitly, given s, find the minimum weight solution to the linear equation $H_X e = s$. The X-decoding problem is derived from Eq. (7) by duality, exchanging the role of X and Z.

It goes without saying that, if any CSS code defines two classical parity check matrices, the converse is also true. Namely, starting from any two binary matrices H_1, H_2 such that $H_1 H_2^T = 0$, this defines a CSS code with $H_X = H_1$, $H_Z = H_2$. If the classical linear code with parity check H_i has parameters $[n, k_i, d_i]$, the associated quantum code has parameters $[[n, k_1 + k_2 - n, d_x, d_z]]$ where $d_x \geq d_2$ and $d_z \geq d_1$. A review on quantum codes can be found, for instance, in [25], [26].

In this article we focus on a sub-class of CSS codes, the hypergraph product codes [27]–[30]. We give a minimal description of these codes in Section III and we refer the reader to Appendix B for a more detailed presentation. We study some homology invariants for the logical operators of the hypergraph product codes in Section IV-A. These invariants are the algebraic core upon which we design a decoder for these codes, the ReShape decoder. We prove that ReShape is an efficient and correct decoder for adversarial noise in Section IV-B. We conclude with some consideration on the performance of ReShape under stochastic noise in Section IV-C.

III. Hypergraph Product Codes

We here present a bottom-up overview on hypergraph product codes. The purpose of this Section is dual: we both want to describe the hypergraph product codes with the least possible technical overhead and introduce the notation necessary to motivate and give an intuition for the results presented in Section IV. We refer the reader interested in the homology theory approach to Appendix B.

The most well-known example of hypergraph product code is the toric code and its variations [31], [32]. The toric code is conventionally represented by a square lattice where qubits sit on edges, X-stabilisers are identified with vertices and Z-stabilisers with faces. Since a square lattice has two kind of edges, vertical and horizontal edges, the first evident feature of this identification is that, accordingly, there are two type of qubits. The second is that each vertex/X-stabiliser uniquely identifies a row of horizontal edges and a column of vertical one, starting from the four ones that are incident to it. The third is that faces/Z-stabilisers, similarly to vertices, uniquely identify a column of horizontal edges and a row of vertical ones, starting from the four which lie on its boundary. Very similar attributes can be found in all the hypergraph product codes, as we now explain.

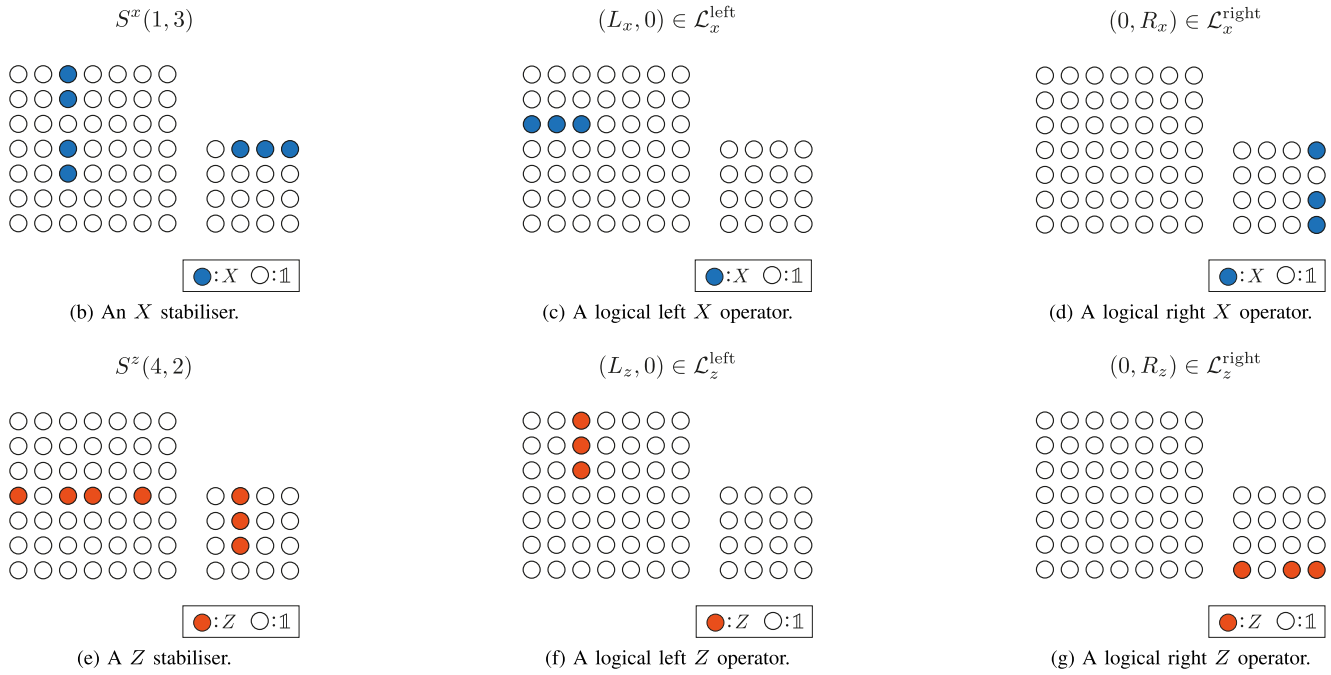
Consider two classical parity check matrices δ_A, δ_B of size $m_a \times n_a$ and $m_b \times n_b$; we indicate with $\mathcal{C}(\delta_A, \delta_B)$ their hypergraph product code and refer to the matrices δ_A and δ_B as *seed* matrices. The qubits of the code $\mathcal{C}(\delta_A, \delta_B)$ can be labelled as *left* and *right* qubits. Left qubits can be placed in a $n_a \times n_b$ grid and right qubits in a $m_a \times m_b$ grid, see Figure 1. Under this labelling, left and right qubits are uniquely identified by pair of indices (j_a, j_b) and (i_a, i_b) respectively, where j_a, j_b vary among the column indices of δ_A, δ_B while i_a, i_b vary among their row indices. Given a pair (L, R) of binary matrices, of size $n_a \times n_b$ and $m_a \times m_b$ respectively, we define the Z-operator:

$$Z(L, R) = \left(\bigotimes_{j_a, j_b} Z^{L_{j_a, j_b}} \right) \otimes \left(\bigotimes_{i_a, i_b} Z^{R_{i_a, i_b}} \right), \tag{8}$$

and similarly for X-operators. We refer to L as the left part of the operator and to R as its right part.

$$\delta = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

(a)



(b) An X stabiliser. (c) A logical left X operator. (d) A logical right X operator.

(e) A Z stabiliser. (f) A logical left Z operator. (g) A logical right Z operator.

Fig. 1. A graphical representation of the qubit space of the homological product code $\mathcal{C}(\delta_A,\delta_B)$ where $\delta_A=\delta_B=\delta$ and δ is a degenerate parity check matrix for the $[7,4,3]$ Hamming code reported in (1a). In (1b), ...,(1g), the two grids represent the left and right qubits respectively. One circle is drawn for every physical qubits of $\mathcal{C}(\delta_A,\delta_B)$. There are 7×7 left qubits and 4×4 right qubits. The support of an operator (L,R) on $\mathcal{C}(\delta_A,\delta_B)$ is represented by filling the corresponding circle: left qubit at position (j_a,j_b) is filled if and only if $L_{j_a,j_b}=1$; similarly the right qubit at position (i_a,i_b) is filled if and only if $R_{i_a,i_b}=1$. The code $\mathcal{C}(\delta_A,\delta_B)$ pictured has parameters $[[65,17,3,3]]$. It has 16 independent logical left operators and 1 logical right operators: $|\mathcal{L}_x^{\text{left}}|=|\mathcal{L}_z^{\text{left}}|=16$ and $|\mathcal{L}_x^{\text{right}}|=|\mathcal{L}_z^{\text{right}}|=1$.

The code $\mathcal{C}(\delta_A,\delta_B)$ has $m_a\times n_b$ X-stabiliser generators which can be indexed by (i_a,j_b). The X-stabiliser $S^x(i_a,j_b)$ has support contained in the union of the j_bth column of left qubits and the i_ath row of right qubits. More precisely,[1] it acts as $X[(\delta_A)_{i_a}]$ on the left qubits located at column j_b and as $X[(\delta_B)^{j_b}]$ on the right qubits located on row i_a, see Figure 1b. Using the X-version of Eq. (8), $S^x(i_a,j_b)$ is uniquely represented by the pair of matrices, $L=\delta_A^T E_{i_a j_b}$ and $R=E_{i_a j_b}\delta_B^T$, so that

$$S^x(i_a,j_b) := X(\delta_A^T E_{i_a j_b}, E_{i_a j_b}\delta_B^T),$$

where E_{i_a,j_b} is the all-zero $m_a\times n_b$ matrix but for the (i_a,j_b)th entry which is 1. From the characteristic 'cross' shape of the stabilisers generators $S^x(i_a,j_b)$, it follows that if (G_L,G_R) is an X-stabiliser for $\mathcal{C}(\delta_A,\delta_B)$, then (i) each column of G_L, as a vector in $\mathbb{F}_2^{n_a}$, belongs to $\operatorname{Im}\delta_A^T$ and (ii) each row of G_R, as a vector in $\mathbb{F}_2^{m_b}$, belongs to $\operatorname{Im}\delta_B$.

Similarly, Z-stabiliser generators are indexed by (j_a,i_b) for $1\le j_a\le n_a$ and $1\le i_b\le m_b$ and $S^z(j_a,i_b)$ is uniquely represented by the pair of matrices:

$$(L,R)=(E_{j_a i_b}\delta_B, \delta_A E_{j_a i_b}),$$

for $E_{j_a i_b}$ of size $n_a\times m_b$, with all entries 0 but for the (j_a,i_b)th entry which is 1.

The syndrome equation for hypergraph product codes can be derived combining Eq. (7) and the expression for the stabiliser generators. By Eq. (7), the ith bit of the syndrome vector $s\in\mathbb{F}_2^{m_x}$ equates the inner product between the ith X-stabiliser generator, which corresponds to the ith row of the matrix H_X, and the error vector. In the same way, by reshaping vectors into matrices (see Appendix B-A), the (i_a,j_b)th bit of the syndrome matrix $S\in\mathbb{F}_2^{m_a\times n_b}$ equates the inner product of the (i_a,j_b)th X-stabiliser generator and the error matrices (L,R):

$$(\delta_A)_{i_a}L + R(\delta_B)^{j_b} = S_{i_a,j_b},$$

and by linearity:

$$\sigma(L,R) := \delta_A L + R\delta_B. \qquad \text{(SE)}$$

It is easy to show that any Z-stabiliser has trivial X-syndrome, which is equivalent to X-stabilisers and Z-stabilisers commuting. As a consequence, $\mathcal{C}(\delta_A,\delta_B)$ is a well-defined CSS code.

[1]Here and in the following, for a $m\times n$ matrix δ we indicate by $\delta_i\in\mathbb{F}_2^n$ the transpose of its ith row, and by $\delta^j\in\mathbb{F}_2^m$ its jth column.

A minimal generating set of logical Z-operators for $\mathcal{C}(\delta_A, \delta_B)$ is given by:

$$\mathcal{L}_z \coloneqq \mathcal{L}_z^{\text{left}} \cup \mathcal{L}_z^{\text{right}} \tag{9}$$

where:

$$\begin{aligned}\mathcal{L}_z^{\text{left}} \coloneqq \big\{&(L,0) : L = k_a \cdot e_{j_b}^T,\\ &k_a \text{ varies among a basis of } \ker \delta_A,\\ &e_{j_b} \text{ varies among a basis of } (\operatorname{Im} \delta_B^T)^\bullet,\\ &|e_{j_b}| = 1\big\},\end{aligned}$$

and

$$\begin{aligned}\mathcal{L}_z^{\text{right}} \coloneqq \big\{&(0,R) : R = e_{i_a} \cdot \bar{k}_b^T,\\ &\bar{k}_b \text{ varies among a basis of } \ker \delta_B^T,\\ &e_{i_a} \text{ varies among a basis of } (\operatorname{Im} \delta_A)^\bullet,\\ &|e_{i_a}| = 1\big\}.\end{aligned}$$

Here, given a vector space $V \subseteq \mathbb{F}_2^n$, $V^\bullet$ denotes any space such that $V \oplus V^\bullet \simeq \mathbb{F}_2^n$. In particular, the space $V^\bullet$ is in general different from the orthogonal complement $V^\perp$ of the space V, see Appendix A for details. Similarly, a minimal generating set of logical X-operators is:

$$\mathcal{L}_x \coloneqq \mathcal{L}_x^{\text{left}} \cup \mathcal{L}_x^{\text{right}} \tag{10}$$

where:

$$\begin{aligned}\mathcal{L}_x^{\text{left}} \coloneqq \big\{&(L,0) : L = e_{j_a} \cdot k_b^T,\\ &k_b \text{ varies among a basis of } \ker \delta_B,\\ &e_{j_a} \text{ varies among a basis of } (\operatorname{Im} \delta_A^T)^\bullet,\\ &|e_{j_a}| = 1\big\},\end{aligned}$$

and

$$\begin{aligned}\mathcal{L}_x^{\text{right}} \coloneqq \big\{&(0,R) : R = \bar{k}_a \cdot e_{i_b}^T,\\ &\bar{k}_a \text{ varies among a basis of } \ker \delta_A^T,\\ &e_{i_b} \text{ varies among a basis of } (\operatorname{Im} \delta_B)^\bullet,\\ &|e_{i_b}| = 1\big\}.\end{aligned}$$

To sum up, the code $\mathcal{C}(\delta_A, \delta_B)$ is a CSS code with parameters $[[n, k, d_x, d_z]]$, where:

$$\begin{aligned} n &= n_a n_b + m_a m_b\\ k &= (n_a - \mathrm{rk}_a)(n_b - \mathrm{rk}_b) + (m_a - \mathrm{rk}_a)(m_b - \mathrm{rk}_b)\\ d_x &= \min\{d_a^T, d_b\}\\ d_z &= \min\{d_a, d_b^T\}\end{aligned}$$

for $\mathrm{rk}_\ell = \operatorname{rank}(\delta_\ell)$ and d_ℓ (resp. d_ℓ^T) distance of the classical code with parity check matrix δ_ℓ (resp. δ_ℓ^T), $\ell = A, B$. By convention, we define the distance of the trivial code $\{0\}$ to be ∞. In particular, whenever one or both seed matrices (or transpose) are full rank, one of the summands in the expression for k cancel out e.g. if δ_A or δ_B have full rank, then $k = (n_a - \mathrm{rk}_a)(n_b - \mathrm{rk}_b)$, $d_x = d_b$ and $d_z = d_a$.

The similarities in structure between general hypergraph product codes $\mathcal{C}(\delta_A, \delta_B)$ and the toric codes (with and without boundaries) should now be clear: the toric code with boundaries (resp. without) of lattice size L is just the hypergraph product code $\mathcal{C}(\delta_L, \delta_L)$ where δ_L is the full-rank $L-1 \times L$ (resp. non-full-rank $L \times L$) parity check matrix of the classical $[L, 1, L]$ repetition code, e.g. for $L = 3$:

$$\begin{pmatrix} 1 & 1 & 0\\ 0 & 1 & 1 \end{pmatrix}, \qquad \text{resp.} \qquad \begin{pmatrix} 1 & 1 & 0\\ 0 & 1 & 1\\ 1 & 0 & 1 \end{pmatrix}. \tag{11}$$

Left and right qubits correspond to vertical and horizontal edges; vertices and faces on the square lattice can be indexed in the natural way yielding the same stabiliser indexing of the general hypergraph product codes; string like (resp. loop like) logical operators correspond precisely to the left and right logical operators described above which have single column/single row support.

In what follows, we focus on Z-errors and their correction. With slight abuse of notation, we will refer to pair of matrices (L, R) as operators (and vice versa sometimes) where the identification is clear via Eq. (8). The corresponding results for X-errors are easily obtained by duality as per any CSS code. More precisely, by swapping the role of X and Z but also the role of rows and columns; alternatively, considering the code $\mathcal{C}(\delta_A^T, \delta_B^T)$, see Appendix B.

IV. Results

Here we present the ReShape decoder. The intuition behind ReShape is that we can look at hypergaph product codes as codes built combining (product) multiple copies of the same classical codes. As such, with due care, we can 'decouple' these copies and retrieve the original classical seed codes.

On a $[[n, k, d]]$ hypergraph product code $\mathcal{C}(\delta_A, \delta_B)$, ReShape works by splitting the decoding problem into k smaller classical decoding problems which can be solved using classical decoding algorithms for the seed matrices. In order to identify the k classical decoding problems, it applies a linear transformation, a change of basis, on the n dimensional codespace of $\mathcal{C}(\delta_A, \delta_B)$, yielding a canonical form for error operators. This canonical form exposes two important features of the codespace: the first one is that logical operators of $\mathcal{C}(\delta_A, \delta_B)$ are naturally partitioned into two sets, of left and right operators; the second is that the weight of each logical operator directly depends on the weight of the classical codewords of the seed codes. By writing an operator in its canonical form, we can immediately assess to which of the two classes it belongs and, via classical decoding, to which logical operator it is closest. Hence, we successfully detect and correct errors.

In this Section, we first proceed to study the algebraic invariants of the logical operators upon which the canonical form is defined. The correctness of ReShape, and so the proof of Theorem 1, strongly relies on the existence of these invariants. We detail the Reshape algorithm in Section IV-B and discuss its limitations in Section IV-C. All the proof of this Section are deferred to Appendix C.

A. Invariants

The characteristic shape of operators on the codespace of $\mathcal{C}(\delta_A, \delta_B)$ and the structure of its stabilisers and logical operators, induces a canonical form for Z-operators in $\mathcal{C}(\delta_A, \delta_B)$.

More precisely, by combining the construction outlined in Section III and the definition of complement of a vector subspace (see Appendix A) we have proven the following:

Proposition 1 (Canonical Form): Let (L, R) be a Z-operator on the codespace of $\mathcal{C}(\delta_A, \delta_B)$. For a vector space $V \subseteq \mathbb{F}_2^n$, we denote by $V^\bullet$ any space such that $V \oplus V^\bullet \simeq \mathbb{F}_2^n$, (see Appendix A). Then, for the operator (L, R), the left part L can be expressed as a sum of a *free part* M_L and a *logical part* O_L such that every row of M_L belongs to $\operatorname{Im}\delta_B^T$ and every row of O_L belongs to $(\operatorname{Im}\delta_B^T)^\bullet$. Similarly, the right part R can be expressed as a sum of a free part M_R and a logical part O_R such that every column of M_R belongs to $\operatorname{Im}\delta_A$ and every column of O_R belongs to $(\operatorname{Im}\delta_A)^\bullet$. Hence, for (L, R) holds:

$$(L, R) = (M_L + O_L, M_R + O_R). \tag{CF}$$

We refer to the writing given by Eq. (CF) as *canonical form* of the operator (L, R).

Crucially, as we detail in Appendix C, it is always possible to 'move' the support of the free part of an operator from the left qubits to the right qubits and vice versa, by adding stabilisers. Opposite is the situation for the logical part: the support of the logical part of an operator cannot be moved from the left to the right qubits without changing its homology class. These two observations justify the name free and logical part in the canonical form of a Z-operator on $\mathcal{C}(\delta_A, \delta_B)$. We refer to Figure 1 and 2 for a visual representation of the canonical form of a Z-operator on $\mathcal{C}(\delta_A, \delta_B)$. In Figures 1b and 1e we see stabiliser operators in their canonical form: their free part has support pictured, their logical part is 0. In Figures 1c, 1d, 1f, 1g we see logical operators in their canonical form: their free part is 0, whilst their logical part, pictured, has support contained in either a line or a column of one of the two grids of qubits. In Figure 2 we see a Z-operator whose free and logical part are both non trivial.

Given a Z-operator (L, R) we define its *row-column weight* as

Definition 1: Let $(L,\ R)$ be any Z-operator on the physical qubits of the code $\mathcal{C}(\delta_A, \delta_B)$. Its row-column weight is the integer pair:

$$\mathrm{wt}_{\mathrm{rc}}(L,\ R) := (\#\mathrm{row}(L),\ \#\mathrm{col}(R))$$

where

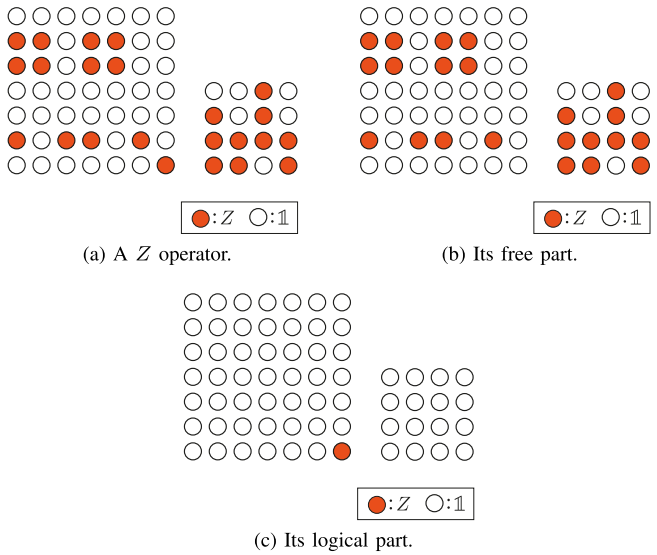
$$\begin{aligned}\mathrm{row}(L) &:= \{L_i \text{ row of } L \,:\, L_i \neq 0\},\\ \mathrm{col}(R) &:= \{R^j \text{ column of } R \,:\, R^j \neq 0\},\end{aligned}$$

and the hash symbol $\#$ denotes the cardinality of a set.

The primary significance of this novel notion of weight is explained by Proposition 2, which also represents a key result towards the construction of the ReShape decoder.

Proposition 2: If $(L,\ R)$ is a non-trivial logical Z-operator of $\mathcal{C}(\delta_A, \delta_B)$ then either $\#\mathrm{row}(L) \geq d_a$ or $\#\mathrm{col}(R) \geq d_b^T$ (or both).

Corollary 1 below further specifies the structure of logical Z-operators and it is easily derived from the proof of Proposition 2, which is deferred to Appendix C.



(a) A Z operator. (b) Its free part. (c) Its logical part.

Fig. 2. A graphical representation of a Z operator on the physical qubits of the code $\mathcal{C}(\delta_A, \delta_B)$ also represented in Figure 1. The filled circles in (2a) represent the support of the operator: its Hamming weight is 23 while its row-column weight is $(4, 4)$, see Definition 1. The operator in (2a) can be written as a sum of its *free* and its *logical* parts represented in (2b) and (2c), see Proposition (1). As per Definition 2 and figure (2c), the logical row-column weight of the operator in (2a) is $(1, 0)$.

Corollary 1: If (L, R) is a non-trivial logical Z-operator on $\mathcal{C}(\delta_A, \delta_B)$, at least one of the following hold:

(i) L has at least d_a rows which are not in $\operatorname{Im}\delta_B^T$ when seen as vectors in $\mathbb{F}_2^{n_b}$.

(ii) R has at least d_b^T columns which are not in $\operatorname{Im}\delta_A$ when seen as vectors of $\mathbb{F}_2^{m_a}$.

Proposition 2 and Corollary 1 naturally yield

Definition 2: Let (L, R) be a Z-operator on $\mathcal{C}(\delta_A, \delta_B)$. Its logical row-column weight is the integer pair

$$\mathrm{wt}_{\mathrm{rc}}^{\log}(L, R) := (\#\mathrm{row}_{\log}(L), \#\mathrm{col}_{\log}(R))$$

where

$$\begin{aligned}\mathrm{row}_{\log}(L) &:= \{L_i \text{ row of } L \,:\, L_i \notin \operatorname{Im}\delta_B^T\},\\ \mathrm{col}_{\log}(R) &:= \{R^j \text{ column of } R \,:\, R^j \notin \operatorname{Im}\delta_A\}.\end{aligned}$$

Equivalently, if (L, R) has canonical form given by

$$(L, R) = (O_L + M_L, O_R + M_R),$$

for its logical row-column weight holds:

$$\mathrm{wt}_{\mathrm{rc}}^{\log}(L, R) = \mathrm{wt}_{\mathrm{rc}}(O_L, O_R).$$

The pivotal property of the logical row-column weight is expressed by Proposition 3.

Proposition 3: The logical row-column weight of a Z-operator on $\mathcal{C}(\delta_A, \delta_B)$ is an invariant of its homology class.

Proposition 3 not only justifies the introduction of the notion of logical row-column weight but also constitutes the core resource upon which we prove the correctness of the ReShape decoder, which we now introduce.

B. The ReShape Decoder

An hypergraph product code $\mathcal{C}(\delta_A, \delta_B)$ is a CSS code and as such the decoding for X and Z error can be treated separately but in a symmetric way. Here we focus on Z-errors and therefore we measure a generating set of X-stabilisers. The Z-error decoding problem for $\mathcal{C}(\delta_A, \delta_B)$ can be stated as: given a $m_a \times n_b$ syndrome matrix S, find a valid and correct solution $(\tilde{L}, \tilde{R})$ to the equation:

$$S = \sigma(L,\, R) \coloneqq \delta_A L + R\delta_B, \tag{SE}$$

where $(\tilde{L}, \tilde{R})$ is *valid* if $\sigma(\tilde{L}, \tilde{R}) = S$ and it is *correct* if it belongs to the homology class of the minimum weight operator with syndrome S. Crucially, finding *a valid* solution (L, R) to Eq. (SE) is always possible by solving the linear system of equation where the parity check matrix of X stabilisers is the matrix of coefficients and the syndrome S is the constant term. The difficulties arise if we are interest in finding *a correct* solution to Eq. (SE).

The ReShape decoder for Z-errors is build upon two classical minimum weight decoding algorithms: $\mathscr{D}_{\delta_A}$ and $\mathscr{D}_{\delta_B^T}$. By this we mean that (i) the algorithms $\mathscr{D}_{\delta_A}$ and $\mathscr{D}_{\delta_B^T}$ are optimal decoders for the classical linear code with parity check matrix δ_A and δ_B^T respectively, and (ii) they solve the classical decoding problem of Eq. (2) for errors of weight up to $(d_a - 1)/2$ and $(d_b^T - 1)/2$ respectively. Reshape takes as input $\mathscr{D}_{\delta_A}$, $\mathscr{D}_{\delta_B^T}$, a syndrome matrix S and a valid solution $(L,\, R)$ of the Syndrome Equation (SE): $\sigma(L, R) = S$. Recall that a valid solution (L, R) can always be efficiently found either solving the associated linear system or querying a lookup table. It outputs a correct solution of (SE): an operator $(\tilde{L}, \tilde{R})$ homologically equivalent to the minimum weight operator $(L_{\min}, R_{\min})$ with syndrome S.

ReShape (Algorithm 1) works separately on the left part L and on the right part R of the operator $(L,\, R)$ and in fact it could be run in parallel (lines 1-10 and lines 11-20). Starting from a valid solution $(L,\, R)$, it minimises its logical row-column weight by minimizing $\#\mathrm{row}_{\log}(L)$ (lines 1-10) first and $\#\mathrm{col}_{\log}(R)$ after (lines 11-20). Because the logical row-column weight is an homology invariant for Z-operators (Proposition 3) and ReShape minimises it, this suffices to assure that ReShape is correct, as stated in Proposition 4. ReShape works on the left part L of the inputted valid solution (L, R) (lines 1-10) into two steps: Decode and Split. Each of these two steps exploits a characteristic feature of the Z-operators on the codespace of $\mathcal{C}(\delta_A, \delta_B)$:

(i) Split step: a Z-stabilizer (G_L, G_R) has left part G_L such that every row is in the image of δ_B^T;
(ii) Decode step: a logical Z-operator which acts non-trivially on the left qubits has a representative (L_z, R_z) such that at least one column of L_z is in $\ker \delta_A \setminus \{0\}$.

The Split and Decode steps are similarly performed on the right part R, as specified in lines 11 - 20 of the pseudocode in Algorithm 1. Again with reference to the left part as guide case, we now describe the Split and Decode steps in details and specify their computational cost. By extending this analysis to the right part, and thanks to Proposition 4, Theorem 1 is proved.

Algorithm 1 ReShape Decoder for Z-Errors

Input: Classical decoder $\mathscr{D}_{\delta_A}$ and $\mathscr{D}_{\delta_B^T}$. Syndrome matrix S and operator $(L,\, R)$ on $\mathcal{C}(\delta_A, \delta_B)$ s.t. $\sigma(L,\, R) = S$.

Output: Operator $(\tilde{L}, \tilde{R})$ on $\mathcal{C}(\delta_A, \delta_B)$ s.t. $\sigma(\tilde{L}, \tilde{R}) = S$ and $[\tilde{L}, \tilde{R}] = [L_{\min}, R_{\min}]$, where $(L_{\min}, R_{\min})$ is a minimum weight operator with syndrome S.

1: **for all** L_i rows of L **do**
2: Split: $L_i = m_i + \mu_i \in \mathrm{Im}\, \delta_B^T \oplus (\mathrm{Im}\, \delta_B^T)^\bullet$, as in (16)
3: **end for**
4: $M_L \leftarrow$ matrix whose rows are m_i
5: $O_L \leftarrow$ matrix whose rows are μ_i
6: **for all** O_L^j columns of O_L **do**
7: Decode: $\rho^j = \mathscr{D}_{\delta_A}(O_L^j)$
8: **end for**
9: $\tilde{L} \leftarrow$ matrix whose columns are ρ^j
10: $\tilde{L} \leftarrow \tilde{L} + O_L + M_L$
11: **for all** R^j columns of R **do**
12: Split: $R^j = m^j + \mu^j \in \mathrm{Im}\, \delta_A \oplus (\mathrm{Im}\, \delta_A)^\bullet$, as in (16)
13: **end for**
14: $M_R \leftarrow$ matrix whose columns are m^j
15: $O_R \leftarrow$ matrix whose columns are μ^j
16: **for all** $(O_R)_i$ rows of O_R **do**
17: Decode: $\rho_i = \mathscr{D}_{\delta_B^T}((O_R)_i)$
18: **end for**
19: $\tilde{R} \leftarrow$ matrix whose rows are ρ_i
20: $\tilde{R} \leftarrow \tilde{R} + R_L + M_R$
21: **return** $(\tilde{L}, \tilde{R})$

Let $(L,\, R)$ be any valid solution of (SE) given in input to ReShape.

(i) Split. First, in lines 1-3, L is written in its canonical form with respect to the basis described by Eq. (16):

$$L = M_L + O_L.$$

This operation has the cost of a change of basis over the vector space $\mathbb{F}_2^{n_b}$, namely from the canonical basis to the basis described by Eq. (16). A change of basis over a vector space is a linear operation that correspond to a multiplication by an invertible square matrix. Since we are interested in computing the image of this linear transformation for each of the n_a column vectors of L, this amount to the multiplication of an $n_a \times n_b$ and a $n_b \times n_b$ matrix. To sum up, the Split step of ReShape has cost $O(n_a n_b^2)$.

(ii) Decode. The second step performed by ReShape (lines 6-10) aims to minimise the logical row-column weight of $(L,\, R)$ by looking at non-homologically equivalent operators:

$$(L,\, R) + (L_z, 0),$$

as L_z varies in $\mathcal{L}_z^{\mathrm{left}}$. More precisely, ReShape exploits the canonical form of L computed at the previous step and scans through all the columns of its logical part O_L. By construction, any row $(O_L)_i$ of O_L belongs to the complement of $\mathrm{Im}\, \delta_B^T$. For this reason, $(O_L)_i$ can be written as the linear combination

of $k_b = n_b - \mathrm{rank}(\delta_B)$ unit vectors in $\mathbb{F}_2^{n_b}$ which does not belong to $\mathrm{Im}\,\delta_B^T$ i.e. k_b unit vectors which span $(\mathrm{Im}\,\delta_B^T)^\bullet$, see Appendix A. Importantly, when we stack these row vectors $(O_L)_i$ all together and consider the columns of the matrix O_L, we observe that O_L cannot have more than k_b non-zero columns. Each non-zero column of O_L is then treated as if it corresponded to a code-word of the classical code with parity check matrix δ_A (plus eventually a noise vector) and decoded individually using the classical minimum weight decoder $\mathscr{D}_{\delta_A}$ (line 7):

$$\mathscr{D}_{\delta_A}(O_L^j) = \rho^j \iff \rho^j \in \arg\min_{k \in \ker \delta_A} (|k + O_L^j|). \quad (12)$$

If the computational cost of the classical decoder $\mathscr{D}_{\delta_A}$ is $O(c_a)$, the computational cost of the second step of ReShape is $O(k_b c_a)$.

The Split and Decode steps described for the left part are replicated, with opportune modifications, for the right part. To be exact, if one or both δ_A and δ_B are full rank, then the right part does not encode any logical operator so the algorithm terminates.[2]

Proposition 4 below ensures that the recovery operator $(\tilde{L}, \tilde{R})$ found by ReShape is a correct solution of (SE), as long as the classical decoders $\mathscr{D}_{\delta_A}$ and $\mathscr{D}_{\delta_B^T}$ succeed.

Proposition 4: Let S be an X-syndrome matrix for $\mathcal{C}(\delta_A, \delta_B)$ and (L, R) any valid solution to the Syndrome Equation:

$$\sigma(L,\, R) = S. \quad \text{(SE)}$$

Suppose that the minimum weight operator $(L_{\min},\, R_{\min})$ with syndrome S has $(d_a/2, d_b^T/2)$-bounded logical row-column weight i.e.

$$\mathrm{wt}_{\mathrm{rc}}^{\log}(L_{\min},\, R_{\min}) = (\#\mathrm{row}_{\log}(L_{\min}), \#\mathrm{col}_{\log}(R_{\min})),$$

is such that

$$\#\mathrm{row}_{\log}(L_{\min}) < \frac{d_a}{2} \quad \text{and} \quad \#\mathrm{col}_{\log}(R_{\min}) < \frac{d_b^T}{2}. \quad (13)$$

Then, on input $\mathscr{D}_{\delta_A}$, $\mathscr{D}_{\delta_B^T}$, S and $(L,\, R)$, ReShape outputs a correct solution $(\tilde{L},\, \tilde{R})$ of (SE), provided that the classical decoders $\mathscr{D}_{\delta_A}$, $\mathscr{D}_{\delta_B^T}$ succeed. In other words, the solution $(\tilde{L}, \tilde{R})$ found by ReShape is in the same homology class as the minimum weight operator with syndrome S:

$$[L_{\min},\, R_{\min}] = [\tilde{L},\, \tilde{R}].$$

It is important to note that the condition (13) on the weight of the original error is on its row-column weight, while usually decoding success is assessed depending on the weight of an operator, meaning the number of its non-identity factors. Obviously, for any operator $(L,\, R)$ it holds:

$$\#row(L) \leq |L| \quad \text{and} \quad \#col(R) \leq |R|.$$

As a consequence, Proposition 4 entails that ReShape succeeds in correcting any Z-error of weight up to half the code distance $d_z = \min\{d_A, d_B^T\}$. Combining this with the cost analysis of the Split and Decode steps detailed above, gives a proof of Theorem 1.

It is worth to observe that actually ReShape can correct errors of weight strictly bigger than half the code distance, as long as they are not too 'spread'. In fact, whenever an error is homologically equivalent to an operator (L, R) such that L has 'few' non-zero rows and R has 'few' non-zero columns, ReShape succeeds. Formally, because by definition:

$$\#\mathrm{row}(L) \geq \#\mathrm{row}_{\log}(L) \quad \text{and} \quad \#\mathrm{col}(R) \geq \#\mathrm{col}_{\log}(R).$$

Proposition 4 yields

Corollary 2: Provided that the classical decoders succeed, ReShape successfully corrects any Z-error (L, R) with bounded row-column weight:

$$\#\mathrm{row}(L) < \frac{d_a}{2} \quad \text{and} \quad \#\mathrm{col}(R) < \frac{d_b^T}{2}.$$

To sum up, ReShape successfully solves the decoding problem for any hypergraph product code requiring only k oracle calls to a classical decoder for the seed matrices, where k is the logical dimension of the code. Furthermore, it is able to correct for a vast class of errors of weight strictly bigger than half the code distance, provided that they have a 'good' shape. Here by 'good' we mean errors of low logical column-row weight but arbitrary Hamming weight as for instance the Z-operator pictured in Figure 2, that has Hamming weight 23 but logical row-column weight $(1, 0)$ and would therefore be successfully corrected by the ReShape decoder.

The next Section focuses on what happens when we cannot control the shape of the errors but we assume that the probability of a given error to occur decays exponentially in its weight.

C. ReShape for Stochastic Noise

Up till now, we have focused on the adversarial noise model: errors on qubits are always correctable because we assume they have weight less than half the code distance. In real systems though, this is rarely the case and it is more faithful to assume that errors are sampled accordingly to a local stochastic noise model, where qubits errors have arbitrary location but the probability of a given error decays exponentially in its weight [3]. More precisely the probability of a Pauli error $E \in \mathcal{P}_n$ to occur is given by:

$$\mathbb{P}(E) = p^{|E|}(1-p)^{n-|E|}, \quad (14)$$

meaning that Pauli errors on each of the n qubits are independent and identically distributed. Under the binomial distribution associated to Eq. (14), the expected error weight on the encoded state is pn. Because the best possible distance scaling for the hypergraph product codes is $\sim \sqrt{n}$ (when the classical seed codes have linear distance), as n increases, we eventually find $pn > \sqrt{n}/2 \sim d/2$. Nonetheless, it is well known that LDPC hypergraph product codes do have a positive error correcting threshold [2]. A family of codes has threshold $p_{\mathrm{th}} > 0$ if, for noise rate below p_{th}, non-correctable errors that destroy the logical information occur

[2]In fact, as per Eq. (9) and Eq. (10), if $\mathrm{rank}(\delta_A) = m_a$ or $\mathrm{rank}(\delta_B) = m_b$, then $\ker \delta_A^T = (\mathrm{Im}\,\delta_A)^\bullet = \{0\}$ or $\ker \delta_B^T = (\mathrm{Im}\,\delta_B)^\bullet = \{0\}$ and so $\mathcal{L}_x^{\mathrm{right}} = \mathcal{L}_z^{\mathrm{right}}$.

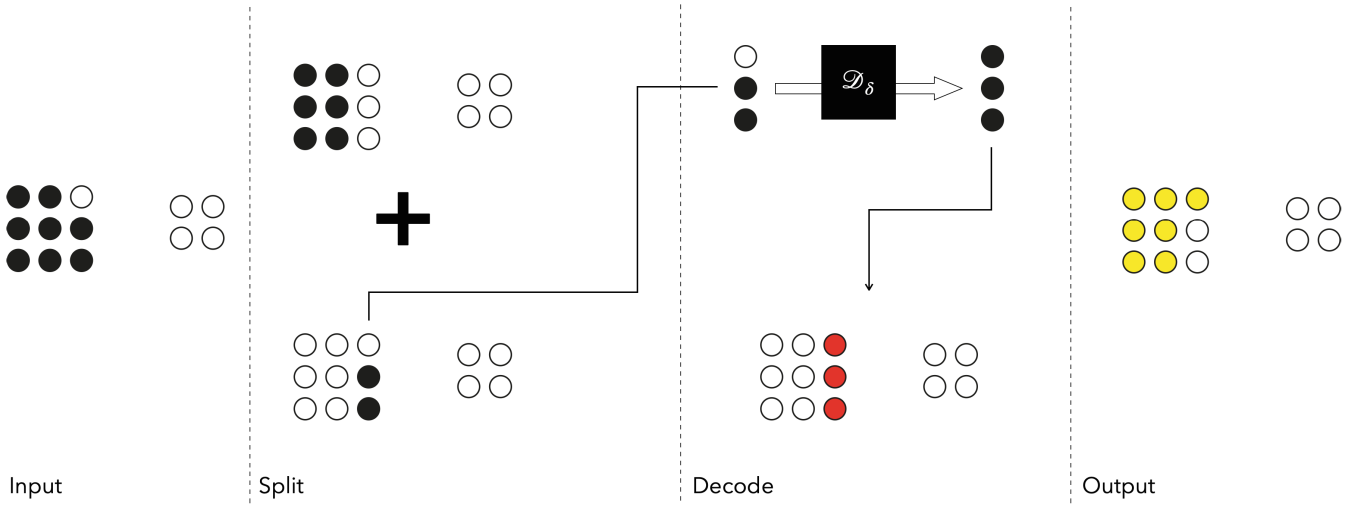


Fig. 3. Graphical representation of one instance of ReShape for Z-errors. The code considered is the planar code of distance 3 (toric code with boundaries) or, equivalently, the $[[13, 1, 3]]$ hypergraph product code $\mathcal{C}(\delta, \delta)$ for δ full-rank parity check matrix of the distance-3 repetition code i.e. leftmost matrix in (11). We use the same graphical representation used in Figure 1. A minimum weight logical Z operator for $\mathcal{C}(\delta, \delta)$ can be chosen to have support on all the qubits of a column of left qubits (Decode, bottom grid of qubits, support in red). The row span of δ consists of all vectors in $\mathbb{F}_2^3$ of even weight, hence for a generic Z-operator on $\mathcal{C}(\delta, \delta)$, all the rows of left qubits that have an even number of filled qubits belongs to its free part and do not contribute to its logical row-column weight. Since δ is full-rank, its column span is the whole space $\mathbb{F}_2^2$, and therefore a column displaying any choice of filled qubits is in the image of δ^T. As such, there is no contribution to the logical-row column weight from the right part of the operator. In particular, there is no need to run the ReShape decoder on the right part: Algorithm 1 will not execute lines 11 - 20.
The figure is divided into to four sectors, one for each stage of the decoding cycle: Input, Split, Decode and Output.
Input: the valid solution in input (L, R) has support represented by the black qubits. The operator (L, R) has Hamming weight 8.
Split - Algorithm 1, lines 1 - 3: (L, R) is written as sum of its free part (M_L, M_R) (top); and its logical part (O_L, O_R) (bottom). For what said on the image of δ, the free part M_L of L is a matrix whose rows have all even Hamming weight. Since O_L has 2 non-zero rows, (L, R) has logical row-column weight $\mathrm{wt}_{\mathrm{rc}}^{\log}(L, R) = 2$.
Decode - Algorithm 1, lines 1 - 9: the non-zero column $(0, 1, 1)^T$ of the logical part O_L of L is given in input to a decoder $\mathscr{D}_\delta$ for the classical distance-3 repetition code. The solution found is $(1, 1, 1)^T$, represented by the single column of black bits on the top. This solution is plugged in the hypergraph product code and yields a logical operator correction represented by the operator at the bottom with support on the red qubits.
Output - Algorithm 1, line 10: the output solution $(\tilde{L}, \tilde{R})$ is obtained by adding the input operator (L, R) and the operator found in the Decode step with support on the red qubits. The support of $(\tilde{L}, \tilde{R})$ is represented by the yellow qubits.
We note that the solution found $(\tilde{L}, \tilde{R})$ has Hamming weight 7 and, by observing that only the first rows has odd weight, we deduce that its logical row-column weight is 1. It is easy to verify that $(\tilde{L}, \tilde{R})$ is indeed homologically equivalent to the minimum weight solution $(\hat{L}_{1,3}, 0)$ where $\hat{L}_{1,3} \in \mathbb{F}_2^{3\times 3}$ is the matrix with all zeros but for the $(1, 3)$-th entry which is 1. In fact, $(\tilde{L}, \tilde{R}) = (\hat{L}_{1,3}, 0) + S^z(1, 1) + S^z(2, 1) + S^z(3, 1)$ thus, by (3), $[\tilde{L}, \tilde{R}] = [\hat{L}_{1,3}, 0]$.

with probability $p_{\mathrm{non-correctable}}$ which decays exponentially in the system size [2], [8], [33]:

$$p_{\mathrm{non-correctable}} \propto \left(\frac{p}{p_{th}}\right)^{\alpha d^\beta} \tag{15}$$

for some $\alpha, \beta > 0$. It is important to stress that Eq. (15) does not contrast with the fact that the typical error under the stochastic noise model will have weight pn. Instead, Eq. (15) entails that, among all the errors sampled, the non-correctable ones are only a small fraction. Beyond the theoretical threshold that Kovalev and Pryadko proved in [2], the literature offers several numerical evidence of decoders for hypergraph product or related families of codes which exhibit a threshold. Nonetheless these decoders either lack a correctness proof, e.g. BP in [10], [15], or need some additional constraints on the seed matrices, e.g. expander codes with small-set flip decoder [8], or augmented surface codes with the union-find decoder in [34].

On the contrary, for any choice of the seed matrices in the hypergraph product, ReShape is provably correct for adversarial errors. Not surprisingly though, ReShape does not show a threshold and a possible intuition for its anti-threshold behaviour is the following.

If we contrast Reshape with pairs of LDPC codes families and decoders which exhibit a threshold, such as expander codes with the small-set flip decoder [8] or hypergraph product codes with BP [10], [15], a feature of difference is the 'locality' of the decoding algorithms. Loosely, we say that a decoding algorithm is local if errors affecting distant regions on the qubit graph are dealt with separately and independently. We stress that a decoder's locality is a feature of the algorithm and it is not related to the locality of the code's stabiliser generators. A code can have local stabilisers, meaning that for a given layout of qubits in the space, stabiliser generators only involve qubits in a limited area, and yet be equipped with a non-local decoding algorithm. Indeed, ReShape is such a decoder. It is a non-local decoder that can be used on the very much local planar code. Locality of the decoding algorithm is relevant because local stochastic errors tend to form small disjoint clusters on the qubit graph which do not destroy the logical information as long as they are (1) small enough (2) sufficiently far apart. Therefore, if a decoder manages to mimic the error cluster distribution on the qubit graph and finds recovery operators accordingly, then it is likely to preserve the logical information and show a threshold. ReShape, on the other hand, has a deeply global nature.

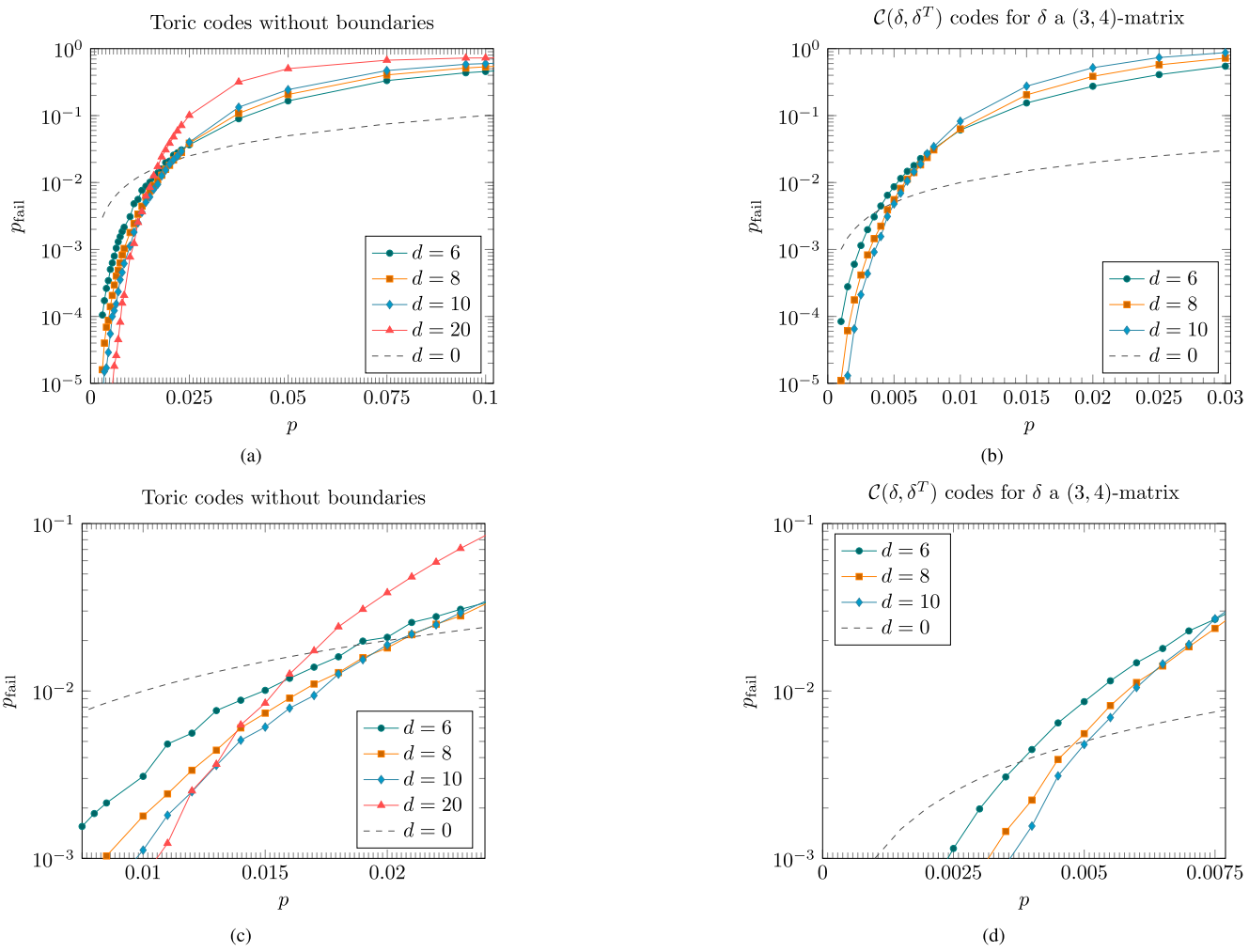


Fig. 4. Anti-threshold behaviour of ReShape on two families of hypergraph product codes. For both families, we plot the failure probability p_{fail} as a function of the phase-flip noise rate p for codes with Z-distance d. All data points are generated with at least 10^4 Monte Carlo trials. (a) and (c): The toric code without boundaries. (b) and (d): A family of hypergraph product codes $\mathcal{C}(\delta, \delta^T)$ where δ is a full-rank $(3,4)$-matrix randomly generated, see [10].

The Split step *groups* all the clusters of flipped qubits scattered across the qubit graph in a small pre-assigned region; a recovery is then chosen (Decode step) based on the syndrome information in this pre-assigned region. If we take the planar code as an example (see Figure 3), the Split step groups the error (and the syndrome) weight on one column of the left qubits. The subsequent Decode step decodes that column and finds a recovery operator with supported on the column. Because for the planar code a logical Z-operator can be chosen to have support on only one column of the left qubits, this procedure can easily destroy the logical information.

Our intuition on the performance of ReShape under stochastic noise finds confirmation in the plots reported in Figure 4. Even if at first sight the plots in Figure 4a and 4b could indicate the presence of a very low threshold (below 1%), a closer analysis suggests that this is not the case. In fact, as d increases, the crossing point between the dashed curve labelled $d = 0$ and the d-curves slips leftwards. Since the dashed curve is the locus of points where the failure probability p_{fail} equates the noise rate p, it corresponds to the case of no encoding i.e. $d = 0$. The common crossing point, in other words, represents the *pseudo-threshold* of the code [35]. Importantly, if a code family has a threshold p_{th} in the sense of Eq. (15), then all the codes of the family crosses the curve $d = 0$ at the same point of coordinates $(p_{\text{th}}, p_{\text{th}})$. Figure 4c clearly illustrates this left slipping phenomenon for the toric codes without boundaries. For close distances $d = 6, 8, 10$, there it seems to be a common crossing point with the $d = 0$ curve. However, the crossing point lowers if we increase d more substantially, e.g. $d = 20$. The situation appears less clear in Figure 4d because the pseudo-threshold seems to increase with the distance of the code. Still though, there is no common crossing point of the three curves; besides, we would expect the same trend as the one observed for the toric codes if codes of bigger distance were considered.

In conclusion, ReShape is not suited to tackle stochastic errors on $[[n, k, d]]$ code in the regime where typical errors have weight exceeding $d/2$.

V. Conclusion and Outlook

In this paper we determined some important homology invariants of hypergraph product codes. Exploiting these invariants, we designed the ReShape decoder. ReShape is the first decoder to efficiently decode for all errors up to half the code distance, across the whole spectrum of hypergraph product codes.

We foresee two natural extensions of this work. The first is to adapt ReShape for it to work in the stochastic noise model settings. Because ReShape actually succeeds in correcting errors of weight substantially bigger than $(d-1)/2$ (namely it corrects error of weight as big as $\sim d^2$, when they have the right shape!), this gives us some hope that ReShape would work under stochastic noise if paired with the right clustering technique. For instance, something on the line of the clustering methods used in the renormalisation group or the union-find decoders [5], [18], [36], [37].

The second is to find the corresponding invariants for other families of homological product codes. Specifically, for the codes in [38], which have 'rectangular' shaped logical operators instead of 'string' like as the standard hypergraph product codes here studied; or the balanced product codes proposed in [39]. Once found, the right invariants could be plugged-in an appropriately modified version of ReShape and yield a provable correct decoder for these class of codes too.

Appendix A
Linear Algebra: Space Complement

In this Appendix we review some known linear algebra facts that we use in our proofs. We refer the reader for instance to [40], [41] for a detailed presentation on the topic.

Consider a $m \times n$ binary matrix δ. If $\mathrm{rank}(\delta) = \mathrm{rk}$ then we can choose binary vectors $v_1, \ldots, v_{\mathrm{rk}}$ in $\mathbb{F}_2^m$ whose span is $\mathrm{Im}\,\delta$:

$$\mathrm{Im}\,\delta = \langle v_1, \ldots, v_{\mathrm{rk}} \rangle.$$

Let $\mathbb{1}$ be the $m \times m$ identity matrix. Perform Gaussian reduction on the $(\mathrm{rk}+m)$-row matrix M:

$$M = \begin{pmatrix} v_1 \\ \vdots \\ v_{\mathrm{rk}} \\ \hline \mathbb{1} \end{pmatrix}.$$

By selecting the pivot rows, we obtain a basis of $\mathbb{F}_2^m$ of the form:

$$\{v_1, \ldots, v_{\mathrm{rk}}, f_{\mathrm{rk}+1}, \ldots, f_m\}, \tag{16}$$

where the f_i are unit vectors. Letting:

$$(\mathrm{Im}\,\delta)^\bullet := \langle f_{\mathrm{rk}+1}, \ldots, f_m \rangle,$$

we have:

$$\mathbb{F}^m = (\mathrm{Im}\,\delta) \oplus (\mathrm{Im}\,\delta)^\bullet. \tag{17}$$

We refer to the space $(\mathrm{Im}\,\delta)^\bullet$ as complement of the space $\mathrm{Im}\,\delta$. We remark that the complement $V^\bullet$ is not equal to the orthogonal complement $V^\perp$. To see how this is the case, consider

$$V = \langle \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \rangle.$$

Then the spaces $V^\bullet$ and $V^\perp$ can be chosen as

$$V^\bullet = \langle \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \rangle, \qquad V^\perp = \langle \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \rangle.$$

In particular, $V^\perp \subseteq V$ while $V^\bullet \cap V = \{0\}$.

Appendix B
Hypergraph Product Codes

CSS codes can be easily described in terms of homology theory [31], [42], [43] via the identification of the objects of the code with a chain complex [44]. For our purposes, a length ℓ chain complex is an object described by a sequence of $\ell+1$ vector spaces $\{\mathcal{C}_i\}_i$ over $\mathbb{F}_2$ and ℓ binary matrices $\{\partial_i : \mathcal{C}_i \longrightarrow \mathcal{C}_{i+1}\}_i$ such that, for each i, $\partial_i \partial_{i-1} = 0$. In the following, we use the symbol ∂ to indicate the maps of a chain complex of length $\ell > 1$ and the symbol δ to indicate the map of a chain complex of length 1. Given a chain complex $\mathfrak{C}$:

$$\mathcal{C}_{-1} \xrightarrow{\partial_{-1}} \mathcal{C}_0 \xrightarrow{\partial_0} \mathcal{C}_1, \tag{$\mathfrak{C}$}$$

we can define a CSS code $\mathcal{C}$ by equating:

$$H_Z = \partial_{-1}^T, \quad H_X = \partial_0.$$

Since $\partial_0 \partial_{-1} = 0$ by construction, X-type and Z-type operators do commute i.e. $H_X \cdot H_Z^T = 0$ and the code $\mathcal{C}$ associated to the chain complex ($\mathfrak{C}$) is well defined. The code $\mathcal{C}$ has length $n = \dim(\mathcal{C}_0)$ and its dimension k equates to the dimension of the 0th homology group $\mathcal{H}_0 = \ker \partial_0 / \mathrm{Im}\,\partial_{-1}$ or, equivalently, to the dimension of the 0th co-homology group $\mathcal{H}_0^* = \ker \partial_{-1} / \mathrm{Im}\,\partial_0$. Its Z-distance and X-distance are given by the minimum Hamming weight of any representative of a non-zero element in $\mathcal{H}_0$ and $\mathcal{H}_0^*$ respectively:

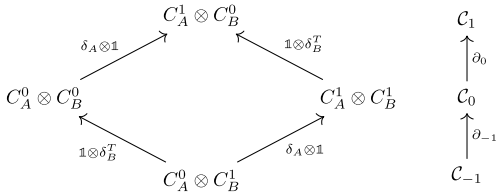
$$d_z = \min_{v \in \mathbb{F}_2^n} \{|v| : [v] \in \mathcal{H}_0, [v] \neq 0\},$$
$$d_x = \min_{v \in \mathbb{F}_2^n} \{|v| : [v] \in \mathcal{H}_0^*, [v] \neq 0\}.$$

An hypergraph product code $\mathcal{C}(\delta_A, \delta_B)$, which is a CSS code, can be easily defined in terms of product of chain complexes. Consider the two length-1 chain complexes defined by the seed matrices δ_A and δ_B:

$$C_A^0 \xrightarrow{\delta_A} C_A^1,$$
$$C_B^0 \xrightarrow{\delta_B} C_B^1.$$

We define their homological product as follows. Take the tensor product spaces $\mathcal{C}_{-1} = C_A^0 \otimes C_B^1$ and $\mathcal{C}_1 = C_A^1 \otimes C_B^0$ and the direct sum space $\mathcal{C}_0 = C_A^0 \otimes C_B^0 \oplus C_A^1 \otimes C_B^1$.

The chain complex $\mathfrak{C}_{A,B}$:



$(\mathfrak{C}_{A,B})$

$$\partial_{-1} = \begin{pmatrix} \mathbb{1} \otimes \delta_B, & \delta_A^T \otimes \mathbb{1} \end{pmatrix}^T, \partial_0 = \begin{pmatrix} \delta_A \otimes \mathbb{1}, & \mathbb{1} \otimes \delta_B^T \end{pmatrix}$$

is well defined. In fact:

$$\partial_{-1}\partial_0 = \delta_A \otimes \delta_B^T + \delta_A \otimes \delta_B^T = 0.$$

Therefore, the complex $\mathfrak{C}_{A,B}$ defines a valid CSS code, which we denote by $\mathcal{C}(\delta_A, \delta_B)$ and refer to as the hypergraph product code of the seed matrices δ_A and δ_B. If the classical code with parity check $\delta_\ell, \delta_\ell^T$ has parameters $[n_\ell, k_\ell, d_\ell]$ and $[n_\ell^T, k_\ell^T, d_\ell^T]$ respectively ($\ell = A, B$) then the hypergraph product code $\mathcal{C}(\delta_A, \delta_B)$ has parameters:

$$[[n_a n_b + n_a^T n_b^T,\, k_a k_b + k_a^T k_b^T,\, d_x,\, d_z]],$$

where $d_x = \min\{d_a^T, d_b\}$ and $d_z = \min\{d_a, d_b^T\}$, see [43].

A. Reshaping of Vectors

One tool we make extensive use of, and from which our decoder takes its name, is the reshaping of vectors of a two-fold tensor product space into matrices (see, for instance, [43], [45]). Consider a basis $\mathcal{B}$ of the vector space $\mathbb{F}_2^{n_1} \otimes \mathbb{F}_2^{n_2}$:

$$\mathcal{B} = \{a_i \otimes b_j \mid i = 1, \ldots, n_1 \text{ and } j = 1, \ldots, n_2\}.$$

Then any $v \in \mathbb{F}_2^{n_1} \otimes \mathbb{F}_2^{n_2}$ can be written as:

$$v = \sum_{a_i \otimes b_j \in \mathcal{B}} v_{ij}(a_i \otimes b_j),$$

for some $v_{ij} \in \mathbb{F}_2$. We call the $n_1 \times n_2$ matrix V with entries v_{ij} the reshaping of the vector v. By this identification, if φ, θ are respectively $m_1 \times n_1$ and $m_2 \times n_2$ matrices, then $(\varphi \otimes \theta)(V) = \varphi V \theta^T$. The inner product between $u \otimes w$ and v in $\mathbb{F}_2^{n_1} \otimes \mathbb{F}_2^{n_2}$ can be computed as

$$\langle u \otimes w, v \rangle = u^T V w. \tag{18}$$

As we here detail, the identification of operators on the code space $\mathcal{C}(\delta_A, \delta_B)$ with pairs of binary matrices that we used in the main text is rigorously justified by the reshaping of vectors into matrices. With slight abuse of notation, we refer to binary vectors and binary matrices as operators and vice versa, where the identification is clear via Eq. (8).

B. Graphical Representation

Physical qubits of the code $\mathcal{C}(\delta_A, \delta_B)$ are in one-to-one correspondence with basis elements of the space $\mathcal{C}_0$. If $\{e_{j_a}\}_{j_a}$, $\{e_{j_b}\}_{j_b}$, $\{e_{i_a}\}_{i_a}$, $\{e_{i_b}\}_{i_b}$ are bases of the spaces $C_A^0, C_B^0, C_A^1, C_B^1$ of dimension n_a, n_b, m_a, m_b respectively, then the union of the two sets

$$\mathcal{B}_L \coloneqq \{(e_{j_a} \otimes e_{j_b},\, 0)\}$$
$$\mathcal{B}_R \coloneqq \{(0,\, e_{i_a} \otimes e_{i_b})\}$$

is a basis of $\mathcal{C}_0$. We refer to qubits associated to elements in $\mathcal{B}_L$, or its span, as *left* qubits and to those associated to $\mathcal{B}_R$, or its span, as *right* qubits. Since qubit operators are vectors in $\mathcal{C}_0$, by reshaping, they can be identified with pairs of matrices $(L,\, R)$ where L has size $n_a \times n_b$ and R has size $m_a \times m_b$; in particular, L acts on the left qubits while R acts on the right qubits.

A Z-stabilizer for the code associated to the complex $\mathfrak{C}_{A,B}$ is any vector in $\operatorname{Im}\partial_{-1}$. A generating set for Z-stabilizers is:

$$\mathcal{S}_z \coloneqq \{\partial_{-1}(e_{j_a} \otimes e_{i_b})\}_{j_a, i_b}$$

where e_{j_a} and e_{i_b} are unit vectors of C_A^0 and C_B^1 respectively, i.e. they are a basis of the two spaces. Let $E_{j_a i_b} \in C_A^0 \otimes C_B^1$ be the reshaping of $(e_{j_a} \otimes e_{i_b})$, i.e. it is the matrix with all zeros entries but for the (j_a, i_b)-th entry which is 1. The reshape of $\partial_{-1}(e_{j_a} \otimes e_{i_b})$ is then given by the pair of matrices:

$$(L, R) = (E_{j_a i_b}\delta_B, \delta_A E_{j_a i_b}).$$

Logical Z-operators are vectors in $\ker\partial_0$ which are not in $\operatorname{Im}\partial_{-1}$. Specifically, a minimal generating set of logical Z-operators is given by [30]:

$$\hat{\mathcal{L}}_z \coloneqq \hat{\mathcal{L}}_z^{\text{left}} \cup \hat{\mathcal{L}}_z^{\text{right}} \tag{19}$$

where

$$\hat{\mathcal{L}}_z^{\text{left}} \coloneqq \big\{(k_a \otimes e_{j_b}, 0) : k_a \text{ varies among a basis of } \ker\delta_A, \; |e_{j_b}| = 1 \text{ and it varies among a basis of } \operatorname{Im}(\delta_B^T)^\bullet\big\},$$

and

$$\hat{\mathcal{L}}_z^{\text{right}} \coloneqq \big\{(0, e_{i_a} \otimes \bar{k}_b) : |e_{i_a}| = 1 \text{ and it varies among a basis of } \operatorname{Im}(\delta_A)^\bullet, \; \bar{k}_b \text{ varies among a basis of } \ker\delta_B^T\big\}.$$

The reshaping of vectors in $\hat{\mathcal{L}}_z$ gives the set $\mathcal{L}_z$ of Eq. (9) in the main text. The vector version of logical X-operators is likewise obtained from the set of matrices $\mathcal{L}_x$ of Eq. (10).

APPENDIX C
PROOFS

This Section contains all the proofs of the statements made in the main text.

Broadly speaking, in this work we wanted to characterize Z-errors operators on the codespace of $\mathcal{C}(\delta_A, \delta_B)$ associated to the chain complex $\mathfrak{C}_{A,B}$. In order to do so, we first studied the logical Z-operators of $\mathcal{C}(\delta_A, \delta_B)$ and introduced a canonical form for them. From homology theory, we know that non-trivial logical Z-operators are associated to vectors in $\ker\partial_0$ which do not belong to $\operatorname{Im}\partial_{-1}$. Lemma 1 below describes all the vectors in $\ker\partial_0$.

Lemma 1: Let $(L, R) \in \mathcal{C}_0$ be in $\ker\partial_0$, then:

$$L \in \ker\delta_A \otimes C_B^0 + C_A^0 \otimes \operatorname{Im}\delta_B^T,$$
$$R \in C_A^1 \otimes \ker\delta_B^T + \operatorname{Im}\delta_A \otimes C_B^1.$$

Proof: Let $(L, R) \in \ker \partial_0$. Then:

$$\begin{aligned}\partial_0(L,R) = 0 &\iff (\delta_A \otimes \mathbb{1})L + (\mathbb{1} \otimes \delta_B^T)R = 0,\\ &\iff \delta_A L + R\delta_B = 0. \end{aligned} \tag{20}$$

Eq. (20) yields:

$$\delta_A L = R\delta_B = V, \tag{21}$$

for some $V \in C_A^1 \otimes C_B^0$. Eq. (21) entails that all columns of V belong to $\operatorname{Im}\delta_A$ while its rows belong to $\operatorname{Im}\delta_B^T$. As a consequence, it must exists $U \in C_A^0 \otimes C_B^1$ such that:

$$V = \delta_A U \delta_B.$$

Therefore Eq. (21) can be re-written as:

$$\delta_A L = \delta_A U \delta_B$$

which yields:

$$\begin{aligned}(\delta_A \otimes \mathbb{1})(L + U\delta_B) &= \delta_A L + \delta_A U \delta_B\\ &= V + V\\ &= 0 \end{aligned} \tag{22}$$

Equivalently, Eq. (22) states that $L + U\delta_B$ has columns in $\ker \delta_A$:

$$L + U\delta_B \in \ker \delta_A \otimes C_B^0$$

and therefore:

$$L \in \ker \delta_A \otimes C_B^0 + C_A^0 \otimes \operatorname{Im}\delta_B^T,$$

as in the thesis. Similarly, we find

$$R \in C_A^1 \otimes \ker \delta_B^T + \operatorname{Im}\delta_A \otimes C_B^1.$$

□

A proof of Proposition 1, reported below for clarity, follows directly combining what said in Appendix B-A and Lemma 1.

Proposition 1 (Canonical Form): Let (L, R) be a Z-operator on the codespace of $\mathcal{C}(\delta_A, \delta_B)$. For a vector space $V \subseteq \mathbb{F}_2^n$, we denote by $V^\bullet$ any space such that $V \oplus V^\bullet \simeq \mathbb{F}_2^n$, (see Appendix A). Then, for the operator (L, R), the left part L can be expressed as a sum of a *free part* M_L and a *logical part* O_L such that every row of M_L belongs to $\operatorname{Im}\delta_B^T$ and every row of O_L belongs to $(\operatorname{Im}\delta_B^T)^\bullet$. Similarly, the right part R can be expressed as a sum of a free part M_R and a logical part O_R such that every column of M_R belongs to $\operatorname{Im}\delta_A$ and every column of O_R belongs to $(\operatorname{Im}\delta_A)^\bullet$. Hence, for (L, R) holds:

$$(L, R) = (M_L + O_L, M_R + O_R). \tag{CF}$$

We refer to the writing given by Eq. (CF) as *canonical form* of the operator (L, R).

In the main text, we have introduced the notions of row-column weight and logical row-column weight for a Z-operator on $\mathcal{C}(\delta_A, \delta_B)$. The definition of these two quantities finds its explanation in Proposition 2, whose proof builds on the results of Lemma 1.

Proposition 2: If (L, R) is a non-trivial logical Z-operator of $\mathcal{C}(\delta_A, \delta_B)$ then either $\#\mathrm{row}(L) \geq d_a$ or $\#\mathrm{col}(R) \geq d_b^T$ (or both).

Proof: If (L, R) is a non-trivial logical Z-operator, it must anti-commute with at least one logical X-operator (L_x, R_x). Because a Z-operator and a X-operator anti-commute if and only if their supports overlap on an odd number of positions, either L and L_x or R and R_x have odd overlap. Without loss of generality, we can assume that the former is verified and we can choose (L_x, R_x) as a left operator of the form

$$(L_x, R_x) = (f \otimes k, 0),$$

where f is a unit vector in $(\operatorname{Im}\delta_A^T)^\bullet$ and $k \in \ker \delta_B$. In other words, we choose logical X-operator $(f \otimes k, 0)$ from the set of generators of X-logical operators $\hat{\mathcal{L}}_x^{\mathrm{left}}$, as in the X-version of Eq. (19). The inner product equation for reshaped vectors Eq. (18) then yields:

$$\begin{aligned}1 = \big\langle (L_x, 0), (L,\, R)\big\rangle &= \langle L_x, L\rangle + \langle 0, R\rangle\\ &= f^T L k. \end{aligned} \tag{23}$$

Now, observe that $(L,\, R) \in \mathcal{C}_0$ is a non-trivial logical Z-operator of $\mathcal{C}(\delta_A, \delta_B)$ if and only if $[L, R] \in \mathcal{H}_0 = \ker \partial_0 / \operatorname{Im}\partial_{-1}$ and $[L, R] \neq 0$ or, equivalently, if and only if:

$$(L,\, R) \in \ker \partial_0 \setminus \operatorname{Im}\partial_{-1}.$$

In particular, (L, R) belongs to $\ker \partial_0$ and thanks to Lemma 1, we can re-write it as:

$$(L, R) = (K_A + U_L\delta_B, \bar{K}_B + \delta_A U_R),$$

where columns of K_A belong to $\ker \delta_A$ and rows of $\bar{K}_B$ belong to $\ker \delta_B^T$. Using Lemma 1's decomposition for $(L, R) \in \ker \partial_0$, we can expand the matrix-vector product Lk as:

$$\begin{aligned}Lk &= (K_A + U_L\delta_B)k\\ &= K_A k + U_L \delta_B k\\ &= K_A k \qquad\qquad \text{since } k \in \ker \delta_B. \end{aligned} \tag{24}$$

Eq. (24) entails $Lk = K_A k$ and therefore that Lk, being a linear combination of column-vectors in $\ker \delta_A$, belongs to $\ker \delta_A$ itself. Furthermore, by Eq. (23), $Lk \neq 0$. To sum up, Lk is a non-zero vector in $\ker \delta_A$ and therefore it must have Hamming weight at least d_a. As a consequence, L is a matrix with at least d_a rows:

$$\#\mathrm{row}(L) \geq d_a.$$

Similarly, we would have found:

$$\#\mathrm{col}(R) \geq d_b^T,$$

if we had assumed that (L, R) anti-commuted with a logical X-operator $(0, R_x)$ in $\hat{\mathcal{L}}_x^{\mathrm{right}}$. □

Corollary 1 follows easily.

Corollary 1: If (L, R) is a non-trivial logical Z-operator on $\mathcal{C}(\delta_A, \delta_B)$, at least one of the following hold:

(i) L has at least d_a rows which are not in $\operatorname{Im}\delta_B^T$ when seen as vectors in C_B^0.

(ii) R has at least d_b^T columns which are not in $\operatorname{Im}\delta_A$ when seen as vectors of C_A^1.

Proof: Write (L, R) in its canonical form:

$$(L, R) = (M_L + O_L, M_R + O_R),$$

and let

$$M_L = N_L \delta_B,$$
$$M_R = \delta_A N_R,$$

for some binary matrices N_L, N_R of size $n_a \times m_b$. As done in the proof of Proposition 2, consider a logical X-operator $(f \otimes k, 0)$ such that it anti-commutes with (L, R). Combining the canonical form of L and Eq. (24), yields:

$$\begin{aligned} Lk &= (O_L + N_L \delta_B)k \\ &= O_L k && \text{since } k \in \ker \delta_B \\ &= K_A k && \text{by Eq. (24)} \end{aligned}$$

for some $n_a \times n_b$ matrix K_A with columns in $\ker \delta_A$. By the same argument used in the proof of Proposition 2, we find:

$$|K_A k| \geq d_A \Rightarrow |O_L k| \geq d_A,$$

and in particular that O_L has at least d_a non-zero rows. Since by definition of canonical form the non-zero rows of O_L are precisely those rows of L which do not belong to $\operatorname{Im} \delta_B^T$, we have proven point (i). Point (ii) follows similarly in the case (L, R) anti-commutes with at least one logical X-operator of the form $(0, R_x)$. □

Corollary 1, together with Proposition 3 below, justifies the definition of the logical row-column weight for Z-operators on $\mathcal{C}(\delta_A, \delta_B)$ (Definition 2). The logical row-column weight of (L, R) is denoted by the symbol $\mathrm{wt}_{\mathrm{rc}}^{\log}(L, R)$ and stands for the integer pair $(\#\mathrm{row}_{\log}(L), \#\mathrm{col}_{\log}(R))$ where $\#\mathrm{row}_{\log}(L)$ is the number of rows of L that are not in $\operatorname{Im} \delta_B^T$ and $\#\mathrm{col}_{\log}(R)$ is the number of columns of R which are not in $\operatorname{Im} \delta_A$. Proposition 3, that we now prove, states that the logical row-column weight of a Z-operator on $\mathcal{C}(\delta_A, \delta_B)$ is an homology invariant of the chain complex $\mathfrak{C}_{A,B}$ and therefore it legitimates the name choice for this quantity.

Proposition 3: The logical row-column weight of a Z-operator on $\mathcal{C}(\delta_A, \delta_B)$ is an invariant of its homology class.

Proof: Let $[L, R]$ be the homology class of (L, R):

$$[L, R] = \big\{(L + G_L, R + G_R) : (G_L, G_R) \text{ is a } Z\text{-stabiliser}\big\}.$$

The operator $(G_L, G_R) \in \mathcal{C}_0$ is a Z-stabilizer for $\mathcal{C}(\delta_A, \delta_B)$ if and only if

$$\begin{aligned} (G_L, G_R) &= \partial_{-1}(U) \\ &= (U\delta_B, \delta_A U) \end{aligned} \tag{25}$$

For some $n_a \times m_b$ binary matrix U. Eq. (25) entails that any row of G_L belongs to $\operatorname{Im} \delta_B^T$ and any column of G_R belongs to $\operatorname{Im} \delta_A$. Therefore, if we write (L, R) in its canonical form:

$$(L, R) = (M_L + O_L, M_R + O_R),$$

we see that we can 'delete' all the rows of M_L by adding a stabiliser and hence 'move' part of the support of the operator (L, R) from the left qubits to the right qubits. Specifically, if $M_L = N_L \delta_B$ for some $n_a \times m_b$ binary matrix N_L, we consider the stabiliser $G = (N_L \delta_B, \delta_A N_L)$ and we obtain:

$$(L, R) + G = (O_L, M_R + O_R + \delta_A N_L).$$

Similarly, we could move the M_R part of the operator (L, R) from the right qubits to the left qubits, by adding the stabilizer $G' = (N_R \delta_B, \delta_A N_R)$, for a $n_a \times m_b$ matrix N_R such that $M_R = \delta_A N_R$.

On the other hand though, it is not possible to delete non-zero rows of O_L via stabiliser addition. In other words, it is not possible to remove, via stabiliser addition, any of the rows of L that are not in $\operatorname{Im} \delta_B^T$. Hence, the number $\#\mathrm{row}_{\log}(L)$ of non-zero rows of O_L is an homology invariant. Likewise, we find that it is not possible to delete any column in O_R by adding stabilisers and therefore $\#\mathrm{col}_{\log}(R)$ is a logical invariant too. □

The proof of Proposition 3 actually entails a stronger result than the invariance of the row-column weight of Z-operators on $\mathcal{C}(\delta_A, \delta_B)$. Namely, we have proven that the indices of the rows and the columns in the sets $\mathrm{row}_{\log}$ and $\mathrm{col}_{\log}$ respectively, are homology invariants of the reshaped Z-operators (L, R) on $\mathcal{C}(\delta_A, \delta_B)$. However, because to prove the correctness of ReShape it is sufficient to look at the cardinality of the two sets $\mathrm{row}_{\log}$ and $\mathrm{col}_{\log}$, we decided to state Proposition 3 in this more compact and elegant form.

We can now prove Proposition 4.

Proposition 4: Let S be a X-syndrome matrix for $\mathcal{C}(\delta_A, \delta_B)$ and (L, R) any valid solution to the Syndrome Equation

$$\sigma(L,\, R) = S. \tag{SE}$$

Suppose that the minimum weight operator $(L_{\min},\, R_{\min})$ with syndrome S has $(d_a/2, d_b^T/2)$-bounded logical row-column weight i.e.

$$\mathrm{wt}_{\mathrm{rc}}^{\log}(L_{\min},\, R_{\min}) = (\#\mathrm{row}_{\log}(L_{\min}), \#\mathrm{col}_{\log}(R_{\min})),$$

is such that

$$\#\mathrm{row}_{\log}(L_{\min}) < \frac{d_a}{2} \quad \text{and} \quad \#\mathrm{col}_{\log}(R_{\min}) < \frac{d_b^T}{2}. \tag{13}$$

Then, on input $\mathscr{D}_{\delta_A}$, $\mathscr{D}_{\delta_B^T}$, S and $(L,\, R)$, ReShape outputs a correct solution $(\tilde{L},\, \tilde{R})$ of (SE), provided that the classical decoders $\mathscr{D}_{\delta_A}$, $\mathscr{D}_{\delta_B^T}$ succeed. In other words, the solution $(\tilde{L}, \tilde{R})$ found by ReShape is in the same homology class as the minimum weight operator with syndrome S:

$$[L_{\min},\, R_{\min}] = [\tilde{L},\, \tilde{R}].$$

Proof: This is a proof by contradiction: we suppose that the minimum weight solution and the solution found by ReShape (Algorithm 1) are not homologically equivalent and we find as a consequence that the minimum weight solution need to have high logical row-column weight.

Let (L, R) be the valid solution of (SE) in input to ReShape and $(\tilde{L},\, \tilde{R})$ be the recovery operator found.

First note that $\sigma(\tilde{L}, \tilde{R}) = \sigma(L, R)$. In fact, the Split step only finds the canonical form of (L, R) and therefore changes neither the operator (L, R) nor its syndrome. The Decode step, possibly adds to (L, R) logical Z-operators (L_z, R_z) such

that $\sigma(L_z, R_z) = 0$ and therefore, even when it changes the operator, it preserves its syndrome.

Suppose now that the solution found by ReShape and the minimum weight solution $(L_{\min}, R_{\min})$ of (SE) belong to two different homology classes:

$$[\tilde{L},\, \tilde{R}] \neq [L_{\min},\, R_{\min}],$$

where:

$$\#\mathrm{row}_{\log}(L) < \frac{d_a}{2} \quad \text{and} \quad \#\mathrm{col}_{\log}(R) < \frac{d_b^T}{2}.$$

Since both $(L_{\min}, R_{\min})$ and $(\tilde{L}, \tilde{R})$ are valid solution of (SE), they must differ for an operator with zero X-syndrome. Because $(L_{\min}, R_{\min})$ and $(\tilde{L}, \tilde{R})$ are not homologically equivalent, they must differ for a non-trivial Z-operator in the normaliser $\mathcal{N}(\mathcal{S})$ of the stabiliser group. As such, they must differ for an operator which is the sum of a Z-stabiliser and a non-trivial logical operator:

$$(L_{\min}, R_{\min}) = (\tilde{L}, \tilde{R}) + (G_L, G_R) + (L_z, R_z), \tag{26}$$

where (G_L, G_R) is a Z-stabiliser and (L_z, R_z) is a non-trivial logical Z-operator.

Without loss of generality we assume that (L_z, R_z) is non-trivial on the left qubits, meaning that L_z has at least one non-zero column in $\ker \delta_A$. The proof is substantially the same in case it is non-trivial on the right qubits.

First, write the left operators $L_{\min}$ and $\tilde{L}$ in their canonical form with respect to the same unit-vector basis used to write the logical operators in $\mathcal{L}_z$ (see Eq. (16)):

$$\begin{aligned} L_{\min} &= M_{\min} + O_{\min}, \\ \tilde{L} &= \tilde{M} + \tilde{O}. \end{aligned}$$

Note that, by construction, the left operator $L_z + G_L$ is already in its canonical form, where L_z is its logical part and G_L is its free part. By Eq. (17), the sum is direct and therefore the equality given by Eq. (26) must hold component-wise for the free part and the logical part:

$$\begin{aligned} M_{\min} &= \tilde{M} + G_L \\ O_{\min} &= \tilde{O} + L_z. \end{aligned} \tag{27}$$

Let now focus on the logical part equality expressed by Eq. (27) and let L_z^j be a non-zero column of L_z in $\ker \delta_A$. Then:

$$O_{\min}^j = \tilde{O}^j + L_z^j, \quad L_z^j \in \ker \delta_A. \tag{28}$$

Eq. (12) for the classical decoder $\mathscr{D}_{\delta_A}$, entails:

$$|\tilde{O}^j| = \min_{k \in \ker \delta_A} |v + k|$$

for some input vector v defined by L. In particular, no vector $k' \in \ker \delta_A$ can overlap with $\tilde{O}^j$ in more than $d_A/2$ positions, otherwise we would have $|v + k'| < |\tilde{O}^j|$, against the assumption that $|\tilde{O}^j|$ is minimum. Thanks to this observation and considering the Hamming weight of the terms in Eq. (28), we obtain:

$$\begin{aligned} |O_{\min}^j| &= |\tilde{O}^j + L_z^j| \\ &= |\tilde{O}^j| + |L_z^j| - 2|\tilde{O}^j \wedge L_z^j| \\ &\geq |L_z^j| - |\tilde{O}^j \wedge L_z^j| \\ &\geq d_a - \frac{d_a}{2} = \frac{d_a}{2}, \qquad \text{by Eq. (28).} \end{aligned}$$

Because the weight of any of the columns of a matrix is a lower bound on the number of its non-zero rows, we have:

$$\#\mathrm{row}(O_{\min}) \geq \frac{d_a}{2}.$$

By definition of canonical form, this is equivalent to:

$$\#\mathrm{row}_{\log}(L_{\min}) \geq \frac{d_a}{2},$$

against the assumption.

We stress that the number $\#\mathrm{row}(O_{\min})$ of rows of $L_{\min}$ which do not belong to $\mathrm{Im}\, \delta_B^T$, does not depend on the particular splitting chosen for the canonical form. In fact, as stated in Proposition 3, the logical row weight of the left part of a Z-operator is an homology invariant. An argument similar to the one just outlined for the left part of $(L_{\min}, R_{\min})$ holds for its right part and yields:

$$\#\mathrm{col}_{\log}(R_{\min}) \geq d_b^T/2,$$

again contradicting the assumption. In conclusion, we have reached a contradiction and therefore it must be:

$$[L_{\min}, R_{\min}] = [\tilde{L}, \tilde{R}].$$

□

ACKNOWLEDGMENT

The authors would like to thank Christophe Vuillot for helpful discussions and for carefully reading a draft of this work. Armanda O. Quintavalle would like to thank Joschka Roffe for providing the matrices used in the simulations. ETC's contributions were made while he was at The University of Sheffield.

REFERENCES

[1] J.-P. Tillich and G. Zemor, "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 1193–1202, Feb. 2014.

[2] A. A. Kovalev and L. P. Pryadko, "Fault tolerance of quantum low-density parity check codes with sublinear distance scaling," *Phys. Rev. A, Gen. Phys.*, vol. 87, Feb. 2013, Art. no. 020304.

[3] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, "Topological quantum memory," *J. Math. Phys.*, vol. 43, no. 9, pp. 4452–4505, 2002.

[4] A. G. Fowler, A. M. Stephens, and P. Groszkowski, "High-threshold universal quantum computation on the surface code," *Phys. Rev. A, Gen. Phys.*, vol. 80, Nov. 2009, Art. no. 052312.

[5] N. Delfosse and N. H. Nickerson, "Almost-linear time decoding algorithm for topological codes," 2017, *arXiv:1709.06218*.

[6] A. Leverrier, J.-P. Tillich, and G. Zémor, "Quantum expander codes," in *Proc. IEEE 56th Annu. Symp. Found. Comput. Sci.*, Oct. 2015, pp. 810–824.

[7] M. Sipser and D. A. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.

[8] O. Fawzi, A. Grospellier, and A. Leverrier, "Efficient decoding of random errors for quantum expander codes," in *Proc. 50th Annual ACM SIGACT Symp. Theory Comput.*, 2018, pp. 521–534.

[9] O. Fawzi, A. Grospellier, and A. Leverrier, "Constant overhead quantum fault-tolerance with quantum expander codes," in *Proc. IEEE 59th Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 2018, pp. 743–754.
[10] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape," *Phys. Rev. Res.*, vol. 2, no. 4, Dec. 2020, Art. no. 043423.
[11] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Inf. Comput.*, vol. 8, no. 10, pp. 987–1000, Nov. 2008.
[12] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, "Fifteen years of quantum LDPC coding and improved decoding strategies," *IEEE Access*, vol. 3, pp. 2492–2519, 2015.
[13] Y.-H. Liu and D. Poulin, "Neural belief-propagation decoders for quantum error-correcting codes," *Phys. Rev. Lett.*, vol. 122, no. 20, May 2019, Art. no. 200501.
[14] A. Rigby, J. C. Olivier, and P. Jarvis, "Modified belief propagation decoders for quantum low-density parity-check codes," *Phys. Rev. A, Gen. Phys.*, vol. 100, Jul. 2019, Art. no. 012330.
[15] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," 2019, *arXiv:1904.02703*.
[16] M. Li and T. J. Yoder, "A numerical study of bravyi-bacon-shor and subsystem hypergraph product codes," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Oct. 2020, pp. 109–119.
[17] A. Grospellier, L. Grouès, A. Krishna, and A. Leverrier, "Combining hard and soft decoders for hypergraph product codes," 2020, *arXiv:2004.11199*.
[18] N. Delfosse, V. Londe, and M. Beverland, "Toward a union-find decoder for quantum LDPC codes," 2021, *arXiv:2103.08049*.
[19] H. Bombín, "Single-shot fault-tolerant quantum error correction," *Phys. Rev. X*, vol. 5, no. 3, 2015, Art. no. 031043.
[20] A. O. Quintavalle, M. Vasmer, J. Roffe, and E. T. Campbell, "Single-shot error correction of three-dimensional homological product codes," 2020, *arXiv:2009.11790*.
[21] W. C. Huffman and V. Pless, *Fundamentals Error-Correcting Codes*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
[22] D. Gottesman, "Stabilizer codes and quantum error correction," 1997, *arXivph/9705052*.
[23] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Phys. Rev. A, Gen. Phys.*, vol. 54, no. 2, p. 1098, 1996.
[24] A. Steane, "Multiple-particle interference and quantum error correction," *Proc. R. Soc. Lond. A, Math., Phys. Eng. Sci.*, vol. 452, pp. 2551–2577, Nov. 1996.
[25] J. Roffe, "Quantum error correction: An introductory guide," *Contemp. Phys.*, vol. 60, no. 3, pp. 226–245, Jul. 2019.
[26] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[27] J.-P. Tillich and G. Zémor, "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength," *IEEE Trans. Inf. Theory*, vol. 60, no. 2, pp. 1193–1202, Feb. 2014.
[28] S. Bravyi and M. B. Hastings, "Homological product codes," in *Proc. 46th Annu. ACM Symp. Theory Comput.*, May 2014, pp. 273–282.
[29] B. Audoux and A. Couvreur, "On tensor products of CSS codes," 2015, *arXiv:1512.07081*.
[30] W. Zeng and L. P. Pryadko, "Higher-dimensional quantum hypergraph-product codes with finite rates," *Phys. Rev. Lett.*, vol. 122, no. 23, Jun. 2019, Art. no. 230501.
[31] A. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, vol. 303, no. 1, pp. 2–30, 2003.
[32] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," 1998, *arXivph/9811052*.
[33] E. Dennis, "Toward fault-tolerant quantum computation without concatenation," *Phys. Rev. A, Gen. Phys.*, vol. 63, no. 5, Apr. 2001, Art. no. 052314.
[34] N. Delfosse and M. B. Hastings, "Union-find decoders for homological product codes," 2020, *arXiv:2009.14226*.
[35] K. M. Svore, A. W. Cross, I. L. Chuang, and A. V. Aho, "A flow-map model for analyzing pseudothresholds in fault-tolerant quantum computing," 2005, *arXivph/0508176*.
[36] S. Bravyi and J. Haah, "Analytic and numerical demonstration of quantum self-correction in the 3D cubic code," 2011, *arXiv:1112.3252*.
[37] S. Bravyi and J. Haah, "Quantum self-correction in the 3D cubic code model," *Phys. Rev. Lett.*, vol. 111, Nov. 2013, Art. no. 200501.
[38] S. Evra, T. Kaufman, and G. Zémor, "Decodable quantum LDPC codes beyond the $\sqrt{n}$ distance barrier using high dimensional expanders," 2020, *arXiv:2004.07935*.
[39] N. P. Breuckmann and J. N. Eberhardt, "Balanced product quantum codes," 2020, *arXiv:2012.09271*.
[40] S. Lang, *Undergraduate Algebra*. Cham, Switzerland: Springer, 2005.
[41] I. N. Herstein, *Abstract Algebra*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.
[42] H. Bombin and M. A. Martin-Delgado, "Homological error correction: Classical and quantum codes," *J. Math. Phys.*, vol. 48, no. 5, 2007, Art. no. 052105.
[43] S. Bravyi and M. B. Hastings, "Homological product codes," in *Proc. 46th Annu. ACM Symp. Theory Comput.*, May 2014, pp. 273–282.
[44] A. Hatcher, *Algebraic Topology*, vol. 606, no. 9. Cambridge, U.K.: Cambridge Univ. Press, 2002.
[45] E. Campbell, "A theory of single-shot error correction for adversarial noise," *Quantum Sci. Technol.*, vol. 4, no. 2, 2019, Art. no. 025006.

Armanda O. Quintavalle received the B.Sc. degree in mathematics from the University of Pisa, Pisa, Italy, in 2015, and the M.Sc. degree in mathematics from the University of Trento, Trento, Italy, in 2018. She is currently pursuing the degree with The University of Sheffield. Her research interests include quantum error correction and quantum computation.

Earl T. Campbell received the Ph.D. degree in quantum computing from the University of Oxford in 2008. He was a Royal Commission of 1861 Fellowship at University College London from 2008 to 2010 and a Post-Doctoral Research at the University of Potsdam and Free Universität Berlin from 2010 to 2014. He was a Senior Research Scientist at AWS from 2020 to 2021. He is currently the Head of Architecture at Riverlane and a Senior Lecturer at The University of Sheffield. His research interests include quantum computing, quantum error correction, quantum algorithms, and resource theories.

Chapter 4

Logical gates - Qubit partitions

Context and Results

In [1] we propose a method to implement logical encoded gates on hypergraph product codes via a generalization of the concept of transversal gates [2] that relies on partitions of the code's data qubits. By the Eastin-Knill Theorem, no stabilizer code has a universal set of purely transversal gates [3] and therefore a fault-tolerant universal set will necessarily come with some, maybe substantial, overhead [4, 5, 6]. Bravyi and Konig showed that locality constraints in two dimensions limit the set of fault-tolerant gates of a code to the Clifford group [7]. The planar code obeys Bravyi-Konig constraints and, even if more general hypergraph product codes do not, it has been shown [8, 9, 10] that they still suffer from the same limitations. Krishna and Poulin illustrated how, in principle, the idea of braiding on the planar code can be generalized to hypergraph product codes but their result is not constructive.

We take a generalization approach too and adapt Moussa's folding [11, 12, 13] of the planar code to other hypergraph product code families. Firstly, we show that a clever choice of the qubits partition naturally yield fault-tolerant operations. Secondly, we exhibit a few practical fault-tolerant examples to implement Clifford gates on families of hypergraph product codes that obey some symmetry constraints. Key in our construction is the discovery of a standard basis for the logical space of all hypergraph product codes that we believe is of independent interest. Crucially, transversal gates and the generalization we consider, do not require any qubit overhead. Albeit limited, our work gives a constructive approach to building logical gates on hypergraph product codes.

Limitations

The qubits partition method we proposed falls short in two ways. First, it does not yield the full Clifford group at zero overhead. State injection [4] and pieceable fault tolerance [14] can promote our set to computationally universal, however at the price of substantial physical qubit and time cost. As in [13], we could explore what happens if we relegate some of the logical qubits to the role of gauge qubits. Second, we conjecture that our partition strategy could produce non-Clifford gates on higher dimensional hypergraph product codes [15, 16], but we have not found the right partition – yet.

Authorship declaration

AOQ derived the proofs and wrote the majority of the manuscript.

References

[1] Armanda O Quintavalle, Paul Webster, and Michael Vasmer. "Partitioning qubits in hypergraph product codes to implement logical gates". In: *arXiv preprint arXiv:2204.10812* (2022).

[2] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[3] Bryan Eastin and Emanuel Knill. "Restrictions on transversal encoded quantum gate sets". In: *Physical review letters* 102.11 (2009), p. 110502.

[4] Daniel Gottesman. "Fault-tolerant quantum computation with constant overhead". In: *arXiv preprint arXiv:1310.2984* (2013).

[5] Tomas Jochym-O'Connor. "Fault-tolerant gates via homological product codes". In: *Quantum* 3 (2019), p. 120.

[6] Lawrence Z Cohen et al. "Low-overhead fault-tolerant quantum computing using long-range connectivity". In: *Science Advances* 8.20 (2022).

[7] Sergey Bravyi and Robert König. "Classification of topologically protected gates for local stabilizer codes". In: *Physical review letters* 110.17 (2013), p. 170503.

[8] Tomas Jochym-O'Connor, Aleksander Kubica, and Theodore J Yoder. "Disjointness of stabilizer codes and limitations on fault-tolerant logical gates". In: *Physical Review X* 8.2 (2018), p. 021047.

[9] Simon Burton and Dan Browne. "Limitations on transversal gates for hypergraph product codes". In: *IEEE Transactions on Information Theory* 68.3 (2021), pp. 1772–1781.

[10] Nouédyn Baspin and Anirudh Krishna. "Connectivity constrains quantum codes". In: *Quantum* 6 (2022), p. 711.

[11] Aleksander Kubica, Beni Yoshida, and Fernando Pastawski. "Unfolding the color code". In: *New Journal of Physics* 17.8 (2015), p. 083026.

[12] Jonathan E Moussa. "Transversal Clifford gates on folded surface codes". In: *Physical Review A* 94.4 (2016), p. 042316.

[13] Nikolas P Breuckmann and Simon Burton. "Fold-Transversal Clifford Gates for Quantum Codes". In: *arXiv preprint arXiv:2202.06647* (2022).

[14] Theodore J Yoder, Ryuji Takagi, and Isaac L Chuang. "Universal fault-tolerant gates on concatenated stabilizer codes". In: *Physical Review X* 6.3 (2016), p. 031039.

[15] Michael Vasmer and Dan E Browne. "Three-dimensional surface codes: Transversal gates and fault-tolerant architectures". In: *Physical Review A* 100.1 (2019), p. 012312.

[16] Tomas Jochym-O'Connor and Theodore J Yoder. "Four-dimensional toric code with non-Clifford transversal gates". In: *Physical Review Research* 3.1 (2021), p. 013118.

Partitioning qubits in hypergraph product codes to implement logical gates

Armanda O. Quintavalle[1], Paul Webster[2], and Michael Vasmer[3,4,]

[1]Department of Physics & Astronomy, University of Sheffield, Sheffield, S3 7RH, United Kingdom
[2]Centre for Engineered Quantum Systems, School of Physics, The University of Sydney, Sydney, NSW 2006, Australia
[3]Perimeter Institute for Theoretical Physics, Waterloo, ON N2L 2Y5, Canada
[4]Institute for Quantum Computing, University of Waterloo, Waterloo, ON N2L 3G1, Canada

The promise of high-rate low-density parity check (LDPC) codes to substantially reduce the overhead of fault-tolerant quantum computation depends on constructing efficient, fault-tolerant implementations of logical gates on such codes. Transversal gates are the simplest type of fault-tolerant gate, but the potential of transversal gates on LDPC codes has hitherto been largely neglected. We investigate the transversal gates that can be implemented in hypergraph product codes, a class of LDPC codes. Our analysis is aided by the construction of a symplectic canonical basis for the logical operators of hypergraph product codes, a result that may be of independent interest. We show that in these codes transversal gates can implement Hadamard (up to logical SWAP gates) and control-Z on all logical qubits. Moreover, we show that sequences of transversal operations, interleaved with error-correction, allow implementation of entangling gates between arbitrary pairs of logical qubits in the same code block. We thereby demonstrate that transversal gates can be used as the basis for universal quantum computing on LDPC codes, when supplemented with state injection.

1 Introduction

In recent years, quantum computing has transitioned from a theoretical idea to a real technology that is competitive with state-of-the-art classical computing on carefully selected tasks [1]. However, realising its potential to solve intractable problems of practical importance requires the development of a fault-tolerant quantum computer—a device that can perform long quantum computations to a very high degree of accuracy even in the presence of noise [2]. Fault-tolerant quantum computing can be achieved by encoding quantum information into quantum error-correcting codes, but only at the cost of substantial overhead. For standard fault-tolerant quantum architectures based on the surface code this overhead is prohibitively large for the foreseeable future, with approximately 10^6 to 10^8 qubits being required to realise useful applications [3, 4].

High-rate quantum LDPC codes offer a promising avenue to significantly reducing this overhead [5]. These codes move beyond topological codes such as the surface code [6] by substituting the requirement that stabiliser check operators are geometrically local with a weaker condition that they must be sparse. They thereby preserve the essential benefits of the surface code for error-correction, while allowing much smaller qubit overheads [5, 7].

In this work, we focus on hypergraph product codes—a class of high-rate LDPC codes derived from the tensor product of classical LDPC codes. Even when restricted to this highly-structured class of codes, it is not yet clear what is the best strategy to reliably perform logic. The earliest proposal for fault-tolerant logic on LDPC codes relies on state injection protocols that, while achieving a constant overhead per logical qubit in the asymptotic regime, would have substantial finite-size overheads [8]. In [9, 10], code deformation on hypergraph product codes is used to perform Clifford gates. The protocol proposed preserves the LDPC property of the code throughout the entire computation; however, even if in principle the whole Clifford group is implementable via this framework, there is no promise that this is the case for an arbitrary hypergraph product

code nor is there a promise on the time cost of each gate implementation. Another variation of code deformation, via non-destructive measurement of high-weight logical operators, is proposed in [11]; the method there proposed enables the implementation of the full Clifford group on all hypergraph product codes. Nevertheless, this method requires many ancilla qubits if we want to operate in parallel on all of the encoded qubits.

We take a different approach to fault-tolerance and explore transversal gates [12, 13, 14, 15, 16, 17]. Transversal gates on homological codes [18], close cousins of the hypergraph product codes, were studied in [19]. Universality is there obtained by combining two quantum codes with complementary transversal gates. Hence in [19] the problem of universality is deferred to the one of finding complementary classes of good quantum codes that are then combined via the homological product; the information is always protected by at least one code whilst leveraging the transversal gates of the other. We relax the constraints on the underlying (classical) codes used as seed codes in the hypergraph product and investigate how the core symmetries that arise from the product structure itself enable transversal gates.

The transversal gates we find are only a small subset of the Clifford group. Nonetheless, we further the knowledge on hypergraph product codes by proving that it is possible to efficiently construct a canonical basis for their logical space. The existence of such a canonical basis is non-trivial and may be of independent interest in the development of other computational frameworks on hypergraph product codes or related families of codes (e.g. higher dimensional homological product codes [18, 20, 21], or the codes introduced in [22]). Furthermore, we illustrate how pieceably fault-tolerant circuits [23] and state injection can be used on symmetric hypergraph product codes in conjunction with the transversal gates proposed to give a universal gate set. Importantly, our scheme is relevant to small, low-overhead quantum error-correcting codes with practical near-term potential.

After a short introduction to hypergraph product codes in Section 2, we detail the existence of the canonical basis for their logical space in Section 2.1. In Section 3, we illustrate how the idea of unfolding the color code into surface codes relies on more general symmetries of the product structure, symmetries that are inherited by square and symmetric codes. Thanks to this observation and the use of our canonical basis, we showcase the transversal implementation of some Clifford gates on hypergraph product codes. We comment on how pieceable fault tolerance and state injection, in combination with the transversal gates introduced, could be used on 'small' codes to perform arbitrary computations in Section 4.

2 Hypergraph Product Codes

Any binary matrix H in $\mathbb{F}_2^{m\times n}$ defines a classical code whose codewords are vectors in the kernel of H, $\ker H$. The classical code so defined uses n bits to protect from errors $k = n - \text{rank}(H)$ bits and detects all errors of Hamming weight less than d, the minimum weight of a codeword. Briefly, we say that H defines an $[n, k, d]$ code [24]. Similarly, any pair of binary matrices H_x in $\mathbb{F}_2^{m_x\times n}$ and H_z in $\mathbb{F}_2^{m_z\times n}$ such that $H_x \cdot H_z^T = 0 \mod 2$ defines a stabiliser CSS code [25, 26, 27]. The stabiliser group is $\mathbf{S} = \langle \mathbf{S_x} \cup \mathbf{S_z} \rangle$, where $\mathbf{S_x}$ is the set of X operators whose support vector[1] is equal to a row of H_x, and $\mathbf{S}_z$ is the set of Z operators whose support vector is equal to a row of H_z. Since an X and a Z operator commute if and only if their supports have even overlap, the orthogonality of H_x and H_z in $\mathbb{F}_2$ ensures that the stabiliser group $\mathbf{S}$ is well defined. The quantum code with stabiliser group $\mathbf{S}$ uses n physical qubits to encode $k = n - \text{rank}(H_x) - \text{rank}(H_z)$ logical qubits and detects all errors of weight less than $d = \min(d_x, d_z)$, where d_x is the minimum weight of a vector in $\ker H_z$ that is not in the image of H_x^T, $\text{Im}\, H_x^T$, and d_z is the minimum weight of a vector in $\ker H_x$ that is not in $\text{Im}\, H_z^T$. Briefly, we say that the quantum code is $[\![n, k, d]\!]$.

A hypergraph product code [28, 18, 29] is a CSS code produced from the hypergraph product of two linear codes. Given H_a in $\mathbb{F}_2^{m_a\times n_a}$ and H_b in $\mathbb{F}_2^{m_b\times n_b}$ we define:

$$H_x = \begin{pmatrix} H_a \otimes I_{n_b} & I_{m_a} \otimes H_b^T \end{pmatrix}, \tag{1}$$

$$H_z = \begin{pmatrix} I_{n_a} \otimes H_b & H_a^T \otimes I_{m_b} \end{pmatrix}, \tag{2}$$

[1] Given a Pauli operator on n qubits $P = P_1 \otimes P_2 \otimes \cdots \otimes P_n \in \mathcal{P}_n$, its support vector is the unique vector $v \in \mathbb{F}_2^n$ with ith coordinate $v[i] = 1$ if and only if $P_i \neq I$.

where $\otimes$ is the tensor product. Since $H_x \cdot H_z^T = 2H_a \otimes H_b^T = 0 \mod 2$, this choice is valid and we indicate by $\mathrm{HGP}(H_a, H_b)$ this CSS code. If H_η (resp. H_η^T) define a $[n_\eta, k_\eta, d_\eta]$ (resp. $[m_\eta, k_\eta^T, d_\eta^T]$) classical code, then $\mathrm{HGP}(H_a, H_b)$ has parameters

$$[\![n_a n_b + m_a m_b, k_a k_b + k_b^T k_a^T, d]\!], \tag{3}$$

where $d = \min(d_a, d_a^T, d_b, d_b^T)$.

The physical qubits of $\mathrm{HGP}(H_a, H_b)$ can be arranged in two rectangular grids of sizes $n_a \times n_b$ and $m_a \times m_b$, see fig. 2 and [30]. As such, we enumerate the physical qubits of $\mathrm{HGP}(H_a, H_b)$ by the triplet (i, h, L) and (j, ℓ, R), where $1 \leq i \leq n_a$, $1 \leq h \leq n_b$, $1 \leq j \leq m_a$, $1 \leq \ell \leq m_b$. The first two coordinates of each triplet refer to the row and column positions of the physical qubit in the grid. The third coordinate L or R, short for left and right respectively, distinguishes the two grids and is referred to as *sector* of the physical qubit. Via this spacial mapping of physical qubits, each stabiliser generator will have support contained in a row of one sector and a column of the other. Specifically, elements of $\mathbf{S_x}$ can be indexed by a pair j, h, such that $S_x(j, h)$ has support on qubits on row j of the right sector and column h of the left sector. Elements of $\mathbf{S_z}$, indexed by a pair i, ℓ are similar, but have support on rows of qubits in the left sector and columns in the right sector.

We say that a hypergraph product code is a *square* code if it is derived from one classical matrix only. Square codes $\mathrm{HGP}(H, H)$ take their name from the shape of the two grids of physical qubits: if $H \in \mathbb{F}_2^{m \times n}$, the left and right sectors are square grids of sizes $n \times n$ and $m \times m$ respectively. A square code $\mathrm{HGP}(H, H)$ such that $H = H^T$ or, more loosely, such that $H \in \mathbb{F}_2^{n \times n}$ is square and $\mathrm{Im}(H) = \mathrm{Im}(H^T)$, is said *symmetric* and denoted as $\mathrm{HGP}_{\mathrm{sy}}(H)$.

2.1 Logical Pauli Operators

Logical X and Z operators of $\mathrm{HGP}(H_a, H_b)$ can be chosen to have support on a 'line' of qubits, meaning that each operator acts non-trivially either on the left or right sector, and either on a column or on a row of physical qubits [18, 30]. We build on this and show that such a 'line basis' can, in addition, be chosen to be symplectic. The existence of such canonical basis is the key ingredient in the identification of transversal gates on hypergraph product codes. Formally:

Theorem 1. *For any hypergraph product code there exist bases $\mathcal{B}^x_{\mathrm{line}}$ and $\mathcal{B}^z_{\mathrm{line}}$ of logical X and Z operators respectively such that:*

1. *Any operator in $\mathcal{B}^x_{\mathrm{line}}$ or $\mathcal{B}^z_{\mathrm{line}}$ has support on a 'line' of qubits; by line here we intend that the support of the operator is contained in either a column or a row of left or right sector qubits, when qubits are arranged on two grids as explained above.*
2. *For any operator in $\mathcal{B}^x_{\mathrm{line}}$ there exists one and only one operator in $\mathcal{B}^z_{\mathrm{line}}$ that anticommutes with it. More precisely for every pairs of operators, one in $\mathcal{B}^x_{\mathrm{line}}$ and one in $\mathcal{B}^z_{\mathrm{line}}$, either their support overlaps on exactly one qubit or does not overlap at all.*

We refer to any such pair of bases as canonical basis for $HGP(H_a, H_b)$.

The proof of Theorem 1 is constructive and relies on a modified version of the Gaussian elimination algorithm over $\mathbb{F}_2$ (Algorithm 1), which yields a special kind of triangular matrices that we call strongly lower triangular matrices.

Definition 1. *An $m \times n$ matrix A is said to be strongly lower triangular if:*

1. *Any column j has a pivot p such that all the elements below the pivot are zero.*
2. *All the pivots are distinct.*
3. *Reordering if necessary, for any pivot $A_{p,j} = 1$, the coefficients to its right are zero i.e. $A_{p,j+1} = A_{p,j+2} = \ldots A_{p,m} = 0$.*

We indicate by $\pi(A)$ the set of row pivots of A:

$$\pi(A) = \{p \text{ s.t. } A_{p,j} \text{ is a pivot for some column index } j \text{ of } A\}$$

Given a vector space, we say that a set of vectors is strongly lower triangular if its matrix representation is.

An example of a strongly lower triangular matrix is:

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4}$$

where $\pi(A) = \{3, 5, 6, 7\}$. Observe in particular the pivot $A_{3,1}$: all the coefficients to its right are zero even if the submatrix on its right is not zero.

As Lemma 1 below shows, strongly lower triangular bases for the classical codes and their transposes used as seed codes in the hypergraph product naturally give a canonical basis for the associated hypergraph product code.

Lemma 1. *Given strongly lower triangular bases for* $\ker H_a$, $\ker H_a^T$, $\ker H_b$ *and* $\ker H_b^T$, *is it possible to construct a canonical basis for the associate hypergraph product code* $HGP(H_a, H_b)$.

Proof. We begin with some notation. Given a vector space $V \subseteq \mathbb{F}_2^n$ its complement, $V^\bullet \subseteq \mathbb{F}_2^n$, is the vector space such that $V \oplus V^\bullet = \mathbb{F}_2^n$. Crucially, $V^\bullet$ has same dimension as the orthogonal complement $V^\perp$ of V:

$$V^\perp = \{w \in \mathbb{F}_2^n \text{ s.t. } \langle v, w\rangle = 0 \text{ for all } v \in V\},$$

but they are, in general, different spaces. For example, if $V = \text{Span}\left((1,1,1)^T, (0,1,0)^T\right)$ then $V^\perp = \text{Span}\left((1,0,1)^T\right)$ and $V^\bullet = \text{Span}\left((0,0,1)^T\right)$.

Before detailing the proof of Lemma 1, we remind the reader that the logical Z operators of $\text{HGP}(H_a, H_b)$ are spanned by:

$$\left\{(k \otimes \bar{f}, 0) \text{ for } k \in \ker H_a, \bar{f} \in (\text{Im}\, H_b^T)^\bullet\right\} \cup \left\{(0, g \otimes \bar{h}) \text{ for } \bar{h} \in \ker H_b^T, g \in (\text{Im}\, H_a)^\bullet\right\},$$

and the logical X operators are spanned by:

$$\left\{(\bar{f}' \otimes k', 0) \text{ for } k' \in \ker H_b, \bar{f}' \in (\text{Im}\, H_a^T)^\bullet\right\} \cup \left\{(0, \bar{h}' \otimes g') \text{ for } \bar{h}' \in \ker H_a^T, g' \in (\text{Im}\, H_b)^\bullet\right\}.$$

See, for instance, [18, 30]. Thus, the problem of finding a canonical basis for $\text{HGP}(H_a, H_b)$ can be reduced to the problem of finding suitable bases for the kernels and the complement spaces defined by the classical seed matrices. In the proof below we follow this approach and show that strongly lower triangular bases for the classical seed matrices indeed induce a canonical basis for $\text{HGP}(H_a, H_b)$.

Let $A = \{a_i\} \in \mathbb{F}_2^{n_a}$, $\bar{A} = \{\alpha_j\} \in \mathbb{F}_2^{m_a}$, $B = \{b_h\} \in \mathbb{F}_2^{n_b}$ and $\bar{B} = \{\beta_\ell\} \in \mathbb{F}_2^{m_a}$ be strongly lower triangular bases for $\ker H_a$, $\ker H_a^T$, $\ker H_b$ and $\ker H_b^T$ respectively. The indices are the pivots: a_i has ith coordinate $a_i[i] = 1$ and for any $i' > i$, its i'th coordinate $a_i[i'] = 0$.

We claim that if $f_i \in \mathbb{F}_2^{n_a}$ is the ith unit vector, then the set $A_p = \{f_i\}_{i \in \pi(A)}$ is a basis of the complement $(\text{Im}\, H_a^T)^\bullet$ of $\text{Im}\, H_a^T$. Since the size $n_a - \text{rank}(H_a)$ of A_p equals the dimension of $(\text{Im}\, H_a^T)^\bullet$ and the vectors in A_p are clearly linearly independent, we only need to prove that they generate a space that has trivial intersection with $\text{Im}\, H_a$ i.e. $\text{Span}\,(A_p) \cap \text{Im}\, H_a = \{0\}$. To see this, suppose by contradiction that $f_i \in \text{Im}\, H_a^T$ for some $f_i \in A_p$. Then, f_i can be written as a linear combination of a set ρ of rows of H_a:

$$\sum_{v^T \in \rho} v = f_i. \tag{5}$$

By construction, for any $i, i' \in \pi(A)$, $\langle f_i, a_{i'}\rangle = 1$ if and only if $i' = i$ and 0 otherwise. Combining

this with eq. (5), yields:

$$\begin{aligned} 1 &= \langle f_i, a_i, \rangle \\ &= \left\langle \sum_{v^T \in \rho} v,\, a_i \right\rangle && \text{by hypothesis,} \\ &= \sum_{v^T \in \rho} \langle v, a_i \rangle && \text{by linearity,} \\ &= 0 && a_i \in \ker H_a, \end{aligned}$$

which is a contradiction. Hence, $A_p \cap \operatorname{Im} H_a^T = \emptyset$ and therefore $A_p \subseteq (\operatorname{Im} H_a^T)^\bullet$. Using this same argument and the linearity of the inner product, we have that no linear combination of vectors in A_p belongs to $\operatorname{Im} H_a^T$. Hence, only the zero vector belongs to both the span of A_p and $\operatorname{Im} H_a$. This completes the proof that A_p is a basis of $(\operatorname{Im} H_a^T)^\bullet$.

Similarly, we can prove that $\bar{A}_p$, B_p and $\bar{B}_p$ are well defined bases of the corresponding complement spaces.

Because the tensor product of bases is a basis of the tensor product space, the sets:

$$\mathcal{B}^x_{\text{line}} = \{(f_i \otimes b_h, 0), (0, \alpha_j \otimes f_\ell)\}, \tag{6}$$
$$\mathcal{B}^z_{\text{line}} = \{(a_i \otimes f_h, 0), (0, f_j \otimes \beta_\ell)\}, \tag{7}$$

for $i \in \pi(A)$, $h \in \pi(B)$, $j \in \pi(\bar{A})$, and $\ell \in \pi(\bar{B})$, are sets of linearly independent logical operators. Moreover, it is straightforward to see that operators in $\mathcal{B}^x_{\text{line}}$ and $\mathcal{B}^z_{\text{line}}$ have support on a line of qubits. Thus, in order to verify that $\mathcal{B}_{\text{line}} = \mathcal{B}^x_{\text{line}} \cup \mathcal{B}^z_{\text{line}}$ is a canonical basis for $\text{HGP}(H_a, H_b)$, we only need to show that it is symplectic.

Clearly, left and right operators do not overlap. If instead we take a left Z operator and a left X operator we find:

$$\langle (a_i \otimes f_h, 0),\, (f_{i'} \otimes b_{h'}, 0) \rangle = a_i[i'] \cdot b_{h'}[h], \tag{8}$$

and by strong lower triangularity:

$$a_i[i'] \cdot b_{h'}[h] = \begin{cases} 1, & \text{if } i = i' \text{ and } h = h', \\ 0, & \text{otherwise .} \end{cases}$$

Since an equivalent relation holds for pairs of right operators, we have shown that for every operator in $\mathcal{B}^x_{\text{line}}$ there exists a unique operator in $\mathcal{B}^z_{\text{line}}$ that is not orthogonal to it. Furthermore, two overlapping operators of different type overlap on exactly one physical qubit e.g. for the two left operators in eq. (8), it is the physical qubit at position (i, h, L) when $i = i'$ and $h = h'$.

In conclusion, $\mathcal{B}_{\text{line}} = \mathcal{B}^x_{\text{line}} \cup \mathcal{B}^z_{\text{line}}$ is a canonical basis for $\text{HGP}(H_a, H_b)$ as per Theorem 1. □

Theorem 1 is a corollary of Lemma 1, provided that strongly lower triangular bases exist and can be found. We show how this is in fact the case in Appendix A, where we present a modification of the Gaussian reduction algorithm, Algorithm 1, which finds a strongly triangular basis for any $n \times n$ binary matrix in time $O(n^3)$.

Practically, if an arbitrary code has a canonical basis as per Theorem 1, any indexing of the physical qubits of the code naturally yields an indexing of the logical qubits. Namely, if the unique physical qubit in the overlap of a basis logical X operator and a basis logical Z operator has index ι, then the index of the corresponding logical qubit is ι too, and we indicate it as q_ι. We call the physical qubit ι the physical *pivot* of the logical qubit q_ι.

For a hypergraph product code $\text{HGP}(H_a, H_b)$, if we display qubits on a grid as explained above and we fix a canonical basis as per Theorem 1, we can uniquely index logical qubits as $q^L_{i,h}$ and $q^R_{j,\ell}$ corresponding to the pivots (i, h, L) and (j, ℓ, R). By construction, the sector L or R indicates whether the logical qubit has canonical operators supported on the left or right sector qubits and the first two coordinates specify where the two canonical operators cross. We refer to physical qubits (i, i, σ), $\sigma = L, R$, as *diagonal* qubits and to the others as *mirror* qubits. The logical qubits inherit the same attribute of their pivots. To sum up, we have found a classification of the logical qubits of $\text{HGP}(H_a, H_b)$ that depends on the physical position of the crossing of the logical X and Z operators, $\bar{X}$ and $\bar{Z}$, see table 1.

symbol	$\bar{Z}$	$\bar{X}$	pivot qubit
$q^L_{i,h}$	$(a_i \otimes f_h, 0)$	$(f_i \otimes b_h, 0)$	(i, h, L)
$q^R_{j,\ell}$	$(0, f_j \otimes \beta_\ell)$	$(0, \alpha_j \otimes f_\ell)$	(j, ℓ, R)

Table 1: Classification and indexing of the logical qubits of a hypergraph product code HGP(H_a, H_b) where $\{a_i\}, \{b_h\}, \{\alpha_j\}, \{\beta_\ell\}$ are strongly lower triangular bases of $\ker H_a, \ker H_b, \ker H_a^T, \ker H_b^T$ indexed over the basis' pivots. We refer to logical qubits whose pivot lies on the diagonal e.g. $q^L_{i,i}$ or $q^R_{j,j}$ as diagonal logical qubits, and to the others as mirror logical qubits.

3 Transversal Clifford gates

Transversal logical operators offer the most straightforward approach to realising fault-tolerant quantum computation since they naturally limit the spread of errors. A unitary operator U is transversal with respect to a partition[2] $\{Q_i\}$ of the physical qubits of an $[\![n, k, d]\!]$ code if it can be expressed as $U = \bigotimes_i U_i$, where each U_i acts non-trivially only on qubits in Q_i [16]. The usual notion of transversal gates [13] is found choosing the singleton partition $\{\{i\}\}$. By construction transversal gates do not spread errors between qubits in different subsets and are therefore inherently fault-tolerant, provided that all the subsets in the partition are correctable[3]. We say that a partition $\{Q_i\}$ is m-local if all its subsets have size at most m: $|Q_i| \leq m$. The partition-distance $\delta_{\{Q_i\}}$ is the minimum number of subsets in the partition that supports a logical operator. Equivalently, the code can detect all errors that are supported on at most $\delta_{\{Q_i\}} - 1$ subsets. The partition-distance is a measure of how many faulty factors U_i the code can tolerate without corrupting the logical information. In the same way, we can think of the distance of a code as a measure of how many faulty 'identity factors' a code can deal with. As an example, for any $[\![n, k, d]\!]$ stabiliser code, the singleton partition $\{\{i\}\}$ is 1-local, has partition distance $\delta_{\{\{i\}\}} = d$ and the logical Pauli operators are transversal with respect to it.

In this Section we propose a partition for square codes of partition-distance $\lfloor d/2 \rfloor$ and one for symmetric codes of partition-distance d; for both partitions, we report some examples of transversal operators. Importantly, as suggested in [16, 31], we expect transversal operators on (2-dimensional) hypergraph product codes to be restricted to be either Pauli or Clifford operators, hence we here focus on Clifford operators.

Clifford operators permute the Pauli operators by mapping the Pauli group on n qubits $\mathcal{P}_n$ into itself. The set of Clifford gates on n qubits, $\mathcal{C}_n$, is a group, that can be generated by:

(i) The Hadamard gate H, that maps X operators into Z operators and vice-versa, $X \leftrightarrow Z$.

(ii) The phase gate S, that maps X operators into Y operators and fixes Z operators, $X \to Y$.

(iii) The CZ gate, a two qubit gate that maps $X \otimes I \to X \otimes Z$, $I \otimes X \to Z \otimes X$ and acts trivially on Z operators.

3.1 Gates on square codes

The first partition we propose builds on the unfolding technique for the color code proposed in [32] and further studied in [33, 34, 35]. In [32, 33], the equivalence between color codes and folded planar codes is leveraged to construct transversal Clifford gates on the planar code. Here we build on that same idea to investigate Clifford gates on square hypergraph product codes.

Hypergraph product codes can be seen as a generalisation of the planar code, which indeed is the hypergraph product code HGP$(H_{\text{rep}}, H_{\text{rep}})$, where H_{rep} is the full-rank parity check matrix of the repetition code, e.g.

$$H_{\text{rep}} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}, \tag{9}$$

[2] A partition of a set is a collection of non-empty and disjoint subsets of the set, whose union is the whole set.

[3] A set of qubits is said to be correctable if it cannot contain the support of a non-trivial logical operator.

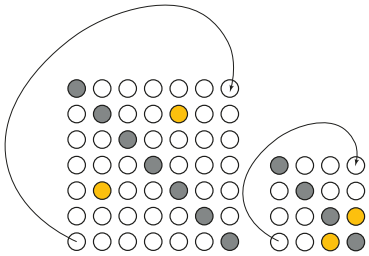


Figure 1: Graphical representation of the diagonal-twin partition for squares hypergraph product codes derived from the color code unfolding into the planar code [32, 33]. The two grids of physical qubits are folded separately along the principal diagonal and qubits that share the same sites upon folding are paired together. The grey circles represent physical diagonal qubits while all the other circles represent mirror qubits. The yellow circles (two on the left, and two on the right) are twin qubits.

for parameters $[3, 1, 3]$. By exploiting the symmetries of the canonical basis of hypergraph product codes (Theorem 1), we are able to generalise the folding of [32] to all square hypergraph product codes $\mathrm{HGP}(H, H)$. We can fold the left and right grid of qubits along the principal diagonal and pair the physical qubits whose sites overlap upon folding, see fig. 1. Upon folding, mirror physical qubits are twinned: (i, h, L) twins with (h, i, L) and (j, ℓ, R) with (ℓ, j, R). We call the partition given by singletons of diagonal qubits and two-qubit sets of twin qubits, diagonal-twin partition. More precisely, if $H \in \mathbb{F}_2^{m\times n}$ and $\mathrm{HGP}(H, H)$ is a $[\![n, k, d]\!]$ code, the diagonal-twin partition of its physical qubits is given by

$$\begin{gathered}\Big\{\{(i,i,L)\},\{(j,j,R)\}\Big\}\\ \cup\\ \Big\{\{(i,h,L),\,(h,i,L)\},\{(j,\ell,R),\,(\ell,j,R)\}\Big\}\end{gathered} \tag{10}$$

where $1 \le i, h \le n$ and $i \neq h$; $1 \le j, \ell \le m$ and $j \neq \ell$.

The diagonal-twin partition is 2-local and has partition-distance $\delta_{\mathrm{dt}} = \lfloor d/2 \rfloor$. In fact, as proven in [30], a non-trivial logical operator for an $[\![n, k, d]\!]$ hypergraph product code has support on at least d rows or columns of qubits in the same sector. Because the diagonal-twin partition is 2-local, the union of any choice of μ subsets from it has size at most 2μ. Thus, in order to fill at least d rows or d columns of physical qubits, we need to pick at least $\mu \ge d/2$ subsets in the diagonal-twin partition.

The nomenclature of physical qubits as diagonal and twins is naturally inherited by the logical qubits via the correspondence of logical qubits and their physical pivots, see table 1. This labelling is key in understanding the logical actions of transversal operations for the diagonal-twin partition and therefore we summarize it here: diagonal logical qubits are indexed as $q^L_{i,i}$ and $q^R_{j,j}$; twin logical qubits as $(q^L_{i,h}, q^L_{h,i})$ and $(q^R_{j,\ell}, q^R_{\ell,j})$ for $i \neq h$ and $j \neq \ell$, see fig. 2.

Similarly to the planar code case, both the `Hadamard-SWAP` and the `CZ-S` gates detailed below are valid logical operators on $\mathrm{HGP}(H, H)$. Whilst it is immediate to verify that the stabilisers group is preserved by these two operators, less immediate is the identification of the logical operation performed. Crucially, this task becomes trivial when we look at the action induced on a canonical basis derived by strongly lower triangular reduction as per Lemma 1.

On the physical level, `Hadamard-SWAP` consist of (i) Hadamard on every physical qubit; (ii) physical SWAP between twin qubits. `Hadamard-SWAP` is a valid logical operator as it preserves the stabiliser group mapping $S_x(j, h) \leftrightarrow S_z(h, j)$ and (i) swaps the logical X and the logical Z operators of twin qubits: on the left $q^L_{i,h}$ and $q^L_{h,i}$, on the right $q^R_{j,\ell}$ and $q^R_{\ell,j}$; (ii) acts as logical Hadamard on the diagonal qubits.

The operator `CZ-S` is defined on the physical level as: (i) S gate on left diagonal qubits; (ii) $\mathrm{S}^\dagger$ on right diagonal qubits; (iii) CZ between twin qubits. `CZ-S` preserves the Z stabilisers and maps the X stabiliser $S_x(j, h)$ into $S_x(j, h)S_z(h, j)$. Importantly, the phase factor in the product $S_x(j, h)S_z(h, j)$ is correctly preserved since, by construction, an X-stabiliser $S_x(j, h)$ has a diagonal left qubit (h, h) in its support if and only if $H_a[j, h] = 1$, if an only if it has a diagonal right qubit

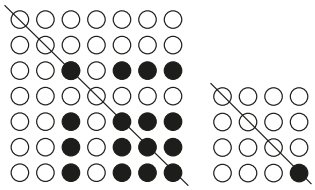


Figure 2: Graphical representation of the physical qubits of the square code $\mathrm{HGP}(\tilde{H}, \tilde{H})$, where $\tilde{H}$ is as in eq. (11). The physical qubits (white circles) are arranged on a left and right grid. The black circles highlight the position of the physical pivots associated to the logical qubits, see table 1. The canonical basis pictured is derived as explained in Lemma 1 from the matrix A of eq. (4), whose columns are a strongly lower triangular basis of $\ker \tilde{H}$.

(j, j) in its support too. As such, if an S gate is applied, an $\mathrm{S}^\dagger$ gate is applied too and the global phase cancels out. Again by looking at the action of the operator `CZ-S` on a canonical basis, since twin logical operators have support on mirror qubits only and each diagonal logical operator has support on exactly one diagonal qubit, we find that the logical action of the operator `CZ-S` is (i) S on left diagonal qubits; (ii) $\mathrm{S}^\dagger$ on right diagonal qubits; (iii) CZ between twin qubits.

3.1.1 An example of a square code

As a guiding example, we consider the square code $\mathrm{HGP}(\tilde{H}, \tilde{H})$, where $\tilde{H}$ a is non-full-rank parity check matrix of the classical $[7, 4, 3]$ Hamming code:

$$\tilde{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}. \tag{11}$$

The matrix $\tilde{H}$ defines an equivalent $[7, 4, 3]$ classical code and $\tilde{H}^T$ defines a $[4, 1, 3]$ code. By eq. (3), $\mathrm{HGP}(\tilde{H}, \tilde{H})$ is a $[\![65, 17, 3]\!]$ CSS code. The columns of A in eq. (4) are a strongly triangular basis of $\ker \tilde{H}$ and $\tilde{v} = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}^T$ generates $\ker \tilde{H}^T$. Trivially, $\{\tilde{v}\}$ is a strongly lower triangular basis of $\ker \tilde{H}^T$, with pivot $\pi(\tilde{v}) = 4$. In fig. 2 qubits are represented as circles. We note how these are divided into a 7×7 grid of left physical qubits and a 4×4 grid of right physical qubits. The characteristic square shape of these two grids makes $\mathrm{HGP}(\tilde{H}, \tilde{H})$ a square code. The physical diagonal qubits are the ones that lie across the principal diagonals of the squares (the two black lines in fig. 2). The black circles correspond to physical pivots, one for each logical qubit of the code. As for any other square hypergraph product code derived from a classical $[n, k, d]$ code whose transpose is a $[m, k^T, d^T]$ code, there are $k^2 = 16$ left logical qubits, $k = 4$ of which diagonal, and $(k^T)^2 = 1$ right logical qubits, $k^T = 1$ of which diagonal.

3.2 Gates on symmetric codes

The second partition we propose improves on the diagonal-twin partition, having partition-distance d against $\lfloor d/2 \rfloor$ for a $[\![n, k, d]\!]$ code. The price paid is the validity of this partition only on a small class of square codes, the symmetric codes: $\mathrm{HGP}_{\mathrm{sy}}(H) = \mathrm{HGP}(H, H)$ for some H such that $\mathrm{Im}\, H = \mathrm{Im}\, H^T$.

Notably, the toric code is a symmetric hypergraph product code $\mathrm{HGP}_{\mathrm{sy}}(H_{\mathrm{toric}})$ where e.g. for $d = 3$,

$$H_{\mathrm{toric}} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

For our purposes, the key feature of symmetric hypergraph product codes is that their physical qubits can be arranged in two square grids of the same size. This fact suggests that we can pair physical qubits by superimposing the left grid of physical qubits on the right one, and pairing

qubits that sit at the same coordinate. In this way, every physical qubit (i, h, L) is paired with its *sibling* qubit (i, h, R). Explicitly, given a symmetric code $\mathrm{HGP}_{\mathrm{sy}}(H)$, with $H \in \mathbb{F}_2^{n\times n}$, we define its sibling partition as:

$$\Big\{\{(i, h, L),\, (i, h, R)\}\Big\}. \tag{12}$$

where $1 \leq i, h, \leq n$. See fig. 3 for a graphical representation of the sibling partition. The sibling partition has partition-distance $\delta_s = d$. In fact, as said above and proven in [30], every non-trivial logical operator of a hypergraph product has support on at least d rows or columns in the same sector. As such, even if the sibling partition is 2-local, every subsets in it can give a contribution of at most 1 towards the covering of an arbitrary logical operator. This observation is more general: any partition whose subsets Q_i contain at most one qubit in each sector has maximum partition-distance d. We call any such partition sector-transversal.

The `Hadamard-SWAP` operation defined above for the square codes on the diagonal-twin partition is similarly defined on the sibling partition too: physical Hadamard on all qubits and SWAP between siblings qubits yields Hadamard on all the logical qubits composed with logical SWAPs between sibling logical qubits.

Via the sibling partition is naturally defined a transversal `CZ` operator. In fact, applying physical CZ gates on all pairs of sibling qubits preserves the Z stabilisers and maps the X stabilisers $S_x(j, h)$ into $S_x(j, h)S_z(h, j)$. On the logical level, the `CZ` operator yields a CZ between pairs of sibling logical qubits $q^L_{i,h}$ and $q^R_{i,h}$.

3.2.1 An example of a symmetric code

Building on our guide example in Section 3.1.1, we illustrate a symmetric hypergraph product code derived from the $[7, 4, 3]$ Hamming code with full-rank parity check matrix H,

$$H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{13}$$

We define the symmetric Hamming code as the symmetric hypergraph product code $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$, where[4]

$$H^T H = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

The symmetric Hamming code $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$ has parameters $[\![98, 32, 3]\!]$ and stabiliser generators of weight 8. It has a rate of $k/n = 32/98 \approx 0.33$ which substantially outperforms[5] the surface and toric code rates and indeed the optimal rate of $k/n = 0.2$ for codes with $k = 1$ and $d \geq 3$. For this reason, we believe that similarly constructed symmetric codes could be promising for low-overhead, near-term quantum computing. In Appendix B we present several more examples of small ($n < 1000$) symmetric hypergraph product codes with stabiliser generators of weight $w \leq 16$ on which all the gates we describe could be implemented.

4 Completing a universal gate set

Transversal gates enable us to implement a subset of logical Clifford gates on hypergraph product codes, but this is not sufficient for performing universal fault-tolerant quantum computation. In

[4] Note that, for any matrix A, $A^T A$ is symmetric.

[5] For $d = 3$, the surface and the toric code have rate $1/13 \approx 0.08$ and $1/18 \approx 0.06$ respectively. The optimal rate for codes with $k = 1$ is achieved by the five qubit code [36].

(a) (b)

Figure 3: Graphical representation of the symmetric hamming code $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$. In both figures, the left sector qubits sit at the front while the right sector qubits sit at the back. Sibling pair of qubits are superimposed. In fig. 3a the orange filled circles represent the support of the X stabiliser $S_x(2,3)$. In fig. 3b the blue filled circles represent the support of the Z stabiliser $S_z(4,6)$.

particular, the Clifford gates detailed in Section 3 act equivalently on logical qubits of the same type (diagonal, twin or sibling) and so cannot be used to e.g. entangle arbitrary pairs of logical qubits. In this Section, we focus on symmetric hypergraph product codes and show how to implement a universal gate set using circuits that consist of sequences of sector-transversal gates[6]. Such circuits are pieceably fault-tolerant, meaning that each gate in the sequence is fault-tolerant, and therefore the overall circuit is fault-tolerant when supplemented with intermediate error correction [23]. In Section 4.1, we show how to implement entangling gates between arbitrary logical qubits by combining pieceably fault-tolerant gates. Then, in Section 4.2, we show how to complete a universal gate via state injection. Lastly, in Section 4.3, we discuss the limitations of our approach.

4.1 Pieceably fault-tolerant two-qubit entangling gates

We begin by constructing a pieceably fault-tolerant circuit for implementing a logical CZ gate between logical qubits in different sectors, via the round-robin method presented in [23].

By Claim 2 of [37], we can implement a logical CZ between two logical qubits by performing a CZ between each pair of physical qubits in the support of the logical Z operators of the qubits considered. For a symmetric code (using the notation of table 1) we can therefore implement a logical CZ between arbitrary qubits in different sectors, $q^L_{i,h}$ and $q^R_{j,\ell}$, by performing a CZ between each pair of physical qubits in the support of $\bar{Z}^L_{i,h} = (a_i \otimes f_h, 0)$ and $\bar{Z}^R_{j,\ell} = (0, f_j \otimes \beta_\ell)$. For $v \in \mathbb{F}_2^n$ we indicate by $\mathrm{supp}(v)$ the ordered set of qubits/indices in the support of the vector v i.e. $\mathrm{supp}(v) = \{i \text{ s.t. } 1 \leq i \leq n \text{ and } v[i] = 1\}$. Claim 2 of [37] states that

$$\prod_{(\eta,\gamma)\in \mathrm{supp}(a_i)\times\mathrm{supp}(\beta_\ell)} \mathrm{CZ}\ (\eta, h, L), (j, \gamma, R) \tag{14}$$

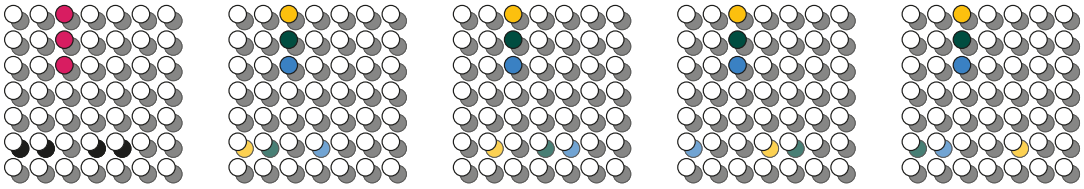
implements the logical operator

$$\mathrm{CZ}\ q^L_{i,h}, q^R_{j,\ell}.$$

In order to use the round-robin method to make the operation described by (14) fault-tolerant, we want to group the physical CZ operations in separate fault-tolerant time steps and perform intermediate error correction between them. For symmetric codes, we can achieve this if we perform physical CZ's in tranches so that each tranche only contains CZ's between left and right sector qubits—as opposed to CZ's between two qubits from the same sector. To this end, we let $\Delta = \max(|\,\mathrm{supp}(a_i)|, |\,\mathrm{supp}(\beta_\ell|)$ and, for $\eta \in \mathrm{supp}(a_i)$, we denote by $\eta_\#$ its index in $\mathrm{supp}(a_i)$, meaning $\eta_\# = \nu$ if η is the νth element in the ordered set $\mathrm{supp}(a_i)$; with a slight abuse of notation, we can write $\eta \oplus_\Delta t$ to indicate the μth element in $\mathrm{supp}(\beta_\ell)$, where $\mu = \eta_\# + t \mod \Delta$, for any integer t. Combining this notation with eq. (14), yields

$$\prod_{t=0}^{\Delta-1} \left(\prod_{\substack{\eta\in\mathrm{supp}(a_i) \\ \eta\oplus_\Delta t \leq |\,\mathrm{supp}(\beta_\ell)|}} \mathrm{CZ}(\eta, h, L), (j, \eta \oplus_\Delta t, R) \right). \tag{15}$$

[6] Transversal gates over a sector-transversal partition.



(a) Logical Z (b) $t = 0$. (c) $t = 1$. (d) $t = 2$. (e) $t = 3$.

Figure 4: Round robin implementation of CZ $q^L_{3,3}q^R_{6,5}$ by the circuit of eq. (15) on the symmetric Hamming code. (a) shows the supports of the logical Z operators. (b)-(e) illustrate the physical CZ gates acting at each time step t, where qubits highlighted with the same colour have a CZ gate applied to them.

Treating each value of t in eq. (15) as a time step, the inner summations,

$$\Omega_t = \prod_{\substack{\eta \in \mathrm{supp}(a_i) \\ \eta \oplus_\Delta t \leq |\,\mathrm{supp}(\beta_\ell)|}} \mathrm{CZ}(\eta, h, L), (j, \eta \oplus_\Delta t, R), \tag{16}$$

are sequences of non-overlapping sector-transversal operators, as desired. For each time step $0 \leq t \leq \Delta$, first we apply the gates of eq. (16) and then perform one round of error correction using the ReShape decoder [30]. The use of ReShape at this stage is fundamental because it corrects errors on different sectors independently. Hence, using ReShape to correct errors between time steps, the protocol preserve the partition-distance d of the operator Ω_t in eq. (16) and can correct up to $\lfloor d/2 \rfloor$ faulty CZ gates.

In conclusion, CZ between arbitrary left and right logical qubits can be implemented in $\sim d_z^\uparrow$ time steps, where $d_z^\uparrow$ is the maximum weight of a canonical Z operator. An example on the symmetric Hamming code is depicted in fig. 4.

We can also construct a pieceably fault-tolerant circuit for the gate

$$\mathrm{XCX} := \mathrm{H}^{\otimes 2} \cdot \mathrm{CZ} \cdot \mathrm{H}^{\otimes 2} \tag{17}$$

Indeed, combining again Claim 2 of [37] and eq. (15), we find that

$$\prod_{t=0}^{\Delta-1} \left(\prod_{\substack{\eta \in \mathrm{supp}(b_h) \\ \eta \oplus_\Delta t \leq |\,\mathrm{supp}(\alpha_j)|}} \mathrm{XCX}(i, \eta, L), (\eta \oplus_\Delta t, \ell, R) \right) \tag{18}$$

implements the logical operator

$$\mathrm{XCX}\ q^L_{i,h}, q^R_{j,\ell}.$$

where $\bar{X}^L_{i,h} = (f_i \otimes b_h, 0)$ and $\bar{X}^R_{j,\ell} = (0, \alpha_j \otimes f_\ell)$ and $\Delta = \max(|\,\mathrm{supp}(b_h)|, |\,\mathrm{supp}(\alpha_j)|)$. As in the CZ case, we can use the round-robin method in [23] and perform error correction between each sector-transversal time step to obtain a fault-tolerant implementation of XCX.

Given our ability to perform CZ and XCX gates between arbitrary pairs of qubits in different sectors, we can use the circuit identity shown in fig. 5 to implement CNOT gates between arbitrary pairs of qubits in the same sector. To summarize, we can implement entangling gates between arbitrary logical qubits using circuits composed of at most four pieceably fault-tolerant circuits.

4.2 Single-qubit gates via state injection

To complete a universal gate set, it is sufficient to implement the H, S and T gates. The gates S and T can be implemented via state injection [38, 39, 40]. We can use a CSS code as a state factory and then use a pieceably fault-tolerant CZ gate to implement the state injection circuits shown in fig. 6. State injections can be performed from an auxiliary $[\![n', 1, d']\!]$ CSS code $\mathcal{C}$ that acts as a state factory by leveraging pieceably fault-tolerant gates to implement the circuits in fig. 6 at a logical level. More precisely, let $\zeta \in \mathbb{F}_2^{n'}$ describe the support of the logical Z operator for the

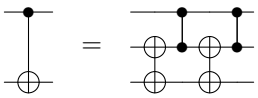


Figure 5: Circuit identity showing how to construct CNOT from CZ and XCX gates, where XCX gates are depicted with targets on both qubits.

Figure 6: State injection gadgets for implementing S and T gates. The top wire in each circuit represents a logical qubit of a a symmetric hypergraph product code and the bottom wire represents a logical qubit of an ancillary CSS code. To implement the CZ gates we use the pieceably fault-tolerant circuit in eq. (15). Note that for CSS codes the X basis measurements, represented as the gate H followed by Z basis measurements, can be done transversally.

logical qubit $q_{\mathcal{C}}$ of $\mathcal{C}$ and suppose that we number the physical qubits of $\mathcal{C}$ as $(\iota)_{\mathcal{C}}$. Then, via Claim 2 of [37] and eq. (15), we obtain that

$$\prod_{t=0}^{\Delta-1} \left(\prod_{\substack{\eta \in \mathrm{supp}(a_i) \\ \eta \oplus_\Delta t \leq |\zeta|}} \mathrm{CZ}\ (\eta, h, L), (\eta \oplus_\Delta t)_{\mathcal{C}} \right) \tag{19}$$

where $\Delta = \max\left(|\,\mathrm{supp}(a_i)|, |\,\mathrm{supp}(\zeta)|\right)$, implements

$$\mathrm{CZ}\ q^L_{i,h}, q_{\mathcal{C}}.$$

And similarly for a right qubit $q^R_{j,\ell}$. The operator of eq. (19) consists of CZ operators between different codes so is transversal at each time step and is therefore pieceably fault-tolerant. As $\mathcal{C}$ is a CSS code, we can measure the logical X operator destructively by measuring all of the physical qubits in the X basis and performing classical error correction on the measurement outcomes. Therefore, to implement S, and T we only need to fault-tolerantly prepare the states $HS\,|+\rangle$ and $HT\,|+\rangle$. For a $d = 3$ symmetric hypergraph product code, one option would be to produce these states using the $[\![15, 1, 3]\!]$ Reed-Muller code [41, 42, 43]. For higher distance symmetric hypergraph product codes, preparing magic states could be accomplished using standard magic state distillation techniques [40, 44, 45, 46]. In addition, multiple magic states could also be injected from a CSS code with multiple encoded qubits, using parallel CZ gates (see section 4.3).

For the implementation of the logical H gate on single qubits we propose the use of the `Hadamard-Swap` operation detailed in Section 3.2 together with three logical S gates. Without loss of generality we here illustrate how to implement the gate H on the left logical qubit $q^L_{i,h}$ with sibling qubit $q^R_{i,h}$. The construction for right logical qubits follows easily by switching the roles of siblings. We indicate by $U(q)$ the logical gate U on the logical qubit q, i.e. $\mathrm{S}(q^L_{i,h})$ indicates the logical S gate on the logical qubit $q^L_{i,h}$. The composition of gates:

$$\mathrm{S}(q^L_{i,h})\ (\ \texttt{Hadamard-SWAP}\ \mathrm{S}(q^R_{i,h})\ \texttt{Hadamard-SWAP}\ )\ \mathrm{S}(q^L_{i,h}) \tag{20}$$

equates:

$$\mathrm{H\ S\ H\ S\ H}\ \ (q^L_{i,h})\,.$$

Since

$$\mathrm{H}\ (\mathrm{S}\ \mathrm{H}\ \mathrm{S})\ \mathrm{H} = e^{i\pi/4}\ \mathrm{H} \tag{21}$$

the composition of gates in eq. (20) implements the logical H gate on qubit $q^L_{i,h}$, up to a global phase factor. Hence, the logical H gate on arbitrary qubits can be implemented at the cost of three S gates and two sector-transversal operations.

4.3 Limitations of pieceable fault-tolerance

We conclude this Section by highlighting two important limitations of pieceable fault tolerant techniques: intermediate correction and time cost. To guide our analysis, let us consider the pieceably fault-tolerant CZ gate of Section 4.1.

Since CZ is diagonal in the Z basis, Z stabilisers are left unchanged during the protocol and hence correction for X errors can be done, at each time step, via standard measurement of Z stabilisers. However, the original X stabilizer generators are transformed at each intermediate time step, with no guarantee on their weight—which could scale with the code distance. To avoid measuring high-weight generators, we can neglect to do error correction for Z errors at intermediate time steps, at the cost of allowing Z errors to build up for a constant number of rounds (for codes of a fixed distance). This strategy will only be fault-tolerant for codes of a fixed distance and therefore will not give a threshold without modification[7]. Nevertheless, the asymptotic behaviour encapsulated by a threshold is less important in the short- to medium-term regime where space overhead will likely be the most important constraint.

As regards to time cost, pieceably fault-tolerant circuits necessarily have a time overhead higher than transversal gates because of intermediate error-correction. If the time cost of one cycle of error-correction followed by the application of a transversal gate is τ, then each sector-transversal gate has time cost τ. By contrast, the pieceably fault-tolerant CZ of eq. (15), has cost at least τd for a distance d code. For instance, in the symmetric Hamming code example given in section 3.2.1, CZ or XCX have cost $\tau 4$ because 4 is the maximum weight of a canonical logical operator.

In general, composing m pieceable fault tolerant gates takes time $\tau d^{\uparrow} m$, where $d^{\uparrow}$ is the maximum weight of a canonical logical operator. However, this overhead can be reduced via parallelization. For instance,

$$\mathrm{CZ}\ q^L_{i,h},\ q^R_{j,\ell} \quad \text{and} \quad \mathrm{CZ}\ q^L_{i',h'},\ q^R_{j',\ell'}$$

can be performed in parallel via eq. (15), provided that the logical Z operators of the qubits considered have all disjoint support e.g. whenever $h \neq h'$ and $j \neq j'$; see fig. 7 for an example. Therefore, if we want to implement m pieceable fault-tolerant gates, we can divide them into subsets of parallelizable gates and, if n_p is the minimum number of gates in any of these subsets, we can implement all the gates in time $\tau d^{\uparrow} m/n_p$, reducing the average time overhead per gate. Importantly, performing gates in parallel has no additional space overhead, in contrast to the measurement-based scheme of [11].

5 Conclusion

We have investigated the structure of hypergraph product codes to characterize transveral Clifford gates on them. In fact, we proved that every hypergraph product code has a canonical basis for its logical space whose features suggest qualitative differences between logical qubits (diagonal, twin and sibling). We showcased how to leverage such differences to implement some transversal Clifford gates on square and symmetric hypergraph product codes. The logical transversal gates we have found are limited but we comment on how these could be augmented via pieceable fault tolerance and state injection to achieve universality.

While we have focused on symmetric hypergraph product codes, we believe that the choice of structured seed classical codes in the hypergraph product could be exploited to construct other

[7] Another possible solution is to perform weight reduction [11, 47, 48] on the X stabiliser generators at each time step, likely at the cost of a high space overhead.

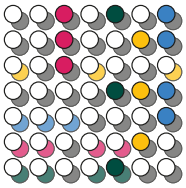


Figure 7: Graphical representation of the parallel implementation of pieceably fault tolerant gates. Columns of left (front) qubits and rows of right qubits (back) filled in the same color represent pairs of logical operators whose supports do not overlap. In the image, magenta: $q^L_{3,3}, q^R_{6,5}$, green $q^L_{7,5}, q^R_{7,5}$, yellow: $q^L_{6,6}, q^R_{3,7}$, blue: $q^L_{5,3} q^R_{5,7}$. Pieceably fault tolerant gates that act on the support of these logical operators (e.g. CZ) can be implemented in parallel. We note that, for any symmetric hypergraph product code of dimension k^2, we can always find k pairs of logical qubits whose logical operators do not overlap as shown here (one for each left line pivot and one for each right line pivot).

transversal gates. We also conjecture that a partitioning strategy similar to the ones here presented could be used to implement transversal non-Clifford gates on higher dimensional homological codes [34, 21]. More generally, it would be interesting to extend the construction of a canonical basis to the more efficient product codes construction such as the ones in [49, 50, 51, 52].

Acknowledgements

This work is supported by the Australian Research Council via the Centre of Excellence in Engineered Quantum Systems (EQUS) project number CE170100009. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Colleges and Universities. The authors would like to thank Stephen Bartlett, Earl Campbell and Lawrence Cohen for helpful discussions.

References

[1] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. "Quantum supremacy using a programmable superconducting processor". Nature **574**, 505–510 (2019).

[2] Peter W. Shor. "Fault-tolerant quantum computation". In Proceedings of 37th Conference on Foundations of Computer Science. Pages 56–65. (1996).

[3] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". Quantum **5**, 433 (2021).

[4] Isaac H Kim, Eunseok Lee, Ye-Hua Liu, Sam Pallister, William Pol, and Sam Roberts. "Fault-tolerant resource estimate for quantum chemical simulations: Case study on Li-ion battery electrolyte molecules". Physical Review Research **4**, 023019 (2022).

[5] Nikolas P Breuckmann and Jens Niklas Eberhardt. "Quantum low-density parity-check codes". PRX Quantum **2**, 040101 (2021).

[6] A Yu Kitaev. "Fault-tolerant quantum computation by anyons". Annals of Physics **303**, 2–30 (2003).

[7] Pavel Panteleev and Gleb Kalachev. "Asymptotically good quantum and locally testable classical LDPC codes" (2021). arXiv:2111.03654.

[8] Daniel Gottesman. "Fault-tolerant quantum computation with constant overhead". Quantum Information & Computation **14**, 1338–1372 (2014).

[9] Anirudh Krishna and David Poulin. "Fault-tolerant gates on hypergraph product codes". Physical Review X **11**, 011023 (2021).

[10] Anirudh Krishna and David Poulin. "Topological wormholes: Nonlocal defects on the toric code". Physical Review Research **2**, 023116 (2020).

[11] Lawrence Z Cohen, Isaac H Kim, Stephen D Bartlett, and Benjamin J Brown. "Low-overhead fault-tolerant quantum computing using long-range connectivity" (2021). arXiv:2110.10794.

[12] Bei Zeng, Andrew Cross, and Isaac L. Chuang. "Transversality versus universality for additive quantum codes". IEEE Transactions on Information Theory **57**, 6272–6284 (2011).

[13] Bryan Eastin and Emanuel Knill. "Restrictions on transversal encoded quantum gate sets". Physical Review Letters **102**, 110502 (2009).

[14] Sergey Bravyi and Robert König. "Classification of topologically protected gates for local stabilizer codes". Phys. Rev. Lett. **110**, 170503 (2013).

[15] Fernando Pastawski and Beni Yoshida. "Fault-tolerant logical gates in quantum error-correcting codes". Phys. Rev. A **91**, 012305 (2015).

[16] Tomas Jochym-O'Connor, Aleksander Kubica, and Theodore J Yoder. "Disjointness of stabilizer codes and limitations on fault-tolerant logical gates". Physical Review X **8**, 021047 (2018).

[17] Paul Webster, Michael Vasmer, Thomas R Scruby, and Stephen D Bartlett. "Universal fault-tolerant quantum computing with stabilizer codes". Physical Review Research **4**, 013092 (2022).

[18] Sergey Bravyi and Matthew B Hastings. "Homological product codes". In Proceedings of the 46th Annual ACM Symposium on Theory of Computing. Pages 273–282. ACM (2014).

[19] Tomas Jochym-O'Connor. "Fault-tolerant gates via homological product codes". Quantum **3**, 120 (2019).

[20] Armanda O Quintavalle, Michael Vasmer, Joschka Roffe, and Earl T Campbell. "Single-shot error correction of three-dimensional homological product codes". PRX Quantum **2**, 020340 (2021).

[21] Tomas Jochym-O'Connor and Theodore J Yoder. "Four-dimensional toric code with non-Clifford transversal gates". Physical Review Research **3**, 013118 (2021).

[22] Shai Evra, Tali Kaufman, and Gilles Zémor. "Decodable quantum LDPC codes beyond the $\sqrt{n}$ distance barrier using high dimensional expanders" (2020). arXiv:2004.07935.

[23] Theodore J Yoder, Ryuji Takagi, and Isaac L Chuang. "Universal fault-tolerant gates on concatenated stabilizer codes". Physical Review X **6**, 031039 (2016).

[24] David JC MacKay. "Information theory, inference and learning algorithms". Cambridge University Press. (2003).

[25] A Robert Calderbank and Peter W Shor. "Good quantum error-correcting codes exist". Physical Review A **54**, 1098 (1996).

[26] Andrew Steane. "Multiple-particle interference and quantum error correction". Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **452**, 2551–2577 (1996).

[27] Michael A Nielsen and Isaac Chuang. "Quantum computation and quantum information". Cambridge University Press. (2002).

[28] Jean-Pierre Tillich and Gilles Zémor. "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength". IEEE Transactions on Information Theory **60**, 1193–1202 (2014).

[29] Benjamin Audoux and Alain Couvreur. "On tensor products of CSS codes" (2015). arXiv:1512.07081.

[30] Armanda O Quintavalle and Earl T Campbell. "Lifting decoders for classical codes to decoders for quantum codes" (2021). arXiv:2105.02370.

[31] Simon Burton and Dan Browne. "Limitations on transversal gates for hypergraph product codes". IEEE Transactions on Information Theory **68**, 1772–1781 (2022).

[32] Aleksander Kubica, Beni Yoshida, and Fernando Pastawski. "Unfolding the color code". New Journal of Physics **17**, 083026 (2015).

[33] Jonathan E Moussa. "Transversal Clifford gates on folded surface codes". Physical Review A **94**, 042316 (2016).

[34] Michael Vasmer and Dan E Browne. "Three-dimensional surface codes: Transversal gates and fault-tolerant architectures". Physical Review A **100**, 012312 (2019).

[35] Nikolas P Breuckmann and Simon Burton. "Fold-transversal Clifford gates for quantum codes" (2022). arXiv:2202.06647.

[36] Raymond Laflamme, Cesar Miquel, Juan Pablo Paz, and Wojciech Hubert Zurek. "Perfect quantum error correcting code". Physical Review Letters **77**, 198 (1996).

[37] Rui Chao and Ben W Reichardt. "Fault-tolerant quantum computation with few qubits". npj Quantum Information **4**, 1–8 (2018).

[38] Daniel Gottesman and Isaac L. Chuang. "Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations". Nature **402**, 390–393 (1999).

[39] Xinlan Zhou, Debbie W. Leung, and Isaac L. Chuang. "Methodology for quantum logic gate construction". Physical Review A **62**, 052316 (2000).

[40] Sergey Bravyi and Alexei Kitaev. "Universal quantum computation with ideal Clifford gates and noisy ancillas". Physical Review A **71**, 022316 (2005).

[41] Emanuel Knill, Raymond Laflamme, and Wojciech Zurek. "Resilient quantum computation". Science **279**, 342–345 (1996).

[42] A.M. Steane. "Quantum Reed-Muller codes". IEEE Transactions on Information Theory **45**, 1701–1703 (1999).

[43] Jonas T. Anderson, Guillaume Duclos-Cianci, and David Poulin. "Fault-tolerant conversion between the Steane and Reed-Muller quantum codes". Physical Review Letters **113**, 080501 (2014).

[44] Sergey Bravyi and Jeongwan Haah. "Magic-state distillation with low overhead". Physical Review A **86**, 052329 (2012).

[45] Daniel Litinski. "Magic state distillation: Not as costly as you think". Quantum **3**, 205 (2019).

[46] Christopher Chamberland and Kyungjoo Noh. "Very low overhead fault-tolerant magic state preparation using redundant ancilla encoding and flag qubits". npj Quantum Information **6**, 1–12 (2020).

[47] Matthew B Hastings. "Weight reduction for quantum codes". Quantum Information & Computation **17**, 1307–1334 (2016).

[48] MB Hastings. "On quantum weight reduction" (2021). arXiv:2102.10030.

[49] Pavel Panteleev and Gleb Kalachev. "Degenerate quantum LDPC codes with good finite length performance". Quantum **5**, 585 (2021).

[50] Matthew B Hastings, Jeongwan Haah, and Ryan O'Donnell. "Fiber bundle codes: breaking the $N^{1/2}$polylog(N) barrier for quantum LDPC codes". In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing. Pages 1276–1288. (2021).

[51] Nikolas P Breuckmann and Jens N Eberhardt. "Balanced product quantum codes". IEEE Transactions on Information Theory **67**, 6653–6674 (2021).

[52] Anthony Leverrier and Gilles Zémor. "Quantum tanner codes" (2022). arXiv:2202.13641.

[53] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma algebra system I: The user language". Journal of Symbolic Computation **24**, 235–265 (1997).

[54] Markus Grassl. "Bounds on the minimum distance of linear codes" (2008). `http://www.codetables.de`.

A Proofs

In this Section we introduce the strongly lower triangular Gaussian reduction for binary matrices (Algorithm 1) and prove its correctness.

Algorithm 1 takes in input a binary matrix H and outputs a strongly lower triangular basis for its kernel, as per Definition 1, and a unit vector basis for its complement [8]. It iterates over the columns of H (line 4) and iteratively removes from the set of columns indices (line 8) the independent columns; the matrix of interest, K, which will contain the column vectors forming a basis of the kernel of H is iteratively reduced in triangular form (line 10). We prove that Algorithm 1 is correct in Lemma 2 below.

[8] We remind the reader that, given a vector space $V \subseteq \mathbb{F}_2^n$, its complement $V^\bullet$ is any vector space such that $V \oplus V^\bullet = \mathbb{F}_2^n$, see the proof of Lemma 1.

Algorithm 1 Strong lower triangular Gaussian reduction.

Input: An $m \times n$ binary matrix H.

Output: A strongly triangular basis of its kernel $\mathcal{K}$ and a unit vector basis $\mathcal{F}$ of $(\operatorname{Im} H^T)^\bullet$, where $\mathbb{F}_2^n = \operatorname{Im} H^T \oplus (\operatorname{Im} H^T)^\bullet$.

1: $\hat{H} \leftarrow H$
2: $K \leftarrow I_n$
3: $\pi(\mathcal{K}) = \{1, \dots, n\}$
4: **for** $j \leftarrow 1$ **to** n **do**
5: **while** $\hat{H}[i][j] \neq 1$ and $1 \leq i \leq m$ **do**
6: $i \leftarrow i + 1$
7: **end while**
8: **if** $\hat{H}[i][j] = 1$ **then**:
9: $\pi(\mathcal{K}) \leftarrow \pi(\mathcal{K}) \setminus \{j\}$
10: **for** $\ell \leftarrow j + 1$ **to** n **do**
11: **if** $\hat{H}[i][\ell] = 1$ **then**
12: $\hat{H}^\ell \leftarrow \hat{H}^j + \hat{H}^\ell$
13: $K^\ell \leftarrow K^j + K^\ell$
14: **end if**
15: **end for**
16: **end if**
17: **end for**
18: $\mathcal{K} \leftarrow \{K^j : j \in \pi(\mathcal{K})\}$
19: $F \leftarrow \{I_n^j : j \in \pi(\mathcal{K})\}$
20: **return** $\mathcal{K}$, $\mathcal{F}$

Lemma 2. *Algorithm 1 terminates and is correct. More precisely:*

1) The set $\mathcal{H} = \{H^j : 1 \leq j \leq n,\, j \notin \pi(\mathcal{K})\}$ *is a basis of the column span of* H.

2) The set $\mathcal{K}$ *is a basis of the kernel of* H.

3) The set $\mathcal{F}$ *is a unit-vector basis of* $(\operatorname{Im} H^T)^\bullet$.

Proof. Let v be any vector in the column span of H:

$$v = \sum_{j \in V} H^j.$$

If $V \cap \pi(\mathcal{K}) = \emptyset$ then there is nothing to prove. Conversely, suppose that there exists $j \in V$ such that $j \in \pi(\mathcal{K})$. Then it must be $\hat{H}^j = 0$ (see line 8). In other words, the jth column can be written as a linear combination of other columns of H, and by substitution if necessary, point 1) is proved.

To prove point 2), observe that for what said on the zero columns of $\hat{H}$, all the vectors in $\mathcal{K}$ are in the kernel of H. Moreover, they are linearly independent and lower triangular by construction. Now suppose that the matrix is not strongly lower triangular (e.g. $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$) so that there exists a row index i such that $K_{i,\alpha} = K_{i,\alpha+\tau} = 1$ for some $\tau > 0$. This means that the αth column K^α has been added to the $(\alpha + \tau)$th column $K^{\alpha+\tau}$ at step α. However, if $\alpha \in \pi(\mathcal{K})$ is a pivot index, $\hat{H}^\alpha$ is zero at steps $\alpha - 1, \dots, n$ and therefore the if condition at line 11 is not met.

Point 3) follows observing that the vectors in $\mathcal{F}$ are linearly independent by construction (they are unit vectors with different pivots) and none of them belongs to the row span of H. In fact by definition, any vector in the kernel of H is orthogonal to vectors in the row span of H, and by strong lower triangularity, $\langle k, f \rangle = 1$ for any $k \in \mathcal{K}$ and $f \in \mathcal{F}$. □

B Symmetric hypergraph product codes - Examples

We here provide table 2 with some examples of classical seed matrices to construct symmetric hypergraph product codes with $n < 1000$ and maximum stabiliser weight $w \leq 16$. The classical parity check matrices used in the product were found with assistance from the "Best Known Linear Code" database of MAGMA [53] and guidance from the bounds in [54]. For each parity check matrix H in the first column, we have computed the $[\![n, k, d]\!]$ parameters the symmetric hypergraph product code $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$.

Classical Code Parity Check	n	k	k/n	d	w
$\begin{smallmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{smallmatrix}$	98	32	0.33	3	8
$\begin{smallmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{smallmatrix}$	242	98	0.41	3	12
$\begin{smallmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{smallmatrix}$	450	242	0.54	3	16
$\begin{smallmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{smallmatrix}$	98	18	0.18	4	8
$\begin{smallmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{smallmatrix}$	288	98	0.34	4	12
$\begin{smallmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{smallmatrix}$	200	18	0.09	5	8
$\begin{smallmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{smallmatrix}$	242	32	0.13	5	16
$\begin{smallmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{smallmatrix}$	392	32	0.08	7	16
$\begin{smallmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{smallmatrix}$	722	32	0.04	9	16

Table 2: Examples of symmetric hypergraph product codes $\mathrm{HGP}_{\mathrm{sy}}(H^T H)$, where H is the binary matrix in the first column. The length n, the dimension k, the rate k/n, the distance d and the maximum stabiliser weight w are reported in the others columns.

Chapter 5

Single-shot error correction - Confinement

Context and Results

In [1] we investigate the theoretical requirements a code must have to be robust against syndrome measurement errors. In the literature there are two proposed strategies to address noisy measurements: repeated measurement over time [2, 3] and using a single-shot code and decoder [4, 5]. By repeating the same stabilizer measurements multiple times we can infer the most likely measurement result via a majority-vote strategy. In the single-shot setting instead, measurements are performed only once but we allow for some residual error to be present on the code's data qubits after one round of correction. A decoder succeeds if the residual error on the data qubits is kept under control after repeated rounds of single-shot error correction, opposite to the standard scenario where a decoder succeeds if no error is left on the register after correction.

We propose the property of confinement of the syndrome function as essential for single-shot error correction. Loosely, a code has confinement if, for small enough error, the syndrome weight grows with the error weight. Our main result is proving that confinement is indeed sufficient for codes to exhibit some form of single-shot behaviour. Further, we illustrate how the original hypergraph product code construction [6] can be iterated, and made 'three-dimensional' to ensure that the associated quantum code has confinement, independently of the classical codes chosen as seeds in the product.

Limitations

To date, no code family is known to be single-shot without also having the confinement property as defined in [1]. Indeed we shown that confinement is a *sufficient* condition for single-shot error correction. Nonetheless, we were after a *necessary* condition for it and we conjecture that in fact confinement and single-shot properties are equivalent.

Another, minor, limitation of this work is found in the numerical results obtained. We have simulated single-shot decoding on three-dimensional hypergraph product codes using what we call a two-stage decoder. This choice is justified by the proof structure of our theoretical result. However, better numerical performances are possible when the first stage of our decoder is dropped [7].

Authorship declaration

AOQ derived the proofs and wrote the corresponding sections of the manuscript: Sections III and IV; Appendices A, B, C and D. All the authors contributed to Sections I, II and VI.

References

[1] Armanda O Quintavalle et al. "Single-shot error correction of three-dimensional homological product codes". In: *PRX Quantum* 2.2 (2021), p. 020340.

[2] Eric Dennis et al. "Topological quantum memory". In: *Journal of Mathematical Physics* 43.9 (2002), pp. 4452–4505.

[3] Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. "High-threshold universal quantum computation on the surface code". In: *Physical Review A* 80.5 (2009), p. 052312.

[4] Hector Bombin. "Single-shot fault-tolerant quantum error correction". In: *Physical Review X* 5.3 (2015), p. 031043.

[5] Earl T Campbell. "A theory of single-shot error correction for adversarial noise". In: *Quantum Science and Technology* 4.2 (2019), p. 025006.

[6] Jean-Pierre Tillich and Gilles Zémor. "Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength". In: *IEEE Transactions on Information Theory* 60.2 (2013), pp. 1193–1202.

[7] Oscar Higgott and Nikolas Breuckmann. "Improved single-shot decoding of higher dimensional homological product codes". In: *Bulletin of the American Physical Society* (2022).

Editors' Suggestion

Single-Shot Error Correction of Three-Dimensional Homological Product Codes

Armanda O. Quintavalle[1,*] Michael Vasmer[2,3,†] Joschka Roffe[1,4,‡] and Earl T. Campbell[1,5,§]

[1] *Department of Physics and Astronomy, University of Sheffield, Sheffield S3 7RH, United Kingdom*
[2] *Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada*
[3] *Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada*
[4] *Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*
[5] *AWS Center for Quantum Computing, Cambridge, CB1 2GA, United Kingdom*

(Received 17 December 2020; accepted 20 April 2021; published 14 June 2021)

Single-shot error correction corrects data noise using only a single round of noisy measurements on the data qubits, removing the need for intensive measurement repetition. We introduce a general concept of confinement for quantum codes, which roughly stipulates qubit errors cannot grow without triggering more measurement syndromes. We prove confinement is sufficient for single-shot decoding of adversarial errors and linear confinement is sufficient for single-shot decoding of local stochastic errors. Further to this, we prove that all three-dimensional homological product codes exhibit confinement in their X components and are therefore single shot for adversarial phase-flip noise. For local stochastic phase-flip noise, we numerically explore these codes and again find evidence of single-shot protection. Our Monte Carlo simulations indicate sustainable thresholds of 3.08(4)% and 2.90(2)% for three-dimensional (3D) surface and toric codes, respectively, the highest observed single-shot thresholds to date. To demonstrate single-shot error correction beyond the class of topological codes, we also run simulations on a randomly constructed family of 3D homological product codes.

DOI: 10.1103/PRXQuantum.2.020340

I. INTRODUCTION

Quantum error correction encodes logical quantum information into a codespace [1]. Given perfect measurement of the codespace stabilizers we obtain the syndrome of any error present. A suitable decoding algorithm can determine a recovery operation that returns the system to the codespace. Either this recovery is a perfect success, or a failure resulting in a high weight logical error. However, in real quantum systems the measurements are not perfect and this simple story becomes more involved. The three main strategies for tackling noisy measurements are as follows: repeated measurements on the code [2,3]; performing measurement driven error correction on a cluster state [4–9]; or using a single-shot code and decoder [10]. Focusing on the last strategy, the single-shot approach has the advantage of no additional time cost or cluster-state generation cost and provides a resilience against time-correlated noise [11]. In single-shot error correction, some residual error persists after each round of error correction, but this residual error is kept small and does not rapidly accumulate. However, only a special class of codes support single-shot error correction, but exactly which codes and why is not yet fully understood.

Bombín coined the phrase single-shot error correction and remarked that it "*is related to self-correction and confinement phenomena in the corresponding quantum Hamiltonian model.*" [10]. He defined confinement for subsystem codes, and showed that it is sufficient for single-shot error correction with a limited class of subsystem codes. In particular, he proved that the three-dimensional (3D) gauge color code supports single-shot error correction, though it is unknown whether the corresponding Hamiltonian exhibits self-correction. Later single-shot error correction was numerically observed in a variety of higher-dimensional topological codes, including the following: the 3D gauge color code [12], four-dimensional (4D) surface codes [13] and their hyperbolic cousins [14], and 3D surface codes with phase noise [15–17]. Campbell established a general set of sufficient conditions, encapsulated by a code property called good soundness, that ensured adversarial noise could be suppressed using a

*armandaoq@gmail.com
†mvasmer@perimeterinstitute.ca
‡joschka@roffe.eu
§earltcampbell@gmail.com

single-shot decoder [18]. While Campbell's sufficiency conditions explained single-shot error correction in a wide range of codes, around the same time it was shown that quantum expander codes [19–21] supported single-shot error correction [22]. However, quantum expander codes lack the soundness property so neither Bombín's notion of confinement or Campbell's notion of soundness is sufficient to encompass all known examples of single-shot error correction. Our work provides the first framework that captures all known forms of single-shot error correction, encompassing both previous approaches within a single theory.

We can use different classical algorithms to decode a given quantum code, and this choice will affect the utility of the code. Different decoders have various time complexities and error tolerances, which affects the resources required by a quantum computer based on the code [23–25]. Thus far, single-shot decoders come in two flavors. The first are two-stage decoders [12,13], where stage 1 decoding repairs the noisy syndrome using redundancy in the parity check measurements and stage 2 decoding solves the corrected syndrome problem. The second flavor of decoders computes a correction from the noisy syndrome without attempting to repair it. Most examples of such decoders are local decoders, meaning that the whole correction is made up of corrections computed in small local regions of the code using syndrome information in the immediate neighborhood [14–17,21,26,27]. However, there are some examples of nonlocal decoders such as belief propagation (BP) being used for single-shot error correction without syndrome repair [14,27]. A natural question to ask is the following: what is the optimal decoding strategy for single-shot codes? Even in the simple case of the 3D toric code this is not well understood.

The remainder of this paper is structured as follows. In Sec. II, we give a summary of our results. In Sec. III, we formally state our results on confinement and single-shot decoding. In Sec. IV, we detail the construction of 3D product codes. In Sec. V, we present our numerical simulations and analyse their results. Finally, in Sec. VI, we discuss future research directions that flow from this work.

II. SUMMARY OF RESULTS

This paper is in two parts: on the one hand, we propose the concept of confinement as an essential characteristic for a code family to display single-shot properties; on the other, we investigate the single-shot performances of the class of 3D homological product codes [19,28,29], which we call 3D product codes. First, we introduce confinement in Sec. III. Loosely, confinement stipulates that syndrome weight must increase with qubit weight, under some caveats. We formalize the notion of a code family having good confinement, which we prove is a sufficient condition for single-shot decoding in the adversarial noise setting. In addition to that, we prove that good *linear* confinement is a sufficient condition for a family of codes to exhibit a sustainable single-shot threshold for local stochastic noise (Appendix A). Second, we review the construction of the 3D product codes in Sec. IV, and show that the 3D surface and toric codes are particular instances of this more general class of codes in Appendix B. We prove that all 3D product codes have (cubic) confinement for phase-flip errors (Appendix C), and therefore have single-shot error correction for adversarial phase-flip noise. We expect these codes to have single-shot error correction for local stochastic phase-flip noise as well. In fact, our definition of confinement generalizes the definition proposed by Bombín [10] for the gauge color code and the notion of robustness for expander codes [20]; since both class of codes are proven to have a single-shot threshold for local stochastic noise [10,22] we conjecture that low-density parity-check (LDPC) codes with good (superlinear) confinement have a threshold too. We investigate this case numerically.

In the single-shot setting, the code always has some residual error present and the error-correction procedure introduces noise correlations in subsequent rounds of single-shot error correction. How then do we assess success or failure of a decoding algorithm? The concept of sustainable threshold was proposed by Brown *et al*. [12] as a metric for single-shot codes and decoders. We use $p_{\text{th}}(N)$ to denote the threshold of a code-decoder family given N cycles of qubit noise, noisy syndrome extraction, and single-shot decoding, with the N^{th} cycle followed by a single round of noiseless syndrome extraction and decoding. The final round ensures that we return the system to the codespace and assess success by the absence of a logical error. We define the sustainable threshold of the code-decoder family to be

$$p_{\text{sus}} = \lim_{N\to\infty} p_{\text{th}}(N). \tag{1}$$

Numerically, this is estimated by plotting $p_{\text{th}}(N)$ against N and fitting to the following ansatz,

$$p_{\text{th}}(N) = p_{\text{sus}}\{1 - [1 - p_{\text{th}}(0)/p_{\text{sus}}]e^{-\gamma N}\}. \tag{2}$$

We numerically estimate the sustainable error thresholds of 3D toric and surface codes for two different two-stage decoders. We surpass all previous single-shot error thresholds for these code families, and we also obtain the highest phase-flip noise threshold; see Table I. For our single-shot simulations, we use an independent and identically distributed noise model where each qubit experiences a phase-flip error with probability p, and each stabilizer measurement outcome is flipped with probability $q = p$. We investigate two decoding strategies: one where we use minimum-weight perfect matching (MWPM) for stage-1 decoding and belief propagation with ordered statistics

TABLE I. Comparison of the error thresholds of toric code decoders (results from this work are highlighted in bold). For phase-flip noise, BP+OSD outperforms all prior art, and approaches the theoretical upper bound given by mapping to a statistical mechanical model. In the single-shot regime, MWPM and BP+OSD outperforms the Sweep decoder (the theoretical maximum is unknown in this case).

Toric code decoder	Phase-flip threshold
Erasure mapping [30]	12.2%
Toom's rule [15]	14.5%
Sweep [17]	15.5%
Renormalization group [13]	17.2%
Neural network [31]	17.5%
BP+OSD	**21.55(1)%**
Statistical phase transition [32–36]	23.180(4)%
	Single-shot threshold
Sweep	1.7%
MWPM and BP+OSD	**2.90(2)%**

decoding (BP+OSD) for stage-2 decoding and another where we use BP+OSD for both decoding stages. Figure 1 shows the 3D surface code sustainable threshold fit, using the MWPM and BP+OSD decoding strategy. We find a comparable sustainable threshold for the 3D surface code using BP+OSD for both decoding stages, as shown in Table II.

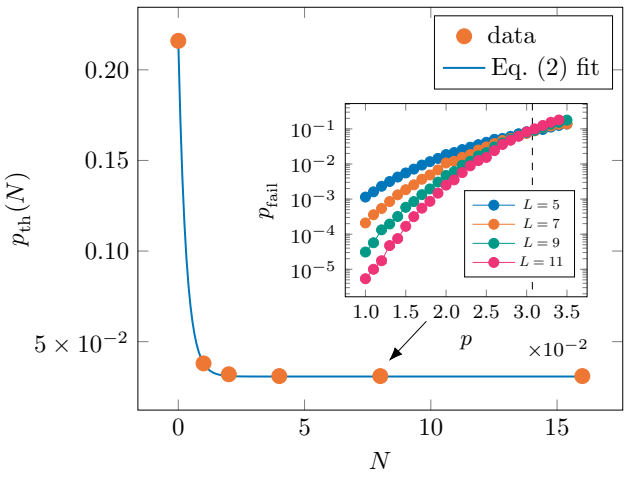


FIG. 1. Numerical estimate of the sustainable threshold of the 3D surface code for a two-stage decoder where we repair the syndrome using MWPM, and solve the corrected syndrome problem using BP+OSD. We plot the errors threshold $p_{\text{th}}(N)$ for different numbers of cycles, N. Using the ansatz in Eq. (2), we estimate the sustainable threshold to be $p_{\text{sus}} = 0.0308(4)$ with $\gamma = 3.23$. The inset shows a plot of the logical error rate, p_{fail}, against the phase-flip and measurement error rate, p, for $N = 8$. The error threshold $p_{\text{th}}(8)$ is the point at which the curves intersect (L is the code distance).

TABLE II. Sustainable thresholds for 3D toric and surface codes for different single-shot decoding strategies. For each entry in the table, we did an analogous simulation to that described in Fig. 1. The numbers in brackets are the standard errors.

Code	MWPM and BP+OSD	BP+OSD $\times 2$
Surface	3.08(4)%	2.90(1)%
Toric	2.90(2)%	2.78(2)%

There is an important difference in single-shot decoding for the 3D toric code when compared with the 3D surface code. Specifically, in the 3D toric code, the syndrome-repair stage of single-shot decoding can fail, producing a "syndrome" for which there is no corresponding error. We find that this failure mode substantially increases the logical error rate of the 3D toric code when compared with the 3D surface code (although the thresholds are very similar). We provide a novel decoding subroutine for dealing with these errors, which dramatically improves the performance of the 3D toric code. Furthermore, our subroutine is applicable to any single-shot LDPC code whose parity-check matrix is not full rank.

The advantage of using BP+OSD for stage-1 decoding is that, unlike MWPM, this decoder does not rely on the special structure of the looplike syndrome present in 3D toric and surface codes. We use BP+OSD for single-shot decoding of a family of nontopological 3D product codes, achieving sustainable thresholds that are comparable to those of the 3D surface code. This is the second example of single-shot decoding of nontopological codes, the first being the quantum expander codes considered in Ref. [27]. Whilst our single-shot threshold (2.7%) is slightly below the corresponding value observed in quantum expander codes (3%), our code family has other advantages over expander codes. Most importantly, the expansion properties of our code family are less severe, which implies that our code family would be easier to implement in architectures with geometrically constrained connectivity.

III. DEFINITIONS AND THEOREMS

In this section, we introduce the definition of *confinement* for a stabilizer code and exhibit a theoretical two-stage decoder, the *shadow decoder*, which we prove is single shot on confined codes against adversarial noise. We refer the reader to Appendix A to see how a variant of the shadow decoder can be used to prove that good families of codes with linear confinement have a single-shot threshold for local stochastic noise.

A stabilizer code encoding k logical qubits into n physical qubits can be described by its stabilizer group $\mathcal{S}$ and a syndrome map $\sigma(\cdot)$. The stabilizer group $\mathcal{S}$ is an Abelian subgroup of the Pauli group $\mathcal{P}_n$ on n qubits, which does not contain $-\mathbb{1}$ and has dimension $n-k$. The syndrome map is not unique: any generating set of the group $\mathcal{S}$ defines a

valid syndrome map for the code. If $\{s_1, \ldots, s_m\}$ is one such generating set, the associated function $\sigma(\cdot)$ maps a qubit operator $p \in \mathcal{P}_n$ into the binary vector $(\bar{s}_1, \ldots, \bar{s}_m)^T \in \mathbb{F}_2^m$, where $\bar{s}_i = 1$ if s_i anticommutes with p and 0 otherwise. Importantly, $\sigma(\cdot)$ is linear, meaning that $\sigma(p \cdot q) = \sigma(p) + \sigma(q)$ over $\mathbb{F}_2^m$. Because any Pauli operator $p \in \mathcal{P}_n$ can be factorized as the product of an X and a Z operator p_X and p_Z, we can identify it with a binary vector $\bar{p} = (\bar{p}_X, \bar{p}_Z)^T \in \mathbb{F}_2^{2n}$, where the ith entry of $\bar{p}_X/\bar{p}_Z$ is 1 if and only if p_X/p_Z acts nontrivially on the ith qubit. Given a Pauli operator p, its weight $|p|$ is the number of qubits on which its action is not the identity. Consider a stabilizer code with syndrome function $\sigma(\cdot)$, then the reduced weight of a Pauli operator $p \in \mathcal{P}_n$ on the physical qubits is

$$|p|^{\text{red}} := \min\{|q| : \sigma(q) = \sigma(p), q \in \mathcal{P}_n\}.$$

A stabilizer code is said to be distance d if d is the minimum weight of a Pauli operator not in $\mathcal{S}$ that has trivial syndrome. We refer to a code of length n, dimension k, and distance d as a $[[n, k, d]]$ code.

For a stabilizer code, we then have the following.

Definition 1 (Confinement)**.** *Let t be an integer and $f : \mathbb{Z} \to \mathbb{R}$ some increasing function with $f(0) = 0$. We say that a stabilizer code has (t, f) confinement if, for all errors e with $|e|^{\text{red}} \leq t$, it holds*

$$f(|\sigma(e)|) \geq |e|^{\text{red}}.$$

The reduced weight of operators is crucial in this definition to avoid making confinement fail by adding stabilizer operators to arbitrarily increase the weight of an otherwise low weight error.

Let us contrast this with Bombín's notion of confinement (Definition 16 of Ref. [10]) that has some similarities but allows only for linear functions of the form $f(x) = \kappa x$ for some constant κ. Many codes, including 3D product codes, have superlinear confinement functions, as such Bombín's definition does not encompass them. Moreover, the concept of confinement is closely related to soundness [18] but it is weaker and so able to encompass more families of codes, such as the expander codes [19–21], which are confined but not sound. Roughly speaking, a code has good confinement if small qubit errors produce small measurement syndromes; this differs from good soundness, which entails that small syndromes can be produced by small errors.

Formally, we define the following notion of good confinement for a family of stabilizer codes.

Definition 2 (Good confinement)**.** *Consider an infinite family of stabilizer codes. We say that the family has good confinement if each code in it has (t, f) confinement, where the following holds:*

1. *t grows with the length n of the code: $t \in \Omega(n^b)$ with $b > 0$;*
2. *and $f(\cdot)$ is monotonically increasing and independent of n.*

We say the code family has good X confinement if the above holds only for Pauli-Z errors.

Our main analytic result is that codes with good confinement are single shot.

Theorem 1. *Consider a family of $[[n, k, d]]$ quantum-LDPC codes with good confinement and growing distance $d \geq an^b$ with $a > 0$ and $b > 0$. This code family is single shot for the adversarial noise model. If the code family only has good X confinement then it is single shot with respect to Pauli-Z noise.*

We conjecture that the result of Theorem 1 can be extended to deal with local stochastic noise and used to show that LDPC codes with good confinement have a single-shot threshold. In this direction, we are able to prove that *linear* confinement is sufficient for codes to exhibit a single-shot threshold in the local stochastic noise setting.

Theorem 2. *Consider a family of $[[n, k, d]]$ quantum-LDPC codes with qubit degree at most $\omega - 1$ and good linear confinement such that $d \geq an^b$ with $a > 0$ and $b > 0$. This code family has a sustainable single-shot threshold for any local stochastic noise model. If the code family only has good X confinement then it has a sustainable single-shot threshold with respect to Pauli-Z noise.*

We further prove that 3D product codes have X confinement.

Theorem 3. *All 3D product codes have (t, f) X confinement, where t is equal to the Z distance of the code and $f(x) = x^3/4$ or better.*

Theorems 1 and 3 together motivate our numerical experiments reported in Sec. V.

We now proceed to prove Theorem 1. To this end, we use the shadow decoder that we introduce in Definition 3. The shadow decoder differs from previous single-shot two stage decoders (e.g., the MW single-shot decoder introduced in Definition 6 of [18]) in that it does not rely on metachecks on syndromes. If syndromes are protected by a classical code, as is the case for X syndromes of 3D product codes introduced in Sec. IV, then a single-shot decoding strategy could work as follows: (1) correct the measured syndrome whenever it does not satisfy all the constraints defined by the metacode; (2) find a recovery operator on qubits that has syndrome equal to the one found at point (1). The shadow decoder, instead, corrects

the syndrome both anytime it fails to satisfy all the constraints of the metacode and when it is generated by high weight errors. We do not describe how to implement it or make statements concerning the complexity of decoding. Our proof makes similar assumptions as the Kovalev-Pryadko quantum-LDPC threshold theorem [37] where they assumed a minimum-weight decoder without addressing implementation issues. Indeed, decoding for arbitrary LDPC codes is an nondeterministic polynomial-time complete (NP complete) problem that we do not expect to be efficiently solvable in full generality.

The building blocks of the shadow decoder are the t shadows of the code. A t shadow is a set in the syndrome space, which contains all the images of Pauli errors e on the physical qubits that have weight at most t. In other words, if we identify Pauli errors e on n qubits with $2n$-bit strings and we consider the metric space $\mathcal{M} = \mathbb{F}_2^{2n}$ endowed with the Hamming distance [i.e., the distance $d(\bar{e}_1, \bar{e}_2)$ between the vectors $\bar{e}_1$ and $\bar{e}_2$ corresponding to the Pauli errors e_1 and e_2, respectively, is defined as $d(\bar{e}_1, \bar{e}_2) = |e_1 + e_2|$] then the t shadow of the code is the image, via the syndrome function $\sigma(\cdot)$, of the ball of radius t centered at 0 in $\mathcal{M}$. Note that, because balls on $\mathcal{M}$ are not vector spaces, the shadows are not vector spaces either.

We are now ready to introduce the shadow decoder.

Definition 3 (Shadow decoder)**.** *The shadow decoder has variable parameter $t > 0$. Given an observed syndrome $\bar{s} = \sigma(e) + \bar{s}_e$ where $\bar{s}_e \in \mathbb{F}_2^m$ is the syndrome error, the shadow decoder of parameter t performs the following two steps:*

1. *Syndrome repair: find a binary vector $\bar{s}_r$ of minimum weight $|\bar{s}_r|$ such that $\bar{s} + \bar{s}_r$ belongs to the t shadow of the code, where*

$$t \text{ shadow} = \{\sigma(e) : |e| \leq t\}.$$

2. *Qubit decode: find e_r of minimum weight $|e_r|$ such that $\sigma(e_r) = \bar{s} + \bar{s}_r$.*

We call $r = e_r \cdot e$ the residual error.

A key result in proving Theorem 1 is the following promise on the performance of the shadow decoder: when a code has confinement, the weight of the residual error after one decoding cycle is bounded by a function of the weight of the syndrome error.

Lemma 1. *Consider a stabilizer code that has (t, f) confinement. Provided that the original error pattern e has $|e|^{\text{red}} \leq t/2$, on input of the observed syndrome $\bar{s} = \sigma(e) + \bar{s}_e$, the residual error r left by the shadow decoder of parameter $t/2$ satisfies*

$$|r|^{\text{red}} \leq f\,(2|\bar{s}_e|). \tag{3}$$

Assume $|e|^{\text{red}} \leq t/2$. By construction, e_r has minimum weight among all errors with syndrome $\sigma(e) + \bar{s}_e + \bar{s}_r \in t/2$ shadow of the code. In particular, $|e_r| \leq t/2$. By the triangular inequality for the weight function,

$$|e_r \cdot e|^{\text{red}} \leq |e_r|^{\text{red}} + |e|^{\text{red}} \leq t. \tag{4}$$

Therefore, we can apply the confinement property on the residual error $r = e_r \cdot e$:

$$f\left(|\sigma(e_r \cdot e)|\right) \geq |e_r \cdot e|^{\text{red}}. \tag{5}$$

By linearity of the syndrome function $\sigma(\cdot)$:

$$\sigma(e_r \cdot e) = \sigma(e_r) + \sigma(e) = \bar{s}_e + \bar{s}_r. \tag{6}$$

Note that the syndrome error $\bar{s}_e$ is a possible solution of the syndrome repair step of the shadow decoder, because by assumption $|e|^{\text{red}} \leq t/2$. Thus, $|\bar{s}_r| \leq |\bar{s}_e|$ and

$$|\bar{s}_e + \bar{s}_r| \leq |\bar{s}_e| + |\bar{s}_r| \leq 2|\bar{s}_e|. \tag{7}$$

Combining these and the monotonicity of f leads to the required bound on the residual error $r = e_r \cdot e$:

$$|e_r \cdot e|^{\text{red}} \leq f\left(2|\bar{s}_e|\right). \tag{8}$$

Theorem 1 follows directly from Lemma 1. In particular, Lemma 1 entails that a code with (t, f) confinement is robust against N cycles of qubit noise, noisy syndrome extraction, and single-shot decoding, as explained below.

At each cycle τ, we assume that a new error e^τ is introduced in the system and it is added to the residual error $r^{\tau-1}$. We assume that for the new physical error e^τ and the syndrome measurement error $\bar{s}_e^\tau$ the following hold:

$$|e^\tau|^{\text{red}} \leq t/4 \text{ and } f\,(2|\bar{s}_e^\tau|) \leq t/4. \tag{9}$$

We perform syndrome extraction on the state $\hat{e}^\tau = e^\tau \cdot r^{\tau-1}$. The noisy syndrome $\bar{s}^\tau = \sigma(\hat{e}^\tau) + \bar{s}_e^\tau$ is used as input for the shadow decoder of parameter $t/2$. The recovery operator e_r^τ found by the shadow decoder is then applied to the state and finally a new cycle starts where $r^\tau = e_r^\tau \cdot \hat{e}^\tau$. Let $r^0 = \mathbb{1}$, so that the initial state of the system is given by $\hat{e}^1 = e^1$, $\bar{s}^1 = \sigma(\hat{e}^1) + \bar{s}_e^1$. Note that if

$$|e^\tau|^{\text{red}}, |r^{\tau-1}|^{\text{red}} \leq t/4 \tag{10}$$

then $|\hat{e}^\tau|^{\text{red}} = |e^\tau \cdot r^{\tau-1}|^{\text{red}} \leq t/2$ and the hypotheses of Lemma 1. Combining this with the bound on the syndrome error, Eq. (9), we obtain

$$|r^\tau|^{\text{red}} \leq f\,(2|\bar{s}_e^\tau|) \leq \frac{t}{4}.$$

In conclusion, provided that the conditions on the physical and the measurement errors, Eq. (9), are satisfied for each

iteration up to $\tau - 1$, the residual error after the τ^{th} cycle is kept under control too.

Theorem 2 is proven in Appendix A. There, we introduce a novel notion of weight to describe local stochastic errors: the *closeness* weight. We then present the stochastic shadow decoder, a variant of the (adversarial) shadow decoder of Definition 3. Importantly, on confined codes, it keeps the closeness weight of the residual error under control over repeated correction cycles. Finally, the proof of Theorem 2 follows by combining these results with some classic percolation theory bounds.

The proof of Theorem 3 is very technical and is deferred to Appendix C. It is an adaption of the one of soundness for 4D codes given in Ref. [18], and it is reported in this paper for completeness. We remind the reader that, for our numerical studies on 3D product codes, we do not use the shadow decoder, but rather heuristics that perform well in practice. In particular, we use a two-stage decoder that exploits a metacheck structure on syndromes and attempts to repair the syndrome if and only if it does not pass all metachecks (see Sec. V C).

Our main motivation to introduce the concept of confinement and the shadow decoder was to find a feature of codes able to encompass all known examples of single-shot codes. Campbell [18] introduced the notion of soundness and showed that this property is a sufficient condition for codes to show single-shot properties in the adversarial setting. Nonetheless, Fawzi *et al.* [22] showed that expander codes have a single-shot threshold for local stochastic noise, even though they do not have the soundness property. As already said though, expander codes do have confinement. In Corollary 9 of Ref. [20] the authors prove that their confinement function is linear and call this property *robustness*. Confinement, in other words, fills the gap leaved by the concept of soundness. Furthermore, as Lemma 2 states, it is a requirement strictly weaker than soundness: any LDPC family of codes with good soundness has good confinement.

Definition 4 (Soundness [18])**.** *Let t be an integer and $f : \mathbb{Z} \to \mathbb{R}$ be a function with $f(0) = 0$. Given a stabilizer code with syndrome map $\sigma(\cdot)$ we say it is (t, f) sound if for all error sets e with $|\sigma(e)| \leq t$ it follows that $f(|\sigma(e)|) \geq |e|^{\text{red}}$.*

Definition 5 (Good soundness [18])**.** *Consider an infinite family of codes with syndrome maps $\sigma_n(\cdot)$. We say that the family has good soundness if each code in it is (t, f) sound where the following holds:*

1. *t grows with n such that $t \in \Omega(n^b)$ with $b > 0$;*
2. *and $f(\cdot)$ is monotonically increasing and independent of n.*

It follows easily from Campbell's definition of soundness and our definition of confinement that the former entails the latter.

Lemma 2. *Consider a LDPC code that is (t, f) sound with f increasing. If its qubit degree is at most ω, then it has $((t/\omega), f)$ confinement.*

If e is an error set with $|e|^{\text{red}} \leq t/\omega$, for its syndrome the following holds:

$$|\sigma(e)| \leq \frac{t}{\omega} \times \omega = t. \tag{11}$$

By soundness of the code,

$$f(|\sigma(e)|) \geq |e|^{\text{red}}. \tag{12}$$

In conclusion, confinement successfully describes general and inclusive properties related to single-shot error correction. More than that, good confinement is a requirement strictly weaker than good soundness as the following example illustrates. We consider the quantum expander code family of Ref. [20], which has the following four properties: (i) they have full-rank parity-check matrices; (ii) they have $(t, 3x)$ confinement, with $t \in \Omega(d)$; (iii) for every small error e with $|e| \leq 3$, we have $|\sigma(e)| > 1$ (see Appendix D for details).

By property (i), every syndrome is a valid syndrome and we can consider some weight-1 syndrome s. Assume to the contrary that there exists an $|e| \leq t$ giving the syndrome $s = \sigma(e)$, then by $(t, 3x)$ confinement

$$3|\sigma(e)| \geq |e|^{\text{red}},$$

and plugging in $|\sigma(e)| = 1$ gives

$$3 \geq |e|^{\text{red}}.$$

We know by property (iii) that this would entail $|\sigma(e)| > 1$, which leads to a contraction. Therefore, the assumption $|e| \leq t$ must be false and so $|e| > t$.

Therefore, if the code family had (τ, φ) soundness, then

$$\varphi(1) > t. \tag{13}$$

Because φ is the same across the whole family, and $t \in \Omega(d)$ increases proportionally to the code distance, by considering bigger codes if necessary, this leads to a contradiction. In other words, we show that this family of expander codes with good linear confinement cannot have good soundness.

The remainder of this paper is devoted to the study of the 3D product codes. We recall their construction in Sec. IV and we numerically assess their single-shot performance under local stochastic noise in Sec. V.

IV. CODE CONSTRUCTION

The identification of Pauli operators $p \in \mathcal{P}_n$ with binary vectors $\bar{p} = (\bar{p}_X, \bar{p}_Z) \in \mathbb{F}_2^{2n}$ is a group homomorphism (i.e., multiplication of Pauli operators corresponds to the sum of their vector representation in $\mathbb{F}_2^{2n}$) and because $\sigma(\cdot)$ is linear, syndrome measurement can be simulated via a matrix-vector product:

$$\sigma : \mathbb{F}_2^{2n} \longrightarrow \mathbb{F}_2^{m}$$
$$\begin{pmatrix} \bar{p}_X \\ \bar{p}_Z \end{pmatrix} \mapsto H \begin{pmatrix} \bar{p}_X \\ \bar{p}_Z \end{pmatrix},$$

where the vector $(\bar{p}_X, \bar{p}_Z)^T \in \mathbb{F}_2^{2n}$ represents a Pauli error on the physical qubits. Following the nomenclature from classical coding theory, we refer to the syndrome matrix H as parity-check matrix and we say that a code is LDPC when its parity check is low density.

A stabilizer code is a Calderbank-Shor Steane (CSS) code if its stabilizer group can be generated by the disjoint union of a set of X operators and a set of Z operators. In this case, its parity check is a block matrix:

$$H = \left(\begin{array}{c|c} 0 & H_X \\ \hline H_Z & 0 \end{array}\right), \tag{14}$$

where H_X has size $m_x \times n$ and H_Z has size $m_z \times n$ if the generating set of X stabilizers and Z stabilizers has cardinality m_x/m_z. Equation (14) entails that syndrome extraction can be performed separately for the X component and for the Z component. In fact, if a Pauli operator has vector representation $(\bar{p}_X, \bar{p}_Z)^T = (\bar{p}_X, 0)^T + (0, \bar{p}_Z)^T \in \mathbb{F}_2^{2n}$, then for its syndrome the following holds:

$$\begin{aligned} H \begin{pmatrix} \bar{p}_X \\ \bar{p}_Z \end{pmatrix} &= H \begin{pmatrix} \bar{p}_X \\ 0 \end{pmatrix} + H \begin{pmatrix} 0 \\ \bar{p}_Z \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ H_Z \bar{p}_X \end{pmatrix} + \begin{pmatrix} H_X \bar{p}_Z \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ \bar{s}_Z \end{pmatrix} + \begin{pmatrix} \bar{s}_X \\ 0 \end{pmatrix}, \end{aligned}$$

where $\bar{s}_Z \in \mathbb{F}_2^{m_z}$ and $\bar{s}_X \in \mathbb{F}_2^{m_x}$. In other words, it is possible to truncate these vectors without losing information and deal with X and Z operators separately. For this reason, we say that a CSS code is provided with two syndrome maps, which correspond to the two blocks or matrices H_X and H_Z, respectively. Accordingly, a CSS code will have a X distance and a Z distance and can be compactly refereed to as a $[[n, k, d_x, d_z]]$ code.

For our purposes, it is useful to describe CSS codes in terms of chain complexes. We first explain how a CSS code yields a chain complex and then how to define valid CSS codes starting from chain complexes. This last step ultimately allows us to use a standard method to iteratively build chain complexes, namely the product of complexes, to build interesting CSS codes (see, for instance, Ref. [38] for a comprehensive discussion on the subject).

Consider a CSS code $\mathcal{C}$ defined by the syndrome maps H_X and H_Z of size $m_x \times n$ and $m_z \times n$ respectively. The sequence of maps and vector spaces,

$$\mathbb{F}_2^{m_z} \xrightarrow{H_Z^T} \mathbb{F}_2^{n} \xrightarrow{H_X} \mathbb{F}_2^{m_x}, \tag{15}$$

contains all the information needed to define $\mathcal{C}$. In fact, the dimension of the vector space in the middle, n, is the length of the code, and the dimensions m_x and m_z of the external spaces are, respectively, the number of X and Z stabilizer generators. The logical dimension k of the code $\mathcal{C}$ equates to

$$\begin{aligned} k &= \dim(\ker H_X) - \dim(\operatorname{Im} H_Z^T) \\ &= \dim(\ker H_Z) - \dim(\operatorname{Im} H_X^T). \end{aligned}$$

We use $\ker H$ for the kernel of H, which is the set of all v such that $Hv = 0$. We use $\operatorname{Im} H$ for the image of H, which is the set of all vectors w that can be written as $w = Hv$ for some v. The distances d_x and d_z are given by

$$d_x = \min\{|v| \text{ such that } v \in \ker H_Z, v \notin \operatorname{Im}(H_X^T)\},$$
$$d_z = \min\{|v| \text{ such that } v \in \ker H_X, v \notin \operatorname{Im}(H_Z^T)\}.$$

Lastly, because rows of H_X and H_Z represent the support of X and Z stabilizer generators, respectively, we can also verify that X and Z stabilizers commute by assuring that the scalar product of each row of H_X and any row of H_Z (or equivalently each row of H_Z and any row of H_X) is 0 in $\mathbb{F}_2$. In fact, this is equivalent to verifying that the supports of any two X and Z stabilizer generators have even overlap and therefore that they represent commuting Pauli operators. In other words, for H_X and H_Z it holds that

$$H_X H_Z^T = 0 \in \mathbb{F}_2^{m_x \times m_z}. \tag{16}$$

To sum up, we completely define the CSS code $\mathcal{C}$ in terms of the sequence described by Eq. (15), which in homology theory is referred to as a *length-2 chain complex*. As we now detail the converse is also true, and any chain complex of length 2 or greater determines a CSS code.

A length ℓ chain complex $\mathfrak{C}$ is a collection of vector spaces $C^0, \ldots, C^\ell$ and linear maps $\delta_i : C^i \to C^{i+1}$:

$$C^0 \xrightarrow{\delta_0} C^1 \to \cdots \to C^i \xrightarrow{\delta_i} C^{i+1} \to \cdots \to C^\ell, \tag{$\mathfrak{C}$}$$

with the only constraint

$$\delta_i \delta_{i-1} = 0, \tag{17}$$

for $i = 1, \ldots, \ell - 1$. Whenever the spaces C^i are vector spaces on the binary field $\mathbb{F}_2$ and $\ell \geq 2$, we can define a

CSS code on $\mathfrak{C}$. To see how this is the case, let $0 < i \leq \ell - 1$. We define a CSS code $\mathcal{C}(\mathfrak{C}_i)$ on the chain complex $\mathfrak{C}$ by equating

$$H_Z = \delta_{i-1}^T, H_X = \delta_i. \tag{18}$$

Notice that the defining property of chain complexes, Eq. (17), entails that our choice, Eq. (18), for H_X and H_Z is valid: the stabilizer group generated by the X and Z operators supported on rows of H_X and H_Z respectively, is Abelian. Therefore, the unique CSS code $\mathcal{C}(\mathfrak{C}_i)$ associated to the syndrome maps given in Eq. (18) is well defined. Importantly, the parameters $[[n, k, d_x, d_z]]$ of $\mathcal{C}(\mathfrak{C}_i)$ all have a translation in the chain-complex language. Using such terminology, we say that the code has length $n = \dim(C_i)$. It is known that the number of logical qubits k is equal to the dimension of the ith homology group $\mathcal{H}_i$ or, equivalently, the dimension of the $(i-1)$th cohomology group $\mathcal{H}^*_{i+1}$, defined, respectively, as the quotient groups:

$$\mathcal{H}_i = {}^{\ker \delta_i}/_{\mathrm{Im}\, \delta_{i-1}},$$
$$\mathcal{H}^*_{i+1} = {}^{\ker \delta^T_{i-1}}/_{\mathrm{Im}\, \delta^T_i}.$$

The X distance equates the minimum weight of any nonzero vector in $\mathcal{H}^*_{i-1}$, while the Z distance is the minimum weight of any nonzero vector in $\mathcal{H}_i$. It is easy to verify that these definitions in terms of homology and cohomology are actually equivalent to the ones given above for the CSS code described by Eq. (15); we refer the interested reader to Ref. [39] for a detailed presentation of homology theory.

We introduce the homology language because it allows us to succinctly describe the class of 3D product codes studied here. By 3D product codes we refer to the CSS codes derived by the homological product of three length-1 chain complexes, as described in Ref. [40]. Given three classical linear codes with parity-check matrices δ_A, δ_B, and δ_C they define three length-1 chain complexes:

$$\delta_A : C_A^0 \longrightarrow C_A^1,$$
$$\delta_B : C_B^0 \longrightarrow C_B^1,$$
$$\delta_C : C_C^0 \longrightarrow C_C^1,$$

where $C_\ell^0 = \mathbb{F}_2^{n_\ell}$ and $C_\ell^1 = \mathbb{F}_2^{m_\ell}$ if δ_ℓ has size $m_\ell \times n_\ell$ for $\ell = A, B, C$. By using tensor product and direct sum of vector spaces and maps, we can combine these three chain complexes to build a bigger length-3 chain complex.

The tensor product is denoted by the symbol $\otimes$. Given two vector spaces A and B over a field $\mathbb{F}$, their tensor product is the vector space $A \otimes B$ generated by the formal sums $\sum a \otimes b$ where $a \in A$ and $b \in B$ and the operator $\otimes$ is bilinear, i.e., for any a_1, a_2, b_1, b_2 in A and B, respectively, it holds that

$$(a_1 + a_2) \otimes b_1 = a_1 \otimes b_1 + a_2 \otimes b_1,$$
$$a_1 \otimes (b_1 + b_2) = a_1 \otimes b_1 + a_1 \otimes b_2.$$

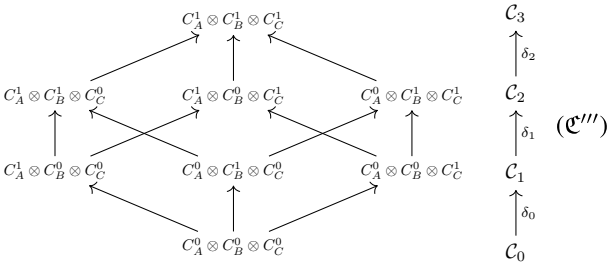
If $\alpha : A \to A'$ and $\beta : B \to B'$ are linear maps, their tensor product is defined as the linear map:

$$\alpha \otimes \beta : A \otimes B \longrightarrow A' \otimes B' :$$
$$a \otimes b \longmapsto \alpha(a) \otimes \beta(b).$$

It is of course possible to iterate this construction and define the tensor product of three (or more) spaces and maps, as we do now in order to obtain a length-3 chain complex $\mathfrak{C}'''$ from the *seed matrices* $\delta_A, \delta_B, \delta_C$. The chain complex $\mathfrak{C}'''$ is compactly described by the sequence of spaces and maps:

$$\mathcal{C}_0 \xrightarrow{\delta_0} \mathcal{C}_1 \xrightarrow{\delta_1} \mathcal{C}_2 \xrightarrow{\delta_2} \mathcal{C}_3,$$

which correspond to the tensor-product structure:



$(\mathfrak{C}''')$

where

$$\mathcal{C}_0 = C_A^0 \otimes C_B^0 \otimes C_C^0,$$
$$\mathcal{C}_1 = C_A^1 \otimes C_B^0 \otimes C_C^0 \oplus C_A^0 \otimes C_B^1 \otimes C_C^0 \oplus C_A^0 \otimes C_B^0 \otimes C_C^1,$$
$$\mathcal{C}_2 = C_A^1 \otimes C_B^1 \otimes C_C^0 \oplus C_A^1 \otimes C_B^0 \otimes C_C^1 \oplus C_A^0 \otimes C_B^1 \otimes C_C^1,$$
$$\mathcal{C}_3 = C_A^1 \otimes C_B^1 \otimes C_C^1,$$

and

$$\delta_0 = \begin{pmatrix} \delta_A \otimes \mathbb{1} \otimes \mathbb{1} \\ \mathbb{1} \otimes \delta_B \otimes \mathbb{1} \\ \mathbb{1} \otimes \mathbb{1} \otimes \delta_C \end{pmatrix},$$

$$\delta_1 = \begin{pmatrix} \mathbb{1}\otimes\delta_B\otimes\mathbb{1} & \delta_A\otimes\mathbb{1}\otimes\mathbb{1} & 0 \\ \mathbb{1}\otimes\mathbb{1}\otimes\delta_C & 0 & \delta_A\otimes\mathbb{1}\otimes\mathbb{1} \\ 0 & \mathbb{1}\otimes\mathbb{1}\otimes\delta_C & \mathbb{1}\otimes\delta_B\otimes\mathbb{1} \end{pmatrix},$$
$$\delta_2 = \begin{pmatrix} \mathbb{1}\otimes\mathbb{1}\otimes\delta_C & \mathbb{1}\otimes\delta_B\otimes\mathbb{1} & \delta_A\otimes\mathbb{1}\otimes\mathbb{1} \end{pmatrix}.$$

It is easy to verify that the chain complex ($\mathfrak{C}'''$) satisfies condition (17) for $i = 1, \ldots, 2$ and it is therefore well defined. As done above, we define a CSS code $\mathcal{C}(\delta_A, \delta_B, \delta_C)$ on ($\mathfrak{C}'''$) by equating

$$H_Z = \delta_0^T, \quad H_X = \delta_1.$$

We refer to the matrix $M = \delta_2$ as the metacheck matrix for the X stabilizers. Condition (18) entails $MH_X = 0$ and as a consequence we can think of the matrix M as a parity-check matrix on the X syndromes: any valid X syndrome satisfies the constraints defined by M.

Remarkably, the parameters of the code $\mathcal{C}(\delta_A, \delta_B, \delta_C)$ can be derived in terms of the properties of the seed matrices $\delta_A, \delta_B, \delta_C$. In fact, let $[n_\ell, k_\ell, \delta_\ell]/[n_\ell^T, k_\ell^T, d_\ell^T]$ be the parameters of the classical linear code with parity-check matrix $\delta_\ell/\delta_\ell^T$, $\ell = A, B, C$. As shown in Ref. [40], the chain complex ($\mathfrak{C}'''$) yields an $[[n, k, d_x, d_z]]$ code such that, if $k \neq 0$,

$$\begin{aligned} n &= n_a^T n_b n_c + n_a n_b^T n_c + n_a n_b n_c^T, \\ k &= k_a^T k_b k_c + k_a k_b^T k_c + k_a k_b k_c^T, \\ d_x &= \min\{d_a^T, d_b^T, d_c^T\}, \\ d_z &= \min\{d_b d_c, d_a d_c, d_a d_b\}. \end{aligned}$$

By convention, the distance of a code with dimension 0 is ∞. We define the single-shot distance d_{SS} [18] of the chain complex ($\mathfrak{C}'''$) as the minimum weight of a vector in $\mathcal{C}_2$ that satisfies all the constraints given by δ_2 (i.e., it belongs to the kernel of δ_2) but is not a valid X syndrome (i.e., it does not belong to the image of δ_1). In other words, d_{SS} is the minimum weight of a vector in the second homology group $\mathcal{H}_2 = \ker\delta_2/\operatorname{Im}\delta_1$ of the chain complex $\mathcal{C}$. Following Ref. [40] it is easy to verify that $d_{\mathrm{SS}} = \min\{d_a, d_b, d_c\}$ if $\mathcal{H}_2 \neq 0$ and ∞ otherwise.

It is important to note that, if the matrices δ_ℓ are LDPC, then their 3D product code is quantum LDPC. In fact, if δ_ℓ has column (row) of weight bounded by c_ℓ (r_ℓ), then δ_i has column and row weight bounded by c_i and r_i, respectively, where

i. $c_0 \leq c_a + c_b + c_c$ and $r_0 \leq \max\{r_a, r_b, r_c\}$;
ii. $c_1 \leq \max\{c_a + c_b, c_a + c_c, c_b + c_c\}$ and $r_1 \leq \max\{r_a + r_b, r_a + r_c, r_b + r_c\}$;
iii. $c_2 \leq \max\{c_a, c_b, c_c\}$ and $r_2 \leq r_a + r_b + r_c$.

A. On geometric locality

In addition to preserving the LDPC properties of the seed matrices, the 3D product yields local codes when qubits are placed on edges of a 3D cubic lattice. We defer the reader to Appendix B for a thorough discussion on the embedding of 3D product codes on a cubic lattice and we here present a loose summary.

Qubits of a 3D product code associated to the chain complex ($\mathfrak{C}'''$) are in bijection with basis elements of the space $\mathcal{C}_1$; since $\mathcal{C}_1$ is the direct sum of the three vector spaces $C_A^1\otimes C_B^0\otimes C_C^0$, $C_A^0\otimes C_B^1\otimes C_C^0$ and $C_A^0\otimes C_B^0\otimes C_C^1$ we introduce three different types of qubits: *transverse*, *vertical*, and *horizontal*. Qubit types naturally correspond to the three different orientations of edges on a cubic lattice, namely edges parallel to each of the three crystal planes. Referring to this particular display of qubits, the stabilizers of the code defined by Eq. ($\mathfrak{C}'''$) have support as follows:

1. X stabilizers have support on a two-dimensional (2D) cross of qubits of two types out of three, contained in one of the three crystal planes; the crossing is defined by a square face of a cube (see Fig. 7);
2. Z stabilizers have support on a 3D cross of qubits, with crossing defined by a vertex of a cube (see Fig. 8).

The cubic lattice considered can present some irregularities: in general it is a cubic lattice with some missing edges. Nonetheless, square faces and vertices are uniquely defined and they correspond to a stabilizer every time they contain at least one edge. More specifically, a square face identifies two perpendicular lines of edges i.e., qubits on a plane, which are the edges parallel to the boundary of the square face itself. The corresponding X stabilizer has support contained on those lines of edges i.e., qubits. Similarly, a vertex identifies three perpendicular lines of qubits and the corresponding Z stabilizer has support there contained. When combined with some locality properties of the seed matrices, this characteristic "cross shape" of the stabilizers support entails that 3D product codes are local on a cubic lattice (Proposition 1 in Appendix B). Here, by locality, we mean that for some positive integer ρ, the following hold:

1. any X-stabilizer generator has weight at most 2ρ with support contained in a 2D box of size $\rho \times \rho$;
2. any Z-stabilizer generator has weight at most 3ρ with support contained in a 3D box of size $\rho \times \rho \times \rho$.

Interestingly, it follows easily as a corollary of our locality proof that the 3D toric and surface codes are in fact 3D product codes. We now detail an explicit construction of the 3D toric and surface codes as 3D product codes and we refer the reader to Appendix B for further details.

The 3D toric code is the 3D product code obtained by choosing $\delta_A = \delta_B = \delta_C = \delta$ as seed matrices, where δ is the parity-check matrix of the repetition code. For instance, the 3D toric code with linear lattice size $L = 3$ is given by

$$\delta = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

In general, the 3D toric code of lattice size L, has parameters

$$[[3L^3, 3, L, L^2]]$$

and single-shot distance $d_{\mathrm{SS}} = L$.

The 3D surface code is obtained from this construction by choosing, for linear lattice size $L = 3$,

$$\delta_A = \delta_B = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

and

$$\delta_C = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}^T.$$

Therefore, for lattice size L, it has parameters

$$[[2L(L-1)^2 + L^3, 1, L, L^2]]$$

and single-shot distance $d_{\mathrm{SS}} = \infty$. Further details can be found in Appendix B.

It should be noted that the surface and toric codes are special instances of 3D product codes that have geometrically local stabilizers. This is beneficial, as it means they could be implemented on a quantum computer using only nearest-neighbor (in 3D) interactions between qubits. The disadvantage of the 3D surface and toric codes is that they have fixed dimension, encoding only one and three qubits, respectively. The 3D product construction can be used to obtain codes that are not geometrically local, but have improved encoding rates over the surface and toric codes. We refer to these codes as "nontopological" codes and investigate their decoding in Sec. V E.

V. NUMERICS

To assess the single-shot performance of the 3D product codes, we simulate the decoding of phase-flip (Z) errors. As 3D product codes are CSS codes, the relevant stabilizers are the X stabilizers. Let $\bar{e}_Z \in \mathbb{F}_2^n$ describe the support of a phase-flip error, i.e., $(\bar{e}_Z)_i = 1$ if qubit i has a phase-flip error. The syndrome, $\bar{s}_X$, of this error is then

$$\bar{s}_X = H_X \bar{e}_Z, \tag{19}$$

where $H_X \in \mathbb{F}_2^{m \times n}$ is the parity-check matrix of the X stabilizers of the code [see Eq. (14)].

Owing to the chain-complex structure of 3D product codes (outlined in Sec. IV) the syndromes $\bar{s}_X$ are themselves the codewords of a classical linear code with parity-check matrix M such that $M\bar{s}_X = 0$ for all $\bar{s}_X \in \mathrm{Im}(H_X)$. We refer to such a code on the syndromes as a metacode. The metacheck matrix can be used to detect and correct syndrome noise.

In a two-stage single-shot decoder, stage-1 decoding corrects the syndrome noise using M before stage-2 decoding corrects the data qubits. In general, decoding is an NP-complete problem that cannot be solved exactly in polynomial time. However, good heuristic techniques exist that allow approximate solutions to be efficiently computed. In this work, we use two such decoding methods: minimum-weight perfect matching and belief propagation plus ordered statistics decoding. Both MWPM and BP+OSD are algorithms that run over graphical models that encapsulate the structure of the code. We now briefly describe each decoder.

A. Minimum-weight perfect matching

The minimum-weight perfect-matching decoder is useful for codes in which chains of errors produce weight-2 syndromes. The method works by mapping the decoding problem to a graph where nodes represent the code's stabilizer generators and weighted edges represent error chains of different lengths. For a given pair of unsatisfied stabilizers, MWPM deduces the shortest error chain that could have caused it [41].

MWPM finds use for a variety of topological codes, most famously for the 2D surface and toric codes [2,42–45]. For 3D codes, MWPM is a suitable candidate for the syndrome-repair step referred to as stage-1 decoding. Specifically, the syndrome of a phase-flip error can be viewed as a collection of closed loops of edges in a simple cubic lattice [46] (with boundary conditions depending on the code). Measurement errors cause loops of syndrome to be broken, and the job of stage-1 decoding is to repair these broken syndromes. To obtain the corresponding matching problem, we create a complete graph whose vertices correspond to the break-points of the broken syndrome loops, with edge weights that are equal to the path lengths between the break points. We use the Blossom V [47] implementation of Edmonds's algorithm to solve this matching problem. The edges in the matching correspond to the syndrome-recovery operators.

B. Belief propagation + ordered statistics decoding

Belief propagation is an algorithm for performing inference on sparse graphs that finds widespread use in high-performance classical coding. Classical LDPC codes, for example, achieve performance close to the Shannon limit when decoded with BP [48]. In the context of quantum

coding, BP is useful for codes that do not produce pairs of syndromes and therefore cannot be decoded with MWPM.

The BP algorithm maps the decoding problem to a bipartite *factor graph* where the two node species represent data qubits and syndromes, respectively. Graph edges are drawn between the data and syndrome nodes according to the code's parity-check matrix. The factor graph is designed to provide a factorization of the probability distribution that describes the relationship between syndromes and errors. The BP algorithm proceeds by iteratively passing "beliefs" between data and syndrome nodes, at each step updating the probability that a data node is errored. The algorithm terminates once the probability distribution implies an error pattern that satisfies the inputted syndrome. For a full description of the BP algorithm we direct the reader to Ref. [49].

For quantum codes, the standard BP algorithm alone does not achieve good decoding performance due to the presence of degenerate errors. These cause "split beliefs" and prevent the algorithm from terminating. Various methods have been proposed for adapting BP decoding to quantum codes [27,50–54]. A particularly effective recent proposal involves combining BP with a postprocessing technique known as ordered statistics decoding [55]. The OSD step uses the probability distribution outputted by BP to select a low-weight recovery operator that satisfies the syndrome equation.

The BP+OSD algorithm was first applied to quantum expander codes by Panteleev and Kalachev [55]. Following this, Roffe *et al.* [56] demonstrated that the BP+OSD decoder applies more widely across a broad range of quantum-LDPC codes, including the 2D surface and toric codes. For this work, we use the software implementation of BP+OSD from Ref. [56], which can be downloaded from Ref. [57].

C. The two-stage single-shot decoding algorithm

Our simulations of the two-stage single-shot decoder employ two strategies. (1) MWPM and BP+OSD: stage-1 decoding is performed using MWPM and stage-2 decoding uses BP+OSD. (2) BP+OSD $\times$ 2: both stages are BP+OSD.

Algorithm 1 describes our methodology for the simulations of the two-stage single-shot decoder.

The 3D toric code has a failure mode that is not present in the 3D surface code. In such codes, syndromes $\bar{s}_X$ exist that satisfy all of the metachecks, $M\bar{s}_X = 0$, but are *invalid* syndromes, meaning that $\bar{s}_X$ does not belong to the image of H_X. In other words, $\bar{s}_X$ is invalid if there is no error vector $\bar{e}_Z \in \mathcal{C}_1$ with syndrome $\bar{s}_X$ but $\bar{s}_X$ is a codeword of the metacode.

Algorithm 1 SINGLE-SHOT ERROR CORRECTION

Input: Decoder 1, decoder 2, number of rounds N, error rate p, X parity check matrix H_X, metacheck matrix M, modified metacheck matrix M'
Output: Success or failure

$\bar{e}_Z \leftarrow 0$ ▷ Qubit error
$\bar{s}_X \leftarrow 0$ ▷ Syndrome
$\bar{m} \leftarrow 0$ ▷ Metasyndrome
for $j \leftarrow 1$ **to** N **do**
  Generate phase-flip error $\bar{e}'_Z$ according to error rate p
  $\bar{e}_Z \leftarrow \bar{e}_Z + \bar{e}'_Z$
  $\bar{s}_X \leftarrow H_X \bar{e}_Z$
  Generate syndrome error $\bar{s}'_X$ according to error rate p
  $\bar{s}_X \leftarrow \bar{s}_X + \bar{s}'_X$
  $\bar{m} \leftarrow M\bar{s}_X$
  Use decoder 1 to obtain syndrome recovery $\bar{r}_M$ such that $M\bar{r}_M = \bar{m}$
  $\bar{s}_X \leftarrow \bar{s}_X + \bar{r}_M$
  if $\bar{s}_X \notin \mathrm{Im}(H_X)$ **then** ▷ Failure-mode subroutine
    $\bar{s}_X \leftarrow \bar{s}_X + \bar{r}_M$
    Use decoder 1 to obtain valid $\bar{r}_M$ s.t. $M'\bar{r}_M = \bar{m}$
    $\bar{s}_X \leftarrow \bar{s}_X + \bar{r}_M$
  end if
  Use decoder 2 to obtain qubit recovery $\bar{r}_Z$ s.t. $H_X\bar{r}_Z = \bar{s}_X$
  $\bar{e}_Z \leftarrow \bar{e}_Z + \bar{r}_Z$
end for
Generate phase-flip error $\bar{e}'_Z$ according to error rate p
$\bar{e}_Z \leftarrow \bar{e}_Z + \bar{e}'_Z$
$\bar{s}_X \leftarrow H_X\bar{e}_Z$
Use decoder 2 to obtain qubit recovery $\bar{r}_Z$ s.t. $H_X\bar{r}_Z = \bar{s}_X$
$\bar{e}_Z \leftarrow \bar{e}_Z + \bar{r}_Z$
if $\bar{e}_Z$ is a stabiliser **then**
  return Success
end if
return Failure

More generally, referring to the chain complex ($\mathfrak{C}'''$):

$$\mathcal{C}_0 \xrightarrow[H_Z^T]{\delta_0} \mathcal{C}_1 \xrightarrow[H_X]{\delta_1} \mathcal{C}_2 \xrightarrow[M]{\delta_2} \mathcal{C}_3$$

we see that nonvalid syndromes do exist whenever $\mathrm{Im}\,\delta_1 \subsetneq \ker\delta_2$. In the homology language, we say that invalid syndromes are nontrivial elements of the second homology group:

$$\mathcal{H}_2 = \ker\delta_2/\,\mathrm{Im}\,\delta_1 = \ker M/\,\mathrm{Im}\,H_X.$$

If k_m is the dimension of $\mathcal{H}_2$, the set of invalid syndromes is a vector subspace of $\mathcal{C}_2$ of dimension k_m whose vectors can be written as $\bar{u} + H_X\bar{e}_Z$ where $\bar{u}$ is a representative of the equivalence class $[\bar{u}] \in \mathcal{H}_2$ and $\bar{e}_Z \in \mathcal{C}_1$. Thus, if F_M is a matrix whose columns are k_m vectors in $\mathcal{C}_2$ that generate $\mathcal{H}_2$ (meaning that they belong to k_m different equivalence classes in $\mathcal{H}_2$), we can write any invalid syndrome $\bar{s}_X$ as

$$\bar{s}_X = F_M\bar{v} + H_X\bar{e}_Z, \tag{20}$$

where $\bar{v} \in \mathbb{F}_2^{k_m}$ is nonzero if and only if $\bar{s}_X$ is invalid and $\bar{e}_Z$ is any error vector in $\mathcal{C}_1$.

By duality on $(\mathfrak{C}''')$, the second cohomolgy group:

$$\mathcal{H}_2^* = \ker \delta_1^T / \operatorname{Im} \delta_2^T = \ker H_X^T / \operatorname{Im} M^T,$$

has order k_m too. If L_M is a matrix whose k_m rows generates $\mathcal{H}_2^*$, then the product $\Pi = L_M F_M$ has full rank k_m because both L_M and F_M have full rank. Moreover, since the rows of L_M in particular belongs to $\ker H_X^T$, it holds $L_M H_X = 0$. Combining these two observations with Eq. (20) yields

$$\begin{aligned} L_M \bar{s}_X &= L_M F_M \bar{v} + L_M H_X \bar{e}_Z \\ &= \Pi \bar{v}, \end{aligned}$$

where $\Pi \bar{v} = 0$ if and only if $\bar{v} = 0$ because Π is full rank. In conclusion, we find that

$$L_M \bar{s}_X \neq 0$$

if and only if $\bar{s}_X$ is an invalid syndrome. As a consequence, we can assess whether a syndrome is invalid or not by calculating this product. The meaning of matrices L_M and F_M can be understood by looking at elements in $\mathcal{H}_2$ and $\mathcal{H}_2^*$ as logical operators of a CSS code defined on $(\mathfrak{C}''')$ with qubits in $\mathcal{C}_2$ (see Sec. IV). In this setting, the full-rank condition $\operatorname{rank}(L_M F_M) = k_m$ translates in the anticommuting relation between logical X and logical Z operators of the code.

In the 3D toric code, these invalid syndromes are loops of edges around one of the handles of the torus, and are equivalent to the logical operators of the metacode. It is therefore possible to check whether stage-1 decoding results in such a failure by checking whether the repaired syndrome anticommutes with a matrix L_M whose rows generate the relevant group of the logical operators of the metacode. When a metacode failure is encountered, a failure-mode subroutine (line 13 of Algorithm 1) is called that forces the repaired syndrome into the correct form. This subroutine involves using BP+OSD to decode a modified version of the metacheck matrix M' defined as follows:

$$M' = \begin{pmatrix} M \\ L_M \end{pmatrix}. \tag{21}$$

The additional constraints in the modified metacheck matrix ensure that the repaired syndrome is never an invalid syndrome. We use this subroutine only when we have an invalid syndrome (rather than all the time) as the L_M component causes M' to have higher maximum row and column weights than M, resulting in a reduction in BP decoding performance. Indeed, the rows of L_M must have weight lower bounded by the transpose distances of the seed codes [58]. Since the transpose distances of the seed codes also determine the Z distance of the quantum code (Sec. IV), we want these quantities to be growing with the length n of the code and therefore the matrix L_M is not, in general, LDPC.

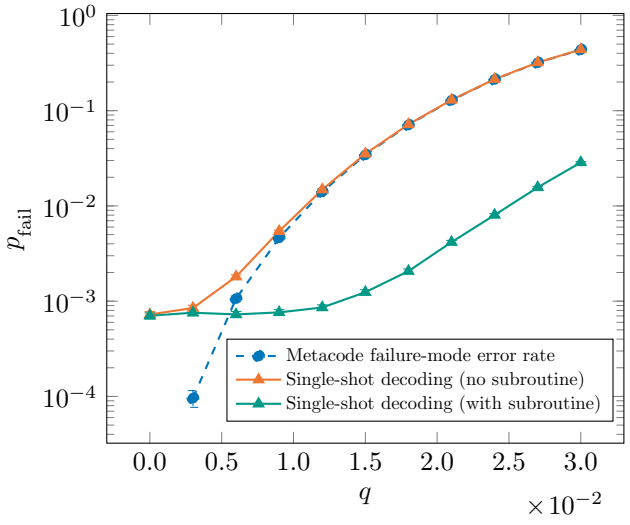


FIG. 2. Single-shot decoding of the 3D toric code with $L = 5$, with and without the metacode failure-mode subroutine. The failure rate p_{fail} is plotted against increasing values of the syndrome error rate q, whilst the phase-flip error rate is set to $p = 0.1$. Without the subroutine, the single-shot decoder rapidly converges to the failure-mode error rate (dotted line). For large values of q the subroutine improves the logical failure rate by over an order of magnitude. In this simulation, BP+OSD is used for both stage-1 and stage-2 decoding.

We find that whilst the failure-mode subroutine does not change the error threshold of the decoder, it does considerably reduce the logical error rate for all values of the lattice parameter L. This is illustrated for $L = 5$ in Fig. 2, which shows the single-shot logical error rate with and without the failure-mode subroutine. For large syndrome error rates, Fig. 2 shows the failure-mode subroutine improves decoding performance by over an order of magnitude.

D. 3D toric and surface codes

We estimate the sustainable threshold of the 3D toric and surface codes using our two decoding strategies. For code-capacity noise (i.e., perfect syndrome measurements), the syndrome-repair step is not required, so both decoding strategies are the same. For each code family, we observe a code capacity threshold of $p_{\text{th}} \approx 21.6\%$, as illustrated in Fig. 3. To obtain our threshold estimates, we use the standard critical exponent method [59] (see Appendix E for details). In the single-shot setting, we find similar performance for both our decoding strategies, as summarized in Table II. Our results compare favorably with the performance of other decoders, which we list in Table I. We

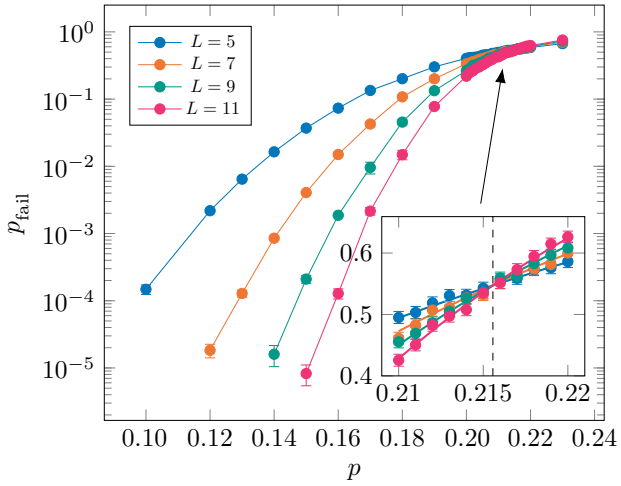


FIG. 3. Code-capacity threshold of the 3D toric code. We plot the logical error rate p_{fail} as a function of the phase-flip error rate p for codes with linear lattice size L. The inset shows an enlargement of the threshold region, where the lines show the threshold fit described in Appendix E. All data points have at least 25 failure events. The error bars show the 95% confidence intervals $p_{\text{fail}} = \hat{p}_{\text{fail}} \pm 1.96\sqrt{p_{\text{fail}}(1-p_{\text{fail}})/\eta}$, where $\eta \geq 10^4$ is the number of Monte Carlo trials.

obtain the highest reported code-capacity threshold and the highest reported single-shot threshold.

We remark that the sustainable threshold that we observe for the 3D toric code is very close to the threshold of MWPM for stringlike errors in the 3D toric code [60]. This implies that the performance of decoder 1 (the syndrome-repair step) is limiting the performance of the entire decoding procedure, as was suggested in Ref. [13]. Although the sustainable thresholds we observe for 3D surface codes are slightly higher than for 3D toric codes, the codes we consider are relatively small, which means that boundary effects may be having an impact on our sustainable threshold estimates.

We also investigate the suppression of the logical error rate below threshold in the 3D toric code, using MWPM and BP+OSD. We use the following ansatz for the logical error rate for values of $p < p_{\text{th}}$,

$$p_{\text{fail}}(L) \propto (p/p_{\text{th}})^{\alpha L^{\beta}}, \tag{22}$$

where α and β are parameters to be determined. The code distance of the 3D toric code for Z errors is L^2, so if the decoder is correcting errors up to this size, we would expect $\beta \approx 2$. Using the fitting procedure described in Appendix E, we estimate $\beta = 1.91(3)$ for $N = 0$ (code capacity) and $\beta = 1.15(3)$ for $N = 8$ (eight rounds of single-shot error correction). Therefore, for the (relatively small) codes that we consider, we find evidence that BP+OSD is correcting errors of weight up to the code distance. Viewed as an error-correction problem, the distance of the syndrome-repair step of decoding (i.e., the single-shot distance d_{SS}) is L, which is consistent with our observed value of β in the single-shot case. This provides further evidence that the bottleneck of our single-shot decoding procedure is the syndrome-repair step.

E. Nontopological codes

Up to this point, we explore the single-shot decoding performance of the 3D surface and toric codes. As explained in Sec. IV A, the 3D surface and toric codes are topological codes obtained by taking the 3D product of classical repetition codes. In this section, we extend our numerical analysis to nontopological codes constructed via the 3D product of random classical codes. Our motivation for investigating nontopological codes is twofold. First, by demonstrating that a random 3D product code has a sustainable threshold, we provide evidence for our conjecture that the results of Theorems 1 and 3 extend to the stochastic noise setting. Second, we provide evidence that BP+OSD is a general decoding method that applies beyond the class of well-studied topological 3D product codes.

Table III shows a family of nontopological codes $\mathcal{C}(\delta_A, \delta_B, \delta_C)$ obtained by taking the 3D product of a random code, δ_A, with two codes obtained from full-rank repetition codes, δ_B and δ_C. For this example, we choose δ_A to be a classical LDPC code constructed by randomly generating parity checks under the constraint that the parity-check matrix δ_A has column and row weights upper bounded by three and four, respectively. The specific advantage of nontopological codes is that they can have nonfixed dimension. For example, the $d = 6$ instance of $\mathcal{C}(\delta_A, \delta_B, \delta_C)$ encodes four logical qubits, whilst the $d = 10$ instance encodes ten logical qubits. In contrast, the dimension of the surface and toric codes are fixed at one and three for all code distances. The trade-off is that nontopological codes have nonlocal stabilizer checks, meaning they would have to be implemented on hardware with the ability to perform beyond-nearest-neighbor interactions between

TABLE III. A family of 3D product codes. The seed codes $\{\delta_A, \delta_B, \delta_C\}$ are set as follows: δ_A is a parity-check matrix of an $[n, k, d]$ LDPC code constructed under the constraint that the column and row weights of its parity-check matrix are upper bounded by three and four, respectively; δ_B is a $[L, 1, L]$ full-rank repetition code; δ_C is the transpose of a $[L, 1, L]$ full-rank repetition code. We denote by $\mathcal{C}(\delta_A, \delta_B, \delta_C)$ the 3D product code with seed matrices $\delta_A, \delta_B, \delta_C$. The code distance is set to ∞ for codes of dimension 0.

δ_A	δ_B	δ_C	$\mathcal{C}(\delta_A, \delta_B, \delta_C)$
$[16, 4, 6]$	$[6, 1, 6]$	$[6, 0, \infty]$	$[[1336, 4, 6]]$
$[20, 5, 8]$	$[8, 1, 8]$	$[8, 0, \infty]$	$[[3100, 5, 8]]$
$[24, 6, 10]$	$[10, 1, 10]$	$[10, 0, \infty]$	$[[5964, 6, 10]]$

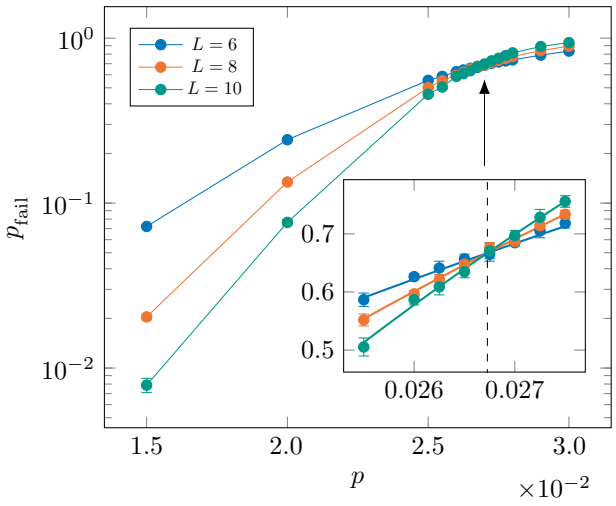


FIG. 4. Threshold plot for the family of nontopological 3D product codes listed in Table III after 16 rounds of single-shot error correction using the BP+OSD × 2 decoder. The simulation results suggest a threshold at 2.7%. The error bars show the 95% confidence intervals $p_{\text{fail}} = \hat{p}_{\text{fail}} \pm 1.96\sqrt{p_{\text{fail}}(1-p_{\text{fail}})/\eta}$, where η is the number of Monte Carlo trials.

qubits. An interesting feature of product-code constructions is that they can be used to interpolate between completely local topological codes and random quantum LDPC codes, as explored for the 2D setting in Ref. [56].

To numerically benchmark the single-shot performance of the nontopological code family listed in Table III, we simulate error correction under the two-stage decoder. The strategy we employ is BP+OSD × 2, for which both stage-1 and stage-2 decoding use the BP+OSD decoder. The MWPM and BP+OSD strategy used for the surface and toric codes would not work in this setting, as nontopological codes do not have the looplike metacheck syndromes required for MWPM decoding. The simulation results are summarized by Fig. 4, which shows a sustained threshold for the $\mathcal{C}(\delta_A, \delta_B, \delta_C)$ family of nontopological codes in the region of 2.7%. This result demonstrates that the BP+OSD decoding strategy can be used to decode new 3D product codes and achieve performance close to that of established codes such as the 3D surface and toric codes.

VI. CONCLUSION

In this paper, we investigate single-shot decoding of 3D product codes. We gave a formal definition of confinement in quantum codes and proved that all 3D product codes have confinement for Z errors. We also prove that confinement is sufficient for single-shot error correction against adversarial noise. This is a strengthening of the result of Campbell [18], who showed that a property called soundness is sufficient for single-shot error correction, in that soundness implies confinement but the converse is not true. Remarkably, there are important classes of codes, such as quantum expander codes, which have confinement but not soundness. Further to that, we prove that codes with linear confinement, and so expander codes, do have a single-shot threshold for local stochastic noise. The obvious open problem arising from our work is how to extend our findings for linear confinement to the superlinear case. Is confinement, in general, a sufficient condition for quantum-LDPC codes to exhibit a single-shot threshold? If not, what other requirements should a code satisfy to ensure the existence of a single-shot threshold?

We simulate single-shot error correction for a variety of 3D product codes, concentrating on 3D toric and surface codes. Using MWPM and BP+OSD, we achieve the best known code-capacity error threshold and sustainable single-shot error threshold for this code family (for phase-flip noise). Our results strongly suggest that the bottleneck of two-stage decoders is the first stage where the noisy syndrome is repaired. For the 3D toric code, the optimal threshold of the syndrome-repair step is 3.3% [33], whereas the optimal threshold of the entire decoding problem is 11.0% [35]. This implies that two-stage decoders can never achieve optimal performance in these codes, so perhaps other single-shot decoding methods ought to be investigated in future.

We also simulate single-shot error correction for a family of nontopological 3D product codes, using BP+OSD for both decoding steps. We achieve performance very close to that of the 3D toric and surface codes, which indicates that BP+OSD is a high-performance single-shot decoder. Furthermore, the versatility of BP+OSD means that we expect it to work as a single-shot decoder for general LDPC 3D product codes. We leave confirmation of this to future work, and we conjecture that BP+OSD will achieve good performance for other classes of quantum-LDPC codes such as topological fracton codes [61,62].

ACKNOWLEDGMENTS

This work is supported by the Engineering and Physical Sciences Research Council [Grants No. EP/P510270/1 (J.R.S.) and No. EP/M024261/1 (E.T.C. and J.R.)]. J.R. and E.T.C. were supported by the QCDA project (EP/R043825/1) which has received funding from the QuantERA ERA-NET Cofund in Quantum Technologies implemented within the European Union's Horizon 2020 Programme. J.R. also acknowledges funding from BMBF (RealistiQ) and the DFG (CRC 183). M.V. thanks Aleksander Kubica and Nikolas Breuckmann for illuminating discussions. We thank Rui Chao for comments on an early version of the paper. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Colleges and Universities. This research was

enabled in part by support provided by Compute Ontario [63] and Compute Canada [64]. This work was completed while E.T.C. was at the University of Sheffield.

APPENDIX A: LINEAR CONFINEMENT AND SINGLE-SHOT THRESHOLD

We present the stochastic shadow decoder, a variant of the (adversarial) shadow decoder described in Definition 3, and prove that it succeeds in correcting errors that have connected components that are sufficiently sparse and of bounded size, both on the syndrome and the qubits (Lemma 6). Theorem 2 will then follow from Lemma 6 on the performance of the stochastic shadow decoder: a family of codes with good linear confinement has a single-shot threshold under the local stochastic noise model.

This appendix is organized as follows. After fixing some graph-theory notation in Appendix A 1, we introduce a novel weight function for node sets in a graph, the *closeness* function, Appendix A 2. We prove that the closeness weight function preserves confinement and that the stochastic shadow decoder can be used on confined codes to keep the closeness of error under control (Appendix A 3). Crucially, the closeness weight function characterizes the structure of local stochastic errors better that the Hamming weight does, as some classic results in percolation theory show. We conclude, in Appendix A 4, by showing that a family of codes with good linear confinement has a sustainable single-shot threshold (Theorem 1). Our proof is built on the results in Refs. [22,65], where the authors prove that expander codes (which have linear confinement) have a single-shot threshold when decoded via the small-set flip decoder.

1. Notation and preliminaries

Given a stabilizer code on n qubits with stabilizer group $\mathcal{S} \subseteq \mathcal{P}_n$, we associate to it two graphs: $(\mathcal{G}_q, \sim_q)$, the qubit graph, and $(\mathcal{G}_s, \sim_s)$, the syndrome graph. The set of nodes are $\mathcal{G}_q$, the n qubits, and $\mathcal{G}_s$, a generating set of the stabilizer group $\mathcal{S}$ [66]. The adjacency relations $\sim_q$ and $\sim_s$ are defined as

$$q_1 \sim_q q_2 \Leftrightarrow \exists s \in \mathcal{G}_s \text{ such that } \{q_1, q_2\} \subseteq \mathrm{supp}(s),$$
$$s_1 \sim_s s_2 \Leftrightarrow \exists q \in \mathcal{G}_q \text{ such that } q \in \mathrm{supp}(s_1) \cap \mathrm{supp}(s_2);$$

where the support $\mathrm{supp}(s)$ of a Pauli operator s in $\mathcal{P}_n$ is the set of all the qubits on which its action is nontrivial. We use lowercase symbols for Pauli operators in $\mathcal{P}_n$ and the corresponding uppercase symbol to indicate its support, e.g., $E := \mathrm{supp}(e)$. We use the term error to refer interchangeably to a Pauli operator or its support, in particular given two Pauli operators e_1 and e_2 we use the symbol $+$ to indicate the support of the product operator $e_1 \cdot e_2$ [67], so that

$$E_1 + E_2 = \mathrm{supp}(e_1 \cdot e_2).$$

In this picture, the syndrome $\sigma(\cdot)$ maps the set of Pauli operators on n qubits $\mathcal{P}_n$ into the power set of $\mathcal{G}_s$,

$$\begin{aligned} \sigma : \mathcal{P}_n &\longrightarrow \mathscr{P}(\mathcal{G}_s) \\ e &\longrightarrow \{s \in \mathcal{G}_s : se = -es\}. \end{aligned}$$

We define the neighbor map Γ as

$$\begin{aligned} \Gamma : \mathcal{P}_n &\longrightarrow \mathscr{P}(\mathcal{G}_s) \\ e &\longrightarrow \{s \in \mathcal{G}_s : \mathrm{supp}(s) \cap E \neq \emptyset\}. \end{aligned}$$

With slight abuse of terminology, we call syndrome any element of $\mathscr{P}(\mathcal{G}_s)$, even when such a set does not belong to the image of σ. When referring to an error as a set E, it is always to be intended as corresponding to a fixed Pauli operator $e \in \mathcal{P}_n$ such that $E := \mathrm{supp}(e)$. We write interchangeably $\sigma(e)/\sigma(E)$ and $\Gamma(e)/\Gamma(E)$ to indicate the image, via the syndrome map and the neighbor map, respectively, of the Pauli error e.

Given two syndromes sets in $\mathcal{G}_s$ we use the symbol $+$ to indicate their symmetric difference. It is easy to verify that the map $\sigma(\cdot)$ preserves the $+$ operation (i.e., it is linear):

$$\sigma(e_1 \cdot e_2) = \sigma(E_1 + E_2) = \sigma(E_1) + \sigma(E_2).$$

Moreover, the image via σ of disjoint nonconnected sets is disjoint. In fact, if E_1, E_2 are two disjoint nonconnected sets in $\mathcal{G}_q$ and we suppose that their syndrome sets are not disjoint we find a contradiction. Let $\hat{s}$ be a stabilizer in $\sigma(E_1) \cap \sigma(E_2)$. By definition of σ, this entails that e_1 and e_2 both anticommute with $\hat{s}$, which is equivalent to saying that their supports have odd overlap with $\mathrm{supp}(\hat{s})$. In particular, there exists $q_i \in E_i$ such that $q_i \in \mathrm{supp}(\hat{s})$ and, by the definition of the adjacency relation $\sim_q$, $q_1 \in E_1$ and $q_2 \in E_2$ would be connected via $\hat{s}$, against the assumption. Note that, in general the image via the syndrome map $\sigma(\cdot)$ of a connected set needs not to be connected. However, the neighbor function $\Gamma(\cdot)$ maps connected sets into connected sets. We make use of these properties in Appendix A 3.

2. The closeness weight function

When errors are local stochastic it can be handy to use definitions of weight other than the cardinality and Hamming weight. For instance, the authors in Ref. [22] define the quantities of Definition 6 and study a related notion of percolation to understand the tolerance to errors of a given connected graph.

Definition 6 (α subsets, $\mathrm{MaxConn}_\alpha(E)$ [22]). *An α subset of a set $E \subseteq \mathcal{G}_q$ is a set K such that $|K \cap E| \geq \alpha |K|$. The maximum size of a connected α subset of E is denoted by* $\mathrm{MaxConn}_\alpha(E)$.

We here introduce a conceptual cousin to $\mathrm{MaxConn}_\alpha(E)$, the β closeness of an error set E, and prove that it is a well-defined weight function (see Lemma 3). We do not explicitly detail the relations between α subsets and closeness here. However, we implicitly use them, as our percolation results and ultimately the proof of Theorem 1 heavily rely on those relations and the proofs in Refs. [22,65].

Definition 7 (β closeness)**.** *Let $\mathcal{G}$ be a connected graph, i.e., a graph in which there exist a path between any two of its nodes. Given a subset E of nodes and a positive integer β, we define its β closeness as the quantity:*

$$\|E\|_\beta := \max\{|K \cap E| : K \text{ is connected}, \ |K| = \beta\}.$$

We call any connected subset of β nodes a β patch and any β patch K such that $|K \cap E| = \|E\|_\beta$ maximal patch for E.

Since we are interested in the β closeness of error sets on a qubit graph $\mathcal{G}_q$, it is natural to introduce the notion of reduced β closeness.

Definition 8. *Given a qubit error set $E \subseteq \mathcal{G}_q$, its reduced β closeness $\|E\|_\beta^{\mathrm{red}}$ is defined as*

$$\|E\|_\beta^{\mathrm{red}} := \min\{\|E + T\|_\beta : \ \sigma(E + T) = \sigma(E), \\ T = \mathrm{supp}(t) \text{ for some } t \in \mathcal{P}_n\}.$$

Crucially, we see in Lemma 5 that the closeness function preserves confinement. As a consequence, we can build a variant of the shadow decoder (Definition 9) that succeeds in correcting errors of small reduced closeness.

We now prove some basic properties of the β-closeness weight function $\|\cdot\|_\beta$ on a connected graph $\mathcal{G}$.

Lemma 3. *Let $\mathcal{G}$ be a connected graph and denote by $|\mathcal{G}|$ the number of its nodes. For any positive integer $\beta < |\mathcal{G}|$, the following hold:*

(i) $\|\cdot\|_\beta \leq |\cdot|$;
(ii) $\|\cdot\|_\beta \leq \beta$; *the equality holds if and only if the considered set of nodes has a connected component of size at least β; conversely, if $\|\cdot\|_\beta < \beta$ then the connected components of the set all have size less than β;*
(iii) it is positive: $\|E\|_\beta \geq 0$ *and equality holds if and only if $E = \emptyset$;*
(iv) it satisfies the triangle inequality: for any E_1, E_2, $\|E_1 \cup E_2\|_\beta \leq \|E_1\|_\beta + \|E_2\|_\beta$.
(v) it is monotonic: if $E_1 \subseteq E_2$ then $\|E_1\|_\beta \leq \|E_2\|_\beta$;

In the following, let $K \subseteq \mathcal{G}$ be a maximal β patch for E, i.e., $\|E\|_\beta = |K \cap E|$.

(i) $\|E\|_\beta = |K \cap E| \leq |E|$.

(ii) $|K \cap E| \leq |K| = \beta$. Equality holds if and only if $K \cap E = K \subseteq E$, which entails that E has a connected component of size at least β, since K is connected.

(iii) If E is nonempty then there exists at least one node $g \in E$. Since $\mathcal{G}$ is connected, for any integer $1 \leq \beta \leq |\mathcal{G}|$ there exists a β patch that contains g so that $\|E\|_\beta \geq 1$.

(iv) Let J be any β patch in $\mathcal{G}$. The following hold:

$$\begin{aligned} |J \cap (E_1 \cup E_2)| &= |(J \cap E_1)| + |(J \cap E_2)| + \\ &\quad - |J \cap (E_1 \cap E_2)| \\ &\leq |J \cap E_1| + |J \cap E_2| \\ &\leq \|E_1\|_\beta + \|E_2\|_\beta. \end{aligned}$$

Since this holds for any β patch, we obtain

$$\|E_1 \cup E_2\|_\beta \leq \|E_1\|_\beta + \|E_2\|_\beta.$$

(v) Let K_1, K_2 be maximal β patches for E_1 and E_2, respectively. Then

$$\begin{aligned} |K_1 \cap E_1| &\leq |K_1 \cap E_2| && \text{because } E_1 \subseteq E_2, \\ &\leq |K_2 \cap E_2| && \text{by maximality of } K_2, \end{aligned}$$

which yields $\|E_1\|_\beta \leq \|E_2\|_\beta$.

Lemma 2 below states that there exists a canonical form for maximal β patches of an error set E. Roughly speaking, a canonical β patch K will be made up of some entire connected components of E, plus at most one connected proper subset of a connected component of E, and some other nodes not in E (see Fig. 5). The existence of a canonical β patch is key in proving that the closeness function preserves confinement in the sense explained by Lemma 5.

Lemma 4 (Canonical β patch)**.** *For any error E on a qubit graph $\mathcal{G}$ there exists a maximal β patch T such that, for all but one connected component E_i of E, the following holds:*

$$\text{either } \ E_i \subseteq T \ \text{ or } \ E_i \cap T = \emptyset.$$

In other words, if $E_1, \ldots, E_m$ are the connected components of E, reordering if necessary, there exists an index ν such that

$$\begin{aligned} |T \cap E_i| &= |E_i| && \text{if } \ i < \nu, \\ |T \cap E_i| &\leq |E_i| && \text{if } \ i = \nu, \\ |T \cap E_i| &= 0 && \text{if } \ i > \nu. \end{aligned} \tag{A1}$$

We call any such T a canonical β patch for the set E.

Let J be any maximal β patch for E, i.e., J is connected, has size β and $|J \cap E| = \|E\|_\beta$. Starting from J we build a

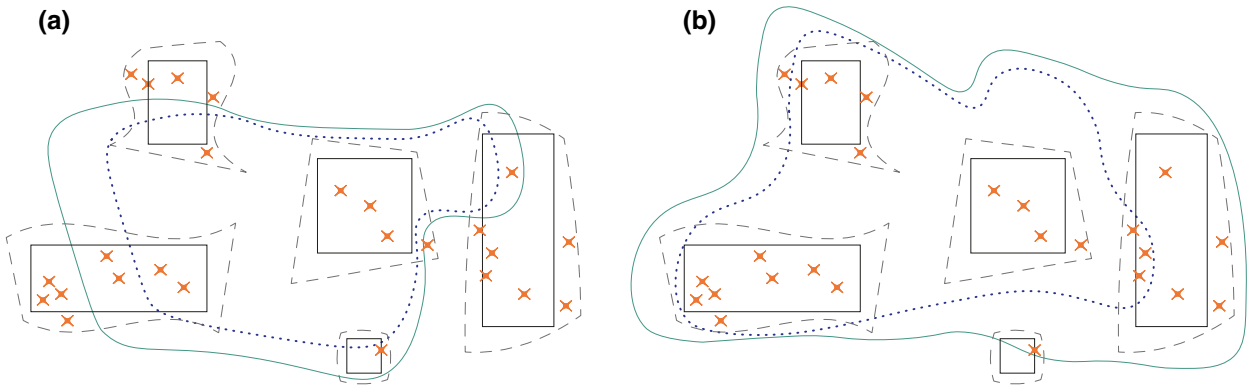


FIG. 5. Graphical representation of patches on a graph. To help the visualization we imagine the qubit graph and the syndrome graph to be superimposed. Black rectangles: connected components of the error $E_1, \ldots, E_5$. Dashed grey lines: neighbour sets $\Gamma(E_i)$ of the underlying rectangle and error component. Orange crosses: syndrome nodes in $\sigma(E_i)$. Dotted blue curve: t patches on the qubit graph. Green curve: ωt patches on the syndrome graph. In (a) the patches are generic while in (b) the dotted and error patch is a canonical patch for the error. The importance of the canonical form for a patch is highlighted in the differences between the patches in (a),(b). We observe how the crosses and syndrome nodes $\sigma(E_i)$ are scattered inside the dashed curve and neighbor set $\Gamma(E_i)$. For this reason, in order to group enough syndrome nodes inside a patch of bounded size, we need some care in the choice of the error nodes. When we include entire connected components of the error in a patch in $\mathcal{G}_q$, we are able to build a patch in $\mathcal{G}_s$, which includes entire neighbor sets and, as a consequence, all the corresponding syndrome nodes. In fact, even if we assume that the dotted blue and error patches in (a),(b) have the same size, when we enlarge them by a factor of ω to build the green and syndrome patch, we obtain dramatically different results. In (a) since the dotted and error patch contains several incomplete components, the corresponding green and syndrome patch contains incomplete portions of the dashed and neighbor sets $\Gamma(E_i)$. Hence, we have no guarantee on the number of crosses and syndrome nodes included in the patch. In (b) the dotted blue patch is a canonical patch for the error. We can see how the green and syndrome patch entirely contains the dashed and neighbor sets of all but one component of the error contained in the dotted blue and qubit patch. In this way we have the certainty to include in the green and syndrome patch a sufficient number of crosses and syndrome nodes to ensure confinement.

set T with the desired form. Write $J \cap E$ as disjoint union of connected sets:

$$J \cap E = J_1 \sqcup \cdots \sqcup J_\nu.$$

We call these J_i's patch-error components. Let $E_1 \ldots E_\mu$ be the connected components of the error E. We recall that a connected component E_i of E is a connected set, which is connected to no additional nodes in $E \setminus E_i$. We say that E_i is *incomplete* with respect to J if it has nontrivial overlap with J but it is not entirely contained in J, i.e.,

$$J \cap E_i \neq \emptyset \;\; \& \;\; E_i \not\subseteq J \Rightarrow |J \cap E_i| < |E_i|.$$

Note that it can be the case for two disjoint (but internally connected) error-patch components J_{i_1} and J_{i_2} to overlap with the same incomplete error component $E_{i'}$.

We consider a metagraph $\mathfrak{G}$ whose metanodes are connected sets in $\mathcal{G}$ and metaedges are paths in $\mathcal{G}$. Because the error-patch components are both internally and reciprocally connected in J, there exists a meta spanning tree $\mathcal{T} \subseteq \mathfrak{G}$ whose ν nodes $\mathcal{J}_i$ are the error-patch components J_i and whose metaedges ε_{ij} are formed by minimum length paths in $\mathcal{G}$ between the J_i's with nodes in $J \setminus E$. In the following we indicate with $\mathcal{T}, \mathcal{J}_i$ and ε_{ij} the metatree, its metanodes, and its metaedges and with T, J_i, and $e_{i,j}$ the corresponding sets of nodes in $\mathcal{G}$. Note that, by this identification, T has at most β nodes. We now show how to modify the metatree $\mathcal{T}$ so that the corresponding set of nodes T in $\mathcal{G}$ is canonical for E. We do this in two steps: the balancing and the enlargement step.

a. BALANCING

We show by induction on the number ν of the metanodes $\mathcal{J}_i$'s that it is possible to modify $\mathcal{T}$ so that the corresponding set of nodes $T \subseteq \mathcal{G}$ satisfies conditions (A1) on its overlap with the connected components of E.

$\nu = 1$: the thesis is trivially verified.

$\nu > 1$: if J is not canonical for E then E must have at least two incomplete components with respect to the patch J. Let $\mathcal{J}_\ell$ be a metaleaf of $\mathcal{T}$ and J_ℓ its corresponding subset of nodes in $\mathcal{G}$. We iteratively remove from T the nodes of J_ℓ, both preserving connectivity of T and the size of $T \cap E$.

For any node q_λ in J_ℓ, we choose a node q_χ such that the following holds:

i. q_χ belongs to some incomplete component of the error disjoint from J_ℓ: $q_\chi \in E_\chi$ and $E_\chi \cap J_\ell = \emptyset$;

ii. q_χ is a new node, i.e., it does not belong to J: $q_\chi \in \mathcal{G} \setminus J$;
iii. q_χ is connected to at least one node in some error-patch component other from J_ℓ: $q_\chi \sim_q q_{\chi'}$, $q_{\chi'} \in J_\chi$ for some $\chi \neq \ell$.

We remove q_λ from J and add q_χ to J, and thereby update T accordingly. This process terminates when either (a) we are not able to find such a new node q_χ or (b) there are no more nodes q_λ in J_ℓ.

Case (a) entails that E has at most one incomplete component with respect to T. In fact, if E had an incomplete component E_χ disjoint from J_ℓ such a node q_χ always exists. As a consequence, if we are not able to find a new error node to enlarge one of the error-patch components $J_\chi \neq J_\ell$ the only incomplete component of E must be the one relative to J_ℓ. The updated node set T has the desired property, provided that we had removed nodes q_λ from J_ℓ preserving connectivity (for instance, considering a spanning tree for nodes in J_ℓ and iteratively removing leaves). If case (b) is verified, we remove from $\mathcal{T}$ all the metaedges that are incident to $\mathcal{J}_\ell$. The updated metatree $\mathcal{T}$ derived from the updated set T has $\nu - 1$ metanodes. By the induction hypothesis, it can be modified to obtain the desired form.

In other words, we pick a metaleaf of $\mathcal{T}$ and we either remove part of its nodes [case (a)] or all of them [case (b)]. We preserve the quantity $|T \cap E|$ by adding new error nodes to some different error-patch component that overlaps with an incomplete component of the error E. By choosing a leaf, we are able to preserve the connectivity of $\mathcal{T}$ and thus the connectivity of the corresponding node sets T.

We iterate this procedure over metaleaves of $\mathcal{T}$ until the overlap of the corresponding set T in $\mathcal{G}$ and the error set E has the desired form.

b. ENLARGEMENT

By contradiction, we prove that it is possible to add nodes to the set T corresponding to the balanced metatree $\mathcal{T}$ so that it is connected, it has size exactly β and $|T \cap E| = \|E\|_\beta$. First note that during the balancing procedure, the number $|T \cap E|$ remains constant and the following holds:

$$|T \cap E| = \sum_{i=1}^{\nu} |J_i| = |J \cap E| = \|E\|_\beta.$$

Moreover, the initial tree is connected and the balancing procedure preserves connectivity. However, we only have an upper bound on the size of T. In fact, if $\mathcal{T}$ is the initial metatree and T is its corresponding subgraph in $\mathcal{G}$, it holds $T \subseteq J$ and therefore $|T| \leq \beta$. During the balancing step the size of T could decrease when we remove nodes of e_{ij}, belonging to a metaedge ε_{ij}. Thus, in general, after the balancing step for the weight of T holds:

$$|T| \leq \beta.$$

If $|T| = \beta$, then T is a β patch with maximum overlap with E and, by balancing, it is canonical. If $|T| < \beta$, then there must exist at least $\beta - \|E\|_\beta$ nodes in $\mathcal{G} \setminus (E \cup T)$ that are connected to T. In fact, a connected proper subset can always be enlarged in a connected graph. If the only way to enlarge T to a β patch were by adding nodes in E, then we would have found a β patch whose overlap with E has size greater than its β closeness, which contradicts the definition of $\|E\|_\beta$. In conclusion, any of such enlargements of the tree T is a canonical β patch for E.

3. Confinement and stochastic shadow decoder

Here, we first prove that the closeness function preserves confinement, as Lemma 5 states. Then, we present the stochastic shadow decoder (Definition 9) and prove, in Lemma 6, that it succeeds in correcting errors of small enough closeness. These findings, together with the percolation results of Appendix A 4, will yield the proof of the existence of a single-shot threshold for codes with linear confinement.

Lemma 5 (Closeness preserves confinement)**.** *Consider a code with qubit degree at most $\tilde{\omega}$ and (t,f) confinement, where f is convex. Then, for any error E with $\|E\|_t^{\mathrm{red}} \leq (t/2)$, the following holds:*

$$f\left(\|\sigma(E)\|_{\omega t}\right) \geq \|E\|_t^{\mathrm{red}},$$

where $\omega = \tilde{\omega} + 1$.

To ease the notation, let F be an error set such that $\sigma(E) = \sigma(F)$, $\|F\|_t = \|E\|_t^{\mathrm{red}}$. If $F_1, \ldots, F_\mu$ are the connected components of F, by Lemma 4 there exists a canonical patch K for F such that

$$\begin{aligned} |K \cap F_i| &= |F_i| && \text{if} \quad i < \nu, \\ |K \cap F_i| &\leq |F_i| && \text{if} \quad i = \nu, \\ |K \cap F_i| &= 0 && \text{if} \quad i > \nu. \end{aligned}$$

for some $\nu \leq \mu + 1$.

First, we prove that there exists a ωt patch J in the syndrome graph $\mathcal{G}_s$ such that it contains the syndrome of the connected components $F_1, \ldots, F_\nu$ of the error, which intersect the canonical patch K:

$$\bigsqcup_{i=1}^{\nu} [\sigma(F_i)] \subseteq J.$$

Then, we prove that such a patch J has overlap with $\sigma(F)$ of Hamming weight large enough to ensure confinement

with respect to the closeness function:

$$f\left(\|\sigma(F)\|_{\omega t}\right) \geq \|F\|_t.$$

We then find the desired bound on E using the initial assumptions $\sigma(F) = \sigma(E)$ and $\|F\|_t = \|E\|_t^{\mathrm{red}}$.

a. Existence of J

We build a ωt patch J on $\mathcal{G}_s$ as follows. We define J as the disjoint union of the (at most) $\tilde{\omega}|F_i|$ connected nodes $\Gamma(F_i)$:

$$J = \bigsqcup_{i=1}^{\nu} \Gamma(F_i).$$

Let π be the set of edges of a minimum length path in K that connects all its ν disjoint error components F_i. These edges correspond naturally to a set $\pi_s \subseteq \mathcal{G}_s$ if we associate to the edge (q_1, q_2), the corresponding stabilizer in $\mathcal{G}_s$, remembering that

$$q_1 \sim_q q_2 \Leftrightarrow \{q_1, q_2\} \subseteq \mathrm{supp}(s), \quad s \in \mathcal{G}_s.$$

Under this identification, importantly, adjacent edges are mapped into neighboring syndrome nodes. We add the set π_s to J. As a result, J is now connected. For the size of J, the following holds:

$$|J| \leq \tilde{\omega} \sum_{i=1}^{\nu} |F_i| + |\pi_s|.$$

By hypothesis, $t/2 \geq \|F\|_t = |K \cap F|$ and because K is canonical for F, i.e., $|K \cap F| \leq \sum_{i=1}^{\nu} |F_i|$, we have

$$\sum_{i=1}^{\nu-1} |F_i| \leq \frac{t}{2}.$$

Combining property (ii) of the closeness weight function and the assumption $\|F\|_t \leq (t/2)$, yields, for any i, and ν in particular,

$$|F_i| \leq \frac{t}{2}. \tag{A2}$$

Since π has edges in K, π_s has size at most $|K|$, i.e.,

$$|\pi_s| \leq t.$$

Adding up, we obtain

$$|J| \leq \omega t,$$

where $\omega = \tilde{\omega} + 1$. By enlarging J if necessary to include exactly ωt nodes, and remembering that by construction it is connected, we find that J is a ωt patch in $\mathcal{G}_s$, as desired.

b. Overlap of J with the error syndrome

Equation (A2) entails in particular that any connected error component $F_1, \ldots, F_\nu$ that has nontrivial overlap with the patch K, has size smaller than t and therefore it has confinement:

$$f\left(|\sigma(F_i)|\right) \geq |F_i|. \tag{A3}$$

Because σ maps disjoint sets of $\mathcal{G}_q$ in disjoint sets of $\mathcal{G}_s$,

$$\sigma\left(\bigsqcup_{i=1}^{\nu} F_i\right) = \bigsqcup_{i=1}^{\nu} \sigma(F_i)$$
$$\Rightarrow \quad \left|\sigma\left(\bigsqcup_{i=1}^{\nu} F_i\right)\right| = \sum_{i=1}^{\nu} |\sigma(F_i)|. \tag{A4}$$

Thus, applying f to each term of the summation of Eq. (A4) we have

$$\sum_{i=1}^{\nu} f\left(|\sigma(F_i)|\right) \geq \sum_{i=1}^{\nu} |F_i|. \tag{A5}$$

For the left-hand side of Eq. (A5), using convexity of f we obtain

$$f\left(\sum_{i=1}^{\nu} |\sigma(F_i)|\right) \geq \sum_{i=1}^{\nu} f\left(|\sigma(F_i)|\right),$$

for the right-hand side of Eq. (A5) instead, since K is canonical for F, it holds that

$$\sum_{i=1}^{\nu} |F_i| \geq |K \cap F|,$$

Combining these two bounds for Eq. (A5) yields

$$f\left(\sum_{i=1}^{\nu} |\sigma(F_i)|\right) \geq \|F\|_t. \tag{A6}$$

To obtain the thesis from Eq. (A6), we just need to substitute the Hamming weight on the left-hand side with the closeness weight $\|\cdot\|_{\omega t}$. By construction, for J it holds that

$$|J \cap \sigma(F)| \geq \sum_{i=1}^{\nu} |\sigma(F_i)|. \tag{A7}$$

Moreover, since J is a ωt patch:

$$\|\sigma(F)\|_{\omega t} \geq |J \cap \sigma(F)|. \tag{A8}$$

Using the monotonicity of f and combining Eqs. (A7), (A8), and (A6) yields

$$f\left(\|\sigma(F)\|_{\omega t}\right) \geq \|F\|_t.$$

c. Conclusion

Because F is an error set equivalent to E, i.e., $\sigma(F) = \sigma(E)$, such that $\|F\|_t = \|E\|_t^{\text{red}}$, we conclude

$$f\left(\|\sigma(E)\|_{\omega t}\right) \geq \|E\|_t^{\text{red}}$$

for $\omega = \tilde{\omega} + 1$.

Lemma 5 in particular entails that the closeness weight is in fact a sensible quantity to look at when dealing with errors on confined codes.

We now introduce the stochastic shadow decoder. The difference between this variant and the one previously presented (Definition 3) is on the weight functions used. While the standard and adversarial shadow decoder tries to minimize the Hamming weight of the residual error, the stochastic shadow decoder attempts to keep under control its closeness.

Definition 9 (Stochastic shadow decoder)**.** *The stochastic shadow decoder has variable parameters* $0 < \alpha \leq 1$*, and* $0 < \beta, \gamma \in \mathbb{Z}$*. Given an observed syndrome* $S = \sigma(E) + S_e$ *where* $S_e \subseteq \mathcal{G}_s$ *is the syndrome error, the stochastic shadow decoder of parameters* (α, β, γ) *performs the following two steps:*

1. *Syndrome repair: find* S_r *of minimum* γ *closeness* $\|S_r\|_\gamma$ *such that* $S + S_r$ *belongs to the* (α, β) shadow *of the code, where*

$$(\alpha, \beta) \text{ shadow} = \{\sigma(E) \text{ such that } \|E\|_\beta \leq \alpha\beta\}.$$

2. *Qubit decode: find* E_r *of minimum* β *closeness* $\|E_r\|_\beta$ *such that* $\sigma(E_r) = S + S_r$.

We call $R = E + E_r$ *the residual error.*

We have the following promise on the stochastic shadow decoder, which mirrors the results of Lemma 1 for the adversarial shadow decoder.

Lemma 6. *Consider a stabilizer code that has* (t, f) *confinement and qubit degree* $\leq \omega - 1$*. Provided that the original error pattern* E *has* $\|E\|_t^{\text{red}} \leq t/2$*, on input of the observed syndrome* $S = \sigma(E) + S_e$*, the residual error* R *left by the stochastic shadow decoder of parameter* $[(1/2), t, \omega t]$ *satisfies*

$$\|R\|_t^{\text{red}} \leq f\left(2\|S_e\|_{\omega t}\right). \tag{A9}$$

Thanks to Lemma 5, we know that the closeness function preserves confinement. The proof is then a straightforward adaption of the proof of Lemma 1, where the Hamming weight has to be substituted with $\|\cdot\|_t$ on error sets and $\|\cdot\|_{\omega t}$ on syndrome sets, respectively. We here briefly report the proof for completeness.

Assume $\|E\|_t^{\text{red}} \leq t/2$, and let E_r be the output of the qubit decode step. By construction, it has minimum t closeness among the errors with syndrome $S + S_r$, which belongs to the $[(1/2), t]$ shadow of the code. In particular, $\|E_r\|_t \leq (t/2)$. We recall that the $+$ operation between two error sets in $\mathcal{G}_q$ denotes the support of the product of the two corresponding Pauli operators and, as such, it holds that (see Appendix A 1)

$$E + E_r \subseteq E \cup E_r.$$

By the property of the closeness weight function, this entails

$$\|E + E_r\|_t \leq \|E \cup E_r\|_t \leq \|E\| + \|E_r\|_t.$$

The linearity of the syndrome function $\sigma(\cdot)$ yields

$$\sigma(E + E_r) = \sigma(E) + \sigma(E_r) = S_e + S_r.$$

Since S_e is a possible solution of the syndrome-repair step $\|S_r\|_{\omega t} \leq \|S_e\|_{\omega t}$ and so,

$$\begin{aligned}\|S_e + S_r\|_{\omega t} &\leq \|S_e\|_{\omega t} + \|S_r\|_{\omega t} \\ &\leq 2\|S_e\|_{\omega t}.\end{aligned}$$

Combining this and the monotonicity of f gives

$$\|E + E_r\|_t^{\text{red}} \leq f\left(2\|S_e\|_{\omega t}\right).$$

Lemma 5 tells us that the stochastic shadow decoder succeeds whenever the t closeness of the error is small enough. Importantly then, if we are able to bound the probability of the complement of this event, we could infer an upper bound on the failure probability of our decoder. This is the subject of the next section.

4. Percolation results and proof of Theorem 2

We consider error sets E on the qubit graph $\mathcal{G}_q$ and error sets S_e on the syndrome graph $\mathcal{G}_s$ and we assume that the probability of observing a particular error is at most exponential in its size. Formally, we use this error model.

Definition 10 (Local stochastic error)**.** *An error set* E *on a graph* $\mathcal{G}$ *is local stochastic of parameter* p *if, for all set of nodes* $G \subseteq \mathcal{G}$*, the following holds:*

$$\mathbb{P}(G \subseteq E) \leq p^{|G|}.$$

We then use some results in percolation theory, Lemmas 7 and 8 below, to understand the probability that errors of closeness linear in the patch size (i.e., $\|E\|_\beta = \alpha\beta$ for some $0 < \alpha \leq 1$) occur when the noise is local stochastic.

Lemma 7 (Corollary 28 of Ref. [22])**.** *Let $\mathcal{G}$ be a graph with vertex degree upper bounded by z. Then the number N_β of connected components of size β (β patches) satisfies*

$$N_\beta \leq |\mathcal{G}|\Phi^\beta,$$

where $\Phi = (z-1)\left(1+\frac{1}{z-2}\right)^{z-2}$.

Lemma 8. *Let $\mathcal{G}$ be a graph with vertex degree upper bounded by z. Let t be a positive integer and $0 < \alpha \leq 1$. Then there exists $p_{\text{th}} > 0$ such that, for local stochastic errors E of parameter $p < p_{\text{th}}$, we have*

$$\mathbb{P}(\|E\|_t \geq \alpha t) \leq \frac{|\mathcal{G}|}{1-2^{h(\alpha)/\alpha}p}\left(\frac{p}{p_{\text{th}}}\right)^{\alpha t}, \tag{A10}$$

where $h(\alpha) = \alpha\log_2(1/\alpha) + (1-\alpha)\log_2(1/1-\alpha)$ is the binary entropy function.

The proof is a straightforward adaption of the proof of Theorem 17 in Ref. [22]. By expanding the left-hand side of Eq. (A10), we find

$$\begin{aligned}\mathbb{P}(\|E\|_t \geq \alpha t) &= \mathbb{P}(\exists K\ t\text{ patch} : |K\cap E| \geq \alpha t)\\ &\leq \sum_{K\text{ is a }t\text{ patch}} \mathbb{P}(|K\cap E| \geq \alpha t).\end{aligned}$$

Observe that, for a t patch K,

$$\begin{aligned}\mathbb{P}(|K\cap E| \geq \alpha t) &\leq \sum_{m\geq\alpha t}\sum_{\substack{K'\subseteq K\\ |K'|=m}} \mathbb{P}\left(K\cap E = K'\right)\\ &\leq \sum_{m\geq\alpha t}\sum_{\substack{K'\subseteq K\\ |K'|=m}} \mathbb{P}\left(K'\subseteq E\right)\\ &\leq \sum_{m\geq\alpha t}\sum_{\substack{K'\subseteq K\\ |K'|=m}} p^m\\ &\leq \sum_{m\geq\alpha t}\binom{t}{m}p^m. \end{aligned}\tag{A11}$$

By Stirling's approximation [68],

$$\mathbb{P}\left(|K\cap E| \geq \alpha t\right) \leq \frac{(2^{h(\alpha)/\alpha}p)^{\alpha t}}{1-2^{h(\alpha)/\alpha}p}. \tag{A12}$$

Combining Eqs. (A11), (A12), and Lemma 7 yields

$$\begin{aligned}\mathbb{P}\left(\|E\|_t \geq \alpha t\right) &\leq N_t\frac{(2^{h(\alpha)/\alpha}p)^{\alpha t}}{1-2^{h(\alpha)/\alpha}p}\\ &\leq \frac{|\mathcal{G}|}{1-2^{h(\alpha)/\alpha}p}\cdot\left(\Phi 2^{h(\alpha)}p^\alpha\right)^t\end{aligned}$$

By imposing the right-hand side to decrease with t, we find

$$p \leq \left(\frac{1}{\Phi 2^{h(\alpha)}}\right)^{\frac{1}{\alpha}} := p_{\text{th}}.$$

And in conclusion,

$$\mathbb{P}(\|E\|_t \geq \alpha t) \leq \frac{|\mathcal{G}|}{1-2^{h(\alpha)/\alpha}p}\left(\frac{p}{p_{\text{th}}}\right)^{\alpha t}.$$

Finally, we are able to prove that there exists a threshold under which the probability of local stochastic errors to be noncorrectable via the stochastic shadow decoder becomes exponentially small in the system size, provided that the graphs $\mathcal{G}_s$ and $\mathcal{G}_q$ have bounded degree and linear confinement.

By Lemma 6, the residual error left by the stochastic shadow decoder on a (t,f)-confined code is kept under control provided that

$$\|E\|_t \leq \frac{t}{4} \text{ and } f\left(2\|S_e\|_{\omega t}\right) \leq \frac{t}{4}. \tag{A13}$$

If the function f is linear, i.e., $f(x) = \kappa x$ for some $\kappa > 0 \in \mathbb{Z}$, then conditions (A13) can be written as

$$\|E\|_t \leq \frac{t}{4} \text{ and } \|S_e\|_{\omega t} \leq \frac{t}{8\kappa}. \tag{A14}$$

If the qubit error E is local stochastic of parameter p and the syndrome error S_e is local stochastic of parameter q, thanks to Lemma 8, we obtain

$$\begin{aligned}\mathbb{P}(\|E\|_t \geq t/4) &\leq \frac{|\mathcal{G}_q|}{1-2^{4h(\frac{1}{4})}p}\left(\frac{p}{p_{\text{th}}}\right)^{\frac{t}{4}}\\ &:= C_q|\mathcal{G}_q|\left(\frac{p}{p_{\text{th}}}\right)^{\frac{t}{4}}\end{aligned}$$

and

$$\begin{aligned}\mathbb{P}\left(\|S_e\|_{\omega t} \geq \frac{t}{8\kappa}\right) &\leq \frac{|\mathcal{G}_s|}{1-2^{8\omega\kappa h(\frac{1}{8\omega\kappa})}q}\left(\frac{q}{q_{\text{th}}}\right)^{\frac{t}{8\omega\kappa}}\\ &:= C_s|\mathcal{G}_s|\left(\frac{q}{q_{\text{th}}}\right)^{\frac{t}{8\omega\kappa}},\end{aligned}$$

where

$$p_{\text{th}} := \left(\frac{1}{\Phi_q 2^{h(\frac{1}{4})}}\right)^4 \text{ and } q_{\text{th}} := \left(\frac{1}{\Phi_s 2^{h(\frac{1}{8\omega\kappa})}}\right)^{8\omega\kappa}.$$

As a result, by Lemma 6, the residual error is correctable except with probability at most

$$\max\left\{C_q|\mathcal{G}_q|\left(\frac{p}{p_{\text{th}}}\right)^{\frac{t}{4}},\ C_s|\mathcal{G}_s|\left(\frac{q}{q_{\text{th}}}\right)^{\frac{t}{8\omega\kappa}}\right\}.$$

In other words, for local stochastic noise of intensity $p \leq p_{\text{th}}$ on the qubits and $q \leq q_{\text{th}}$ on the syndrome, the stochastic shadow decoder has a sustainable single-shot threshold.

We conclude by noting that the assumption of linear confinement is key in the proof of Theorem 2. However, we speculate that the limitations of Theorem 2 are an artefact of our proof and *superlinear* confinement is a sufficient condition for a family of codes to exhibit a single-shot threshold. In fact, the existence of a threshold p_{th} and q_{th} relies on the bounds given in Lemma 8. There, it is fundamental that the relation between the chosen size of the patch and the size of the overlap with the error is linear [see Eqs. (A11) and (A12)]. In other words, Lemma 8 states that, if we take β patches on the error graph and γ patches on the syndrome graph, we are able to estimate the probability that errors have closeness less than $\alpha\beta$ and $\tilde{\alpha}\gamma$, respectively. By Eq. (A13), in order to bound the closeness of the residual error left by the stochastic shadow decoder, we need

$$\|S_e\|_\gamma \leq \frac{1}{2} f^{-1}(\alpha\beta).$$

As a consequence, combining this with the requirements of Lemma 8, entails

$$\gamma = \kappa\left(\frac{1}{2} f^{-1}(\alpha\beta)\right),$$

for some positive constant κ. In conclusion, building up on the results of Lemma 8, we either need to prove that confinement is preserved if we take on the syndrome graph patches of size linear in $f^{-1}(\alpha\beta)$ or, using our Lemma 5 without modification, that the function is itself linear.

APPENDIX B: QUBIT PLACEMENT ON A 3D LATTICE

Here we detail how to embed a 3D product code on a cubic lattice, where qubits sit on edges, Z stabilizers on vertices, X stabilizers on faces and metachecks on cells.

Let C^0 and C^1 be two vector spaces over $\mathbb{F}$ with basis $\mathcal{B}^0 = \{e_1^0, \ldots, e_n^0\}$ and $\mathcal{B}^1 = \{e_1^1, \ldots, e_m^1\}$, respectively. Given a linear map from C^0 into C^1, it can be represented as a $m \times n$ matrix δ over $\mathbb{F}$ such that its action on the elements of the basis $\mathcal{B}^0$ is given by

$$\begin{aligned} \delta : C^0 &\longrightarrow C^1 \\ e_i^0 &\longmapsto \delta e_i^0 = \sum_{\alpha=1}^{m} \delta_{\alpha,i} e_\alpha^1. \end{aligned} \tag{B1}$$

Expression (B1) allows us to write the support of vectors in $\delta(\mathcal{B}^0) = \left\{\delta e_i^0\right\}_i$ in a compact form. In fact, the support of δe_i^0 is the subset of $\mathcal{B}^1$:

$$\text{supp}(\delta e_i^0) = \left\{e_\alpha^1 : \delta_{\alpha,i} \neq 0\right\}_\alpha .$$

Since basis vectors are uniquely identified by their index, we can compactly write Eq. (B1) as a relation $*$ on the set of indices of the basis $\mathcal{B}^0$ and $\mathcal{B}^1$:

$$\begin{aligned} \{1, \ldots, n\} &\longrightarrow \{1, \ldots, m\} \\ \kappa &\longrightarrow \kappa^*, \end{aligned} \tag{B2}$$

where

$$\kappa^* = \{\eta : \delta_{\eta,\kappa} \neq 0\}_\eta.$$

Similarly, the transpose δ^T of the matrix δ induces a map from C^1 to C^0, which is defined on $\mathcal{B}^1$ as

$$\begin{aligned} \delta^T : C^1 &\longrightarrow C^0 \\ e_\alpha^1 &\longmapsto \delta^T e_\alpha^1 = \sum_{i=1}^{n} \delta_{\alpha,i} e_i^0, \end{aligned}$$

yields the relation on indices

$$\begin{aligned} \{1, \ldots, m\} &\longrightarrow \{1, \ldots, n\} \\ \eta &\longrightarrow \eta^*, \end{aligned} \tag{B2 T}$$

where

$$\eta^* = \{\kappa : \delta_{\eta,\kappa} \neq 0\}_\kappa.$$

Referring to the chain complex ($\mathfrak{C}'''$) described in Sec. IV, we choose bases $\mathcal{B}_\ell^\tau = \left\{e_\iota^{\ell\tau}\right\}_\iota$ of C_ℓ^τ for $\tau = 0, 1$ and $\ell = A, B, C$. We accordingly fix matrix representations of the maps δ_A, δ_B, and δ_C; with slight abuse of notation, we indicate with the same symbol the $m_\ell \times n_\ell$ matrix representation of a map and the map itself. We indicate with i, j, k indices of $\mathcal{B}_A^0, \mathcal{B}_B^0$, and $\mathcal{B}_C^0$, respectively, and with α, β, γ indices of $\mathcal{B}_A^1, \mathcal{B}_A^1, \mathcal{B}_C^1$. Since we deal with threefold tensor product spaces (e.g., $C_A^0 \otimes C_B^0 \otimes C_C^0$) we consider triplets (i, j, k) of valid indices; we indicate with (i^*, j, k) the set of indices $\{(\eta, j, k) : \eta \in i^*\}$, and similarly for any possible triplet combination of starred (ι^*) and nonstarred (ι) indices.

As illustrated in Sec. IV, when defining a CSS code on the chain complex ($\mathfrak{C}'''$), the following relations hold:

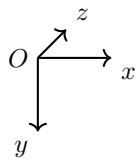
1. basis elements of $\mathcal{C}_0$ are in one-to-one correspondence with a generating set of Z stabilizers;
2. basis elements of the vector space $\mathcal{C}_1$ are in one-to-one correspondence with the qubits;
3. basis elements of the vector space $\mathcal{C}_2$ are in one-to-one correspondence with a generating set of X stabilizers;
4. basis elements of $\mathcal{C}_3$ are in one-to-one correspondence with a generating set of metachecks.

TABLE IV. Notation and correspondences between objects of the chain complex ($\mathfrak{C}'''$).

Object	Indexing	Basis vector
Qubits	(α,j,k)	$\left(e_\alpha^{A_1}\otimes e_j^{B_0}\otimes e_k^{C_0},\quad 0,\quad 0\right)$
	(i,β,k)	$\left(0,\quad e_i^{A_0}\otimes e_\beta^{B_1}\otimes e_k^{C_0},\quad 0\right)$
	(i,j,γ)	$\left(0,\quad 0,\quad e_i^{A_0}\otimes e_j^{B_0}\otimes e_\gamma^{C_1}\right)$
X stabilizers	(α,β,k)	$\delta_2^T\left(e_\alpha^{A_1}\otimes e_\beta^{B_1}\otimes e_k^{C_0},\quad 0,\quad 0\right)$
	(α,j,γ)	$\delta_2^T\left(0,\quad e_\alpha^{A_1}\otimes e_j^{B_0}\otimes e_\gamma^{C_1},\quad 0\right)$
	(i,β,γ)	$\delta_2^T\left(0,\quad 0,\quad e_i^{A_0}\otimes e_\beta^{B_1}\otimes e_\gamma^{C_1}\right)$
Z stabilizers	(i,j,k)	$\delta_1(e_i^{A_0}\otimes e_j^{B_0}\otimes e_k^{C_0})$
Metacheck	(α,β,γ)	$\delta_3^T(e_\alpha^{A_1}\otimes e_\beta^{B_1}\otimes e_\gamma^{C_1})$

Combining these with Eqs. (B2) and (B2 T), we obtain the relations reported in Table IV. More precisely, we choose as bases for the spaces $\mathcal{C}_3,\mathcal{C}_2,\mathcal{C}_1$, and $\mathcal{C}_0$ the product bases obtained by combining $\mathcal{B}^0_{\ell=A,B,C}$ and $\mathcal{B}^1_{\ell=A,B,C}$ and we index qubits, stabilizers, and metachecks on $\mathcal{C}(\delta_A,\delta_B,\delta_C)$ accordingly. Equivalently, basis vectors are labeled with consecutive integers so as to preserve the ordering induced by the bases.

We use the relations of Table IV to visualize the chain complex ($\mathfrak{C}'''$) on a 3D cubic lattice. In order to do so, we first fix a coordinate system



where O is the origin. Since basis vectors are labeled with integers (the ith basis vector corresponds to the integer i, and vice versa) we can build a 3D grid of points where any point corresponds to a basis vector of $\mathcal{C}_0,\mathcal{C}_1,\mathcal{C}_2$, or $\mathcal{C}_3$. More precisely we fix a set of valid coordinates for points in the grid:

1. integer coordinates $(z,y,x)=(i,j,k)$ for $i=1,\ldots,n_a, j=1,\ldots,n_b$, and $k=1,\ldots,n_c$;
2. half-integers coordinates $(z,y,x)=(\alpha+0.5,\ \beta+0.5,\ \gamma+0.5)$ for $\alpha=1,\ldots,m_a,\beta=1,\ldots,m_b$, and $\gamma=1,\ldots,m_c$;
3. the origin has coordinates $O=(1,1,1)$.

In this way, any point with valid coordinates uniquely identifies a basis vector (and therefore an object in the chain complex, see Table IV). For example,

TABLE V. Correspondence between qubits in $\mathcal{C}_1$ and edges of the lattice.

Qubit	Edge
Transverse qubits (α,j,k)	Edges parallel to the z axis Middle point: $(\alpha+0.5,j,k)$
Vertical qubits (i,β,k)	Edges parallel to the y axis Middle point: $(i,\beta+0.5,k)$
Horizontal qubits (i,j,γ)	Edges parallel to the x axis Middle point: $(i,j,\gamma+0.5)$

1. the point $(1,4,2)$ corresponds to the basis vector $(e_1^{A_0}\otimes e_4^{B_0}\otimes e_2^{C_0})\in\mathcal{C}_0$ (Z stabilizers);
2. the point $(1.5,4,2)$ corresponds to the basis vector $(e_1^{A_1}\otimes e_4^{B_0}\otimes e_2^{C_0},\ 0,\ 0)\in\mathcal{C}_1$ (qubits);
3. the point $(1.5,4,2.5)$ corresponds to the basis vector $(0,\ e_1^{A_1}\otimes e_4^{B_0}\otimes e_2^{C_1},\ 0)\in\mathcal{C}_2$ (X stabilizers);
4. the point $(1.5,4.5,2.5)$ corresponds to the basis vector $(e_1^{A_1}\otimes e_4^{B_1}\otimes e_2^{C_1})\in\mathcal{C}_3$ (metachecks).

We draw an edge for any qubit of the code defined on ($\mathfrak{C}'''$). Qubits are in one-to-one correspondence with basis element of $\mathcal{C}_1$ and therefore are of three different types: $(v,0,0)$, $(0,v,0)$, and $(0,0,v)$. Accordingly, we draw edges of three different types as detailed in Table V (see also Fig. 6). In other words, any point with two integer entries and one half-integer entry is the middle point of an edge of unit length, which corresponds to a qubit. In this way we obtain a cubic lattice with (possibly) some missing edges.

Points with two half-integer and one integer entries do not intersect any edge and sit in the center of a

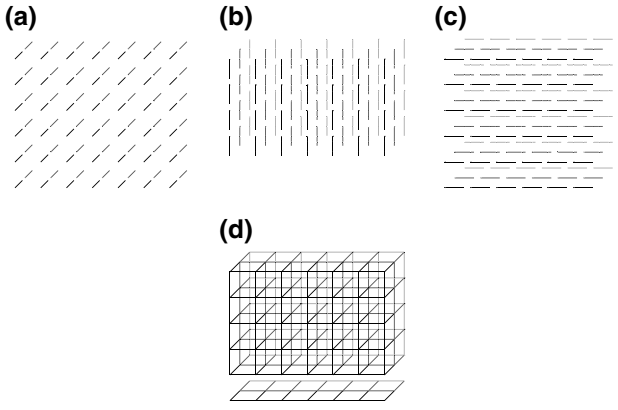


FIG. 6. Graphical representation of the cubic lattice associated to a 3D product code where the seed matrices $\delta_A,\delta_B,\delta_C$ have size 2×3, 4×6, and 6×7, respectively. In (a), (b), and (c) only transversal, vertical, and horizontal edges are depicted. In (d) we can see the complete lattice obtained by matching the origin $O=(1,1,1)$ of the three lattices of edges.

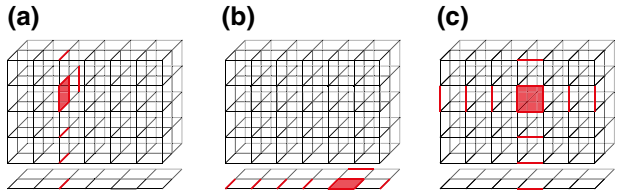


FIG. 7. X stabilizers on the lattice described in Fig. 6. (a) X stabilizer corresponding to the transversal-vertical square indexed by $(\alpha,\beta,k)=(1,2,3)$; its support is contained in the cross of transversal and vertical qubits (red edges) in the y-z plane $\{x=3\}$. The crossing has coordinates $(z,y,x)=(1.5,2.5,3)$ and sits in the center of the red square. (b) X stabilizer corresponding to the transversal-horizontal square indexed by $(\alpha,j,\gamma)=(1,6,5)$; its support is contained in the cross of transversal and vertical qubits (red edges) in the x-z plane $\{y=6\}$. The crossing has coordinates $(z,y,x)=(1.5,6,5.5)$ and sits in the center of the red square. (c) X stabilizer corresponding to the vertical-horizontal square indexed by $(i,\beta,\gamma)=(1,4,2)$; its support is contained in the cross of transversal and vertical qubits (red edges) in the x-y plane $\{z=1\}$. The crossing has coordinates $(z,y,x)=(1,2.5,4.5)$ and sits in the center of the red square.

(possibly incomplete) square face. These points correspond to X stabilizers, which we therefore identify with faces. Given a triplet corresponding to one of such a point, the associated X stabilizer has support contained in the set of edges, which are parallel to the edges of the square, forming a cross in a plane. X stabilizers, like qubits, are of three different types, being in one-to-one correspondence with basis elements of $\mathcal{C}_2$. Namely, each X stabilizer in $\mathcal{C}_2$ has support in two out of three types of qubits: transverse-vertical, transverse-horizontal, or vertical-horizontal (see Table VI and Fig. 7).

Points with integer coordinates are associated to Z stabilizers; these are points where endpoints of edges intersect.

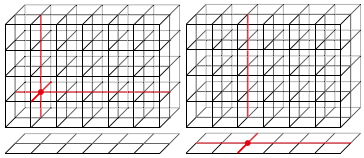


FIG. 8. Z stabilizers on the lattice described in Fig. 6. (a) Z stabilizer corresponding to the vertex indexed by $(i,j,k)=(2,4,2)$; its support is contained in the cross of qubits highlighted as red edges in the picture. The crossing has coordinates $(z,y,x)=(2,4,2)$ (red circle). (b) Z stabilizer corresponding to the vertex indexed by $(i,j,k)=(3,6,2)$; its support is contained in the cross of qubits highlighted as red edges in the picture. The crossing has coordinates $(z,y,x)=(3,6,2)$ (red circle).

The Z stabilizer corresponding to (i,j,k) has support on a 3D cross of edges and qubits centered in $(z,y,x)=(i,j,k)$ (see Table VI and Fig. 8).

Points with half-integer coordinates sit in the center of a cube. To any such cube is associated a metacheck in $\mathcal{C}_3$. Metachecks have support on a 3D cross of faces and X stabilizers parallel to the faces of the cube they are associated to (see Table VI).

1. On geometric locality

One interesting feature of the embedding of 3D product codes on a cubic lattice is that it preserves some locality properties of the seed matrices δ_A,δ_B, and δ_C. Thus, if we were able to place qubits on a 3D cubic lattice we could use the 3D homological product to build LDPC codes with nearest-neighbor interactions.

Let δ be an $m\times n$ matrix with row and column indices $\alpha\in\{1,\ldots,m\}$ and $i\in\{1,\ldots,n\}$, respectively, and let $\nu=\max\{m,n\}$. We say that δ is *geometrically ρ local on a*

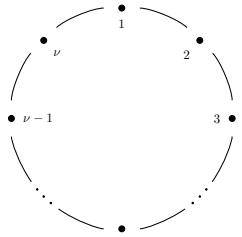
TABLE VI. Correspondence between operators of the chain complex ($\mathfrak{C}'''$), their type as geometric objects on the lattice, and their support. Note that the support of X and Z stabilizers is a set of qubits and edges while the support of metachecks is a set of X stabilizers and faces.

Operator	Type	Support
X stabilizers	Transverse-vertical square (α,β,k)	Transverse qubits: (α,β^*,k) Vertical qubits: (α^*,β,k)
	Transverse-horizontal square (α,j,γ)	Transverse qubits: (α,j,γ^*) Horizontal qubits: (α^*,j,γ)
	Vertical-horizontal square (i,β,γ)	Vertical qubits: (i,β,γ^*) Horizontal qubits: (i,β^*,γ)
Z stabilizers	(i,j,k)	Transverse qubits: (i^*,j,k) Vertical qubits: (i,j^*,k) Horizontal qubits: (i,j,k^*)
Metachecks	(α,β,γ)	Transverse-vertical faces: (α,β,γ^*) Transverse-horizontal faces: (α,β^*,γ) Vertical-horizontal faces: (α^*,β,γ)

torus if for any row and any column index

$$\alpha^* \subseteq U_{\rho,\nu}(\alpha) \quad \text{and} \quad i^* \subseteq U_{\rho,\nu}(i), \tag{B3}$$

where $U_{\rho,\nu}(\zeta)$ is any set of ρ consecutive integers modulo ν, which contains ζ. In particular, we require the αth rows to have support on columns with index that is *close* to the integer α, and similar for columns. The reason for this choice will be clear when we prove Proposition 1. Briefly, conditions (B3) say that δ is geometrically ρ local on a torus if the following hold: (1) any of its rows has support on a bounded box of ρ columns, and the box for row $\alpha+1$ is a right shift of the box for row α; (2) any of its columns has support on a bounded box of ρ rows, and the box for column $i+1$ is a downward shift of the box for column i. In particular, if we associate row and column indices with integer points on a circle of ν points:



locality means that any set α^*/i^* is contained in a closed interval on the circle such that (i) it has length at most ρ and (ii) it contains the point α/i. For instance, the degenerate parity-check matrix of the repetition code:

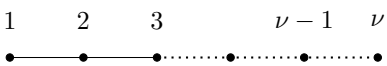
$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

is ρ local for $\rho = 2$.

A closely related notion of locality on a torus is geometric locality in Euclidean space. We say that an $m \times n$ matrix is *geometrically ρ local in Euclidean space* if for any row and column index

$$\alpha^* \subseteq U_{\rho}(\alpha) \quad \text{and} \quad i^* \subseteq U_{\rho}(i), \tag{B4}$$

where $U_{\rho}(\zeta)$ is any set of ρ consecutive integer in $[1,\ldots,\nu]$, $\nu = \max\{m,n\}$, which contains ζ. In this case we can graphically picture locality by associating row and column indices with integer points on a line of ν points:



A matrix is local if any set α^*/i^* is contained in a closed interval on the line such that (i) it has length at most ρ and (ii) it contains the point α/i. For example, the full-rank parity-check matrix of the repetition code:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

is 2 local.

Geometric locality also applies to codes other than the repetition code. For instance, the matrix

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix},$$

obtained via the edge augmentation procedure presented in Ref. [56] is 7 local on a torus. We remark that geometric locality is a property of matrices. For example, the matrix with the same row as H but different ordering $\{1,2,3,4,7,5,6\}$, is geometrically 5 local on a torus.

In general, geometric locality is a relaxation of the locality property of the repetition code, which only allows for interactions between pairs of nearest bits. Importantly, as Proposition 1 states, it is preserved by the 3D product construction. For this reason, geometrically local classical codes, combined with the 3D product construction, could be good candidates in the quest to quantum local codes beyond the toric and the surface codes.

The remainder of this appendix is organized as follows. We first state Proposition 1 and prove that geometric locality is preserved by the 3D product construction. We conclude by observing how this proof provides an explicit identification of the 3D toric and surface codes as 3D product codes.

To ease the notation, in the following we shortly refer to codes as geometrically local, dropping the specification on a torus or in Euclidean space. When considering qubits on a cubic lattice, the lattice would be on a torus or in Euclidean space depending on the definition of locality that applies to the seed matrices.

Proposition 1. *Consider the 3D product code obtained from three seed matrices geometrically ρ local. If its qubits are displayed on the edges of a cubic lattice as detailed in Appendix B, then it is geometrically ρ local in the following sense:*

1. *any X-stabilizer generator has weight at most 2ρ with support contained in a 2D box of size $\rho \times \rho$;*
2. *any Z-stabilizer generator has weight at most 3ρ with support contained in a 3D box of size $\rho \times \rho \times \rho$.*

We prove the condition on the Z stabilizers, the proof for the X stabilizer being similar.

Let S_z be a Z-stabilizer generator. As reported in Tables IV and VI, it is the image of a basis vector $(e_i^{A_0}\otimes e_j^{B_0}\otimes e_k^{C_0}) \in \mathcal{C}_0$ via the map δ_1 and it corresponds to the point on the lattice of integers coordinates (i,j,k). By exploiting the choice of the basis for the spaces $\mathcal{C}_0$ and $\mathcal{C}_1$ and some linear algebra:

$$\begin{aligned}\delta_1(e_i^{A_0}\otimes e_j^{B_0}\otimes e_k^{C_0}) = &\sum_{\alpha\in i^*}(e_\alpha^{A_1}\otimes e_j^{B_0}\otimes e_k^{C_0},\ 0,\ 0)\\ &+\sum_{\beta\in j^*}(0,\ e_i^{A_0}\otimes e_\beta^{B_1}\otimes e_k^{C_0},\ 0)\\ &+\sum_{\gamma\in k^*}(0,\ 0,\ e_i^{A_0}\otimes e_j^{B_0}\otimes e_\gamma^{C_1}).\end{aligned}$$

Again using Table IV, the set of indices, which corresponds to this sum of basis vectors of $\mathcal{C}_1$, can be written as

$$\begin{aligned}\text{indices}(S_z) = &\{(\alpha,j,k) : \alpha \in i^*\}\\ &\cup \{(i,\beta,k) : \beta \in j^*\}\\ &\cup \{(i,j,\gamma) : \gamma \in k^*\}.\end{aligned}$$

Following the nomenclature of qubits as traversal, vertical, and horizontal, we see that the three components of the support of S_z given above respect this division and therefore we can write

$$\text{indices}(S_z) = \text{indices}(S_z)_t \cup \text{indices}(S_z)_v \cup \text{indices}(S_z)_h.$$

Using (B4) [or (B3)], we see that the sets $\text{indices}(S_z)_t$, $\text{indices}(S_z)_v$, and $\text{indices}(S_z)_h$ correspond, respectively, to the three sets of consecutive coordinates on the lattice:

$$\begin{aligned}\Pi_t &= \{(\bar{i},j,k) : \bar{i} \in U_\rho(i)\},\\ \Pi_v &= \{(i,\bar{j},k) : \bar{j} \in U_\rho(j)\},\\ \Pi_h &= \{(i,j,\bar{k}) : \bar{k} \in U_\rho(k)\}.\end{aligned}$$

Since we require $\zeta \in U_\rho(\zeta)$ [or $\zeta \in U_{\rho,v}(\zeta)$], the three sets of coordinates intersect on the point $(z,y,x) = (i,j,k)$. Moreover, all three intervals Π_t, Π_v, Π_h have length at most ρ. Combining these, we find that the support of S_z indexed by (i,j,k) is contained in in a $\rho\times\rho\times\rho$ neighborhood of the point $(z,y,x) = (i,j,k)$ and has cardinality at most 3ρ. In other words, we show that the support of S_z is contained on a 3D cross of qubits with arms of length at most ρ.

As previously said, the 3D toric and planar codes are particular instances of the 3D product construction. Furthermore, it is well known that they are local on a torus and in the Euclidean space, respectively. To see how this is the case, we remind the reader that the 3D toric code is obtained by choosing as seed matrices the degenerate parity-check matrix of the repetition code in the standard basis. For matrix size $L\times L$, it holds that

$$\begin{aligned}\{1,\dots,L\} &\longleftrightarrow \{1,\dots,L\}\\ i &\longrightarrow \{i, i+1 \bmod L\}\\ \{\alpha, \alpha+1 \bmod L\} &\longleftarrow \alpha.\end{aligned}$$

Therefore, stabilizers have support on pairs of consecutive edges, and it is straightforward to see that they have the usual shape:

1. Z stabilizers have support on edges incident to a vertex;
2. X stabilizers have support on edges on the boundary of a square face;
3. metachecks have support on the faces of a cube.

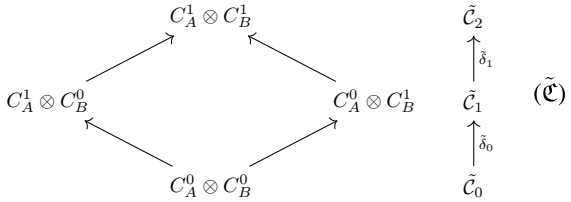
A similar argument holds for the 3D surface code, which is local in Euclidean space.

APPENDIX C: ALL 3D PRODUCT CODES HAVE X-CONFINEMENT

In this section we prove Theorem 1, which states that all 3D product codes have X confinement. Our proof follows the proof of soundness for 4D codes given in Ref. [18] with some minor adaptions and it is here reported for completeness.

First, we show that an opportunely chosen length-2 chain complex has confined maps. Secondly, we explain how to use this chain complex as a building block of the length-3 chain complex ($\mathfrak{C}'''$) described in Sec. IV. Lastly, we prove that the confinement property is preserved and thus 3D codes defined on ($\mathfrak{C}'''$) as explained in Sec. IV have X confinement.

Let $\delta_A : C_A^0 \to C_A^1$ and $\delta_B : C_B^0 \to C_B^1$ be two length-1 chain complexes. We consider the length-2 product complex $\tilde{\mathfrak{C}}$ defined as



$(\tilde{\mathfrak{C}})$

where

$$\tilde{\delta}_0 = \begin{pmatrix}\delta_A\otimes\mathbb{1} & \mathbb{1}\otimes\delta_B\end{pmatrix},$$
$$\tilde{\delta}_1 = \begin{pmatrix}\mathbb{1}\otimes\delta_B\\ \delta_A\otimes\mathbb{1}\end{pmatrix}.$$

We first show that the map $\tilde{\delta}_0$ has confinement.

Lemma 9. *$\tilde{\delta}_0$ has (t,f) confinement where $t = \min\{d_A, d_B\}$ and $f(x) = x^2/4$.*

In order to prove Lemma 9 we first introduce some useful notation. When considering vectors v in a twofold tensor product space $\mathbb{F}^{n_1}\otimes\mathbb{F}^{n_2}$ it can be handy to consider their reshaping, which is a $n_1 \times n_2$ matrix on $\mathbb{F}$. Namely, fixed bases $\mathcal{B}^1 = \{a_1, \ldots, a_{n_1}\}$ and $\mathcal{B}^2 = \{b_1, \ldots, b_{n_2}\}$ of $\mathbb{F}^{n_1}$ and $\mathbb{F}^{n_2}$, respectively, their product

$$\mathcal{B} = \{a_i \otimes b_j\}_{\substack{i=1,\ldots,n_1 \\ j=1,\ldots,n_2}}$$

is a basis of $\mathbb{F}^{n_1}\otimes\mathbb{F}^{n_2}$. Therefore, we can write

$$v = \sum_{a_i\otimes b_j \in \mathcal{B}} v_{ij}\, a_i \otimes b_j \tag{C1}$$

for some $v_{ij} \in \mathbb{F}$. We call the matrix V whose entries are the coefficient v_{ij} the reshaping of v. Given matrices M and N of size $n_1 \times m_1$ and $n_2 \times m_2$ associated to linear maps from $\mathbb{F}^{n_1}$ and $\mathbb{F}^{n_2}$, respectively, the map $M\otimes N$ from $\mathbb{F}^{n_1}\otimes\mathbb{F}^{n_2}$ to $\mathbb{F}^{m_1}\otimes\mathbb{F}^{m_2}$ acts on the reshaping of v as

$$(M\otimes N)V \longmapsto MVN^T. \tag{C2}$$

In the following we always indicate with lowercase symbol vectors and with the corresponding uppercase symbols their reshaping. We can now use this notation to prove Lemma 9. Let $v \in C_0^A \otimes C_0^B$ and let $s = \tilde{\delta}_0(v)$. By reshaping,

$$S = \begin{pmatrix} \delta_A V \\ V\delta_B^T \end{pmatrix}.$$

If we assume $|v| = |v|^{\mathrm{red}} \leq t = \min\{d_A, d_B\}$ then V has no column in $\ker\delta_A$ and no row in $\ker\delta_B^T$ so that

$$\mathrm{col}(\delta_A V) = \mathrm{col}(V) \text{ and } \mathrm{row}(V\delta_B^T) = \mathrm{row}(V),$$

where $\mathrm{col}(V)/\mathrm{row}(V)$ is the number of nonzero columns and rows of the matrix V. Therefore, for the weight of S, it holds that

$$\begin{aligned} |S| = |\delta_A V| + |V\delta_B^T| &\geq \mathrm{col}(\delta_A V) + \mathrm{row}(V\delta_B^T) \\ &= \mathrm{col}(V) + \mathrm{row}(V). \end{aligned}$$

Combining this with $(a+b)^2/4 \geq ab$ for integers a, b yields

$$|S|^2/4 \geq \mathrm{col}(V)\cdot\mathrm{row}(V) \geq |V|.$$

Corollary 1 below follows easily from Lemma 9.

Corollary 1. *If δ_A or δ_B have (g,t) confinement with g increasing and subadditive [69] then $\tilde{\delta}_0$ has (g,t) confinement too.*

Without loss of generality, we assume that δ_A has (t,g) confinement (the proof for δ_B being symmetrical).

Consider the syndrome matrix

$$S = \begin{pmatrix} \delta_A V \\ V\delta_B^T \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix},$$

where V is the reshaping of a vector v of reduced weight less than t, i.e., $|v| = |v|^{\mathrm{red}} \leq t$. Because δ_A has confinement, for the column of V it holds that

$$|S_1^j| \geq g(|V^j|).$$

Thus, we can use confinement columnwise and obtain

$$\begin{aligned} |S_1| &= \sum_j |S_1^j| && \text{by definition of } |\cdot| \\ &\geq \sum_j g(|V^j|) && \text{by confinement of } \delta_A \\ &\geq g\Big(\sum_j |V^j|\Big) && \text{by subadditivity of } g \\ &\geq g(|V|) && \text{by definition of } |\cdot|. \end{aligned}$$

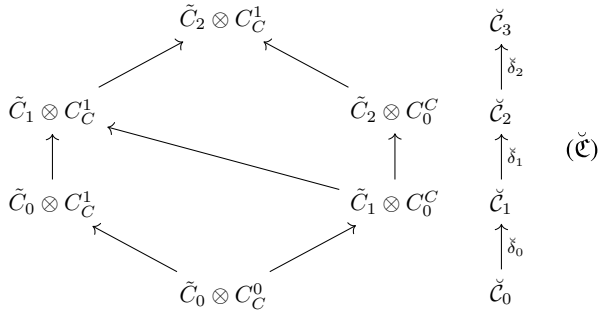
Combining this and

$$|S| = |S_1| + |S_2| \geq |S_1|$$

yields $|S| \geq g(|V|)$.

Loosely, Corollary 1 states that the result of Lemma 9 can be improved whenever at least one of the seed matrices δ_A and δ_B used to build the length-2 product complex $(\tilde{\mathfrak{C}})$ shows linear confinement. However, this is not sufficient to prove that the quantum code $\mathcal{C}(\mathfrak{C}_1)$ associated to $(\tilde{\mathfrak{C}})$ by equating $H_X = \tilde{\delta}_1$, $H_Z = \tilde{\delta}_0^T$ has confinement. In fact, here we prove that the matrix H_Z^T has confinement and not that one of the syndrome matrices H_Z or H_X have it. In other words, Corollary 1 it is not sufficient to infer the construction of expander codes outlined in Ref. [20]; here confinement goes in the "wrong" direction, namely as the transpose of the syndrome map. Even if not interestingly on its own, Corollary 1 can be used to improve the confinement function of the X-syndrome map of the code $\mathcal{C}(\delta_A, \delta_B, \delta_C)$.

More generally, we want to use Lemma 9 to infer that the code $\mathcal{C}(\delta_A, \delta_B, \delta_C)$ defined on the chain complex $(\mathfrak{C}''')$ has X confinement. To see how this is the case, we consider an "asymmetrical" version of $(\mathfrak{C}''')$ as the product of the length-2 chain complex $(\tilde{\mathfrak{C}})$ and the length-1 chain complex $\delta_C : C_0^C \to C_1^C$. The asymmetric product complex $\breve{\mathfrak{C}}$ is then



where

$$\check{\delta}_0 = \begin{pmatrix} \mathbb{1}\otimes\delta_C \\ \tilde{\delta}_0\otimes\mathbb{1} \end{pmatrix},$$
$$\check{\delta}_1 = \begin{pmatrix} \tilde{\delta}_0\otimes\mathbb{1} & \mathbb{1}\otimes\delta_C \\ 0 & \tilde{\delta}_1\otimes\mathbb{1} \end{pmatrix},$$
$$\check{\delta}_2 = \begin{pmatrix} \tilde{\delta}_1\otimes\mathbb{1} & \mathbb{1}\otimes\delta_C \end{pmatrix}.$$

Claim 1. *Let $(v,w) \in \check{C}_1$ have weight less than t and $s = \check{\delta}_1(v,w)$ be its syndrome. If (V,W) is the reshaping of the vector (v,w) then the following syndrome equation holds:*

$$S = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} \tilde{\delta}_0 V + W\delta_C^T \\ \tilde{\delta}_1 W \end{pmatrix}, \qquad \text{(SE)}$$

where S is the reshaping of s.

Note that a stabilizer for the chain complex $\check{C}_0 \to \check{C}_1 \to \check{C}_2 \to \check{C}_3$ and the syndrome map $\check{\delta}_1(\cdot)$ has the form $\check{\delta}_0(m)$ for some $m \in \check{C}_0$. By construction, we can add any stabilizer to (v,w) without violating the syndrome Eq. (SE). In particular,

1. *$|(v,w)| < t$ entails that its reshaping satisfies the following properties:*
   - *(a) Both V and W have at most t nonzero rows. Thus all their columns have weight at most t.*
   - *(b) Both V and W has at most t nonzero columns. Thus all their rows have weight at most t.*
2. *Fix a row index i and a column index j. Let M be a matrix in $\check{C}_0$ with columns*

$$M^h = \begin{cases} V^j & \text{for } h \text{ in } \operatorname{supp}(W\delta_C^T)_i = \operatorname{supp}(W_i\delta_C^T), \\ 0 & \text{elsewhere.} \end{cases}$$

Its image $(M\delta_C^T, \tilde{\delta}_0 M)$ through $\check{\delta}_0$ is a stabilizer. Define V^* and W^* as

$$V^* = V + M\delta_C^T \quad \text{and} \quad W^* = W + \tilde{\delta}_0 M.$$

Observe that

- *(a) M is a matrix whose nonzero columns are equal to a column of V. Therefore, M has row support contained in the row support of V:*

$$\operatorname{row}(V^*) \subseteq \operatorname{row}(V).$$

- *(b) M is a matrix whose column support is $\operatorname{supp}(W_i\delta_C^T)$ for some row W_i of W. Therefore, M has column support contained in the column support of W:*

$$\operatorname{col}(W^*) \subseteq \operatorname{col}(W).$$

Lemma 10 (Inheritance of confinement)**.** *$\check{\delta}_1$ has (t,f) confinement, where $t = \min\{d_A, d_B, d_C\}$ and $f(x) = x^3/4$.*

Let $(v,w) \in \check{C}_1$ be such that $|(v,w)| = |(v,w)|^{\text{red}} \le t$ and $s = \check{\delta}_1(v,w)$ be its syndrome. Reshaping vectors into matrices [see Eqs. (C1) and (C2)] yields the following syndrome equation:

$$S = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} \tilde{\delta}_0 V + W\delta_C^T \\ \tilde{\delta}_1 W. \end{pmatrix} \qquad \text{(SE)}$$

We transform the vector (V,W) by adding stabilizers to it in order to change its column and row support. We do this by iterating the following two steps.

Step 1: Let i,j be row and column indices such that

(a) $(W\delta_C^T)_i \neq 0$ and $(S_1)_i = 0$;
(b) $(\tilde{\delta}_0 V)_{ij} \neq 0$ and $(W\delta_C^T)_{ij} = 1$.

Build a matrix M as in Claim 1.

Transform V and W accordingly:

$$V \longmapsto V + M\delta_C^T,$$
$$W \longmapsto W + \tilde{\delta}_0 M.$$

Note that in this way we are able to delete row i of $W\delta_C^T$.

Iterate this step until we obtain

$$\operatorname{row}(W\delta_C^T) \subseteq \operatorname{row}(S_1). \qquad \text{(C3)}$$

Step 2: Let i,j be row and column indices such that

(a) $(W\delta_C^T)^j \neq 0$ and $(S_1)^j = 0$; this entails $(\tilde{\delta}_0 V)^j = (W\delta_C^T)^j$;
(b) $(\tilde{\delta}_0 V)_{ij} = (W\delta_C^T)_{ij} = 1$.

Build a matrix M as in Claim 1.

Transform V and W accordingly:

$$V \longmapsto V + M\delta_C^T,$$
$$W \longmapsto W + \tilde{\delta}_0 M.$$

Note that in this way we are able to delete row i of $W\delta_C^T$ and by repeatedly doing so we can delete any column j

of $W(\delta_C^T)$, which does not belong to the column support of S_1.Iterate this step until we obtain

$$\mathrm{col}(W\delta_C^T) \subseteq \mathrm{col}(S_1). \tag{C4}$$

Let $\mathbf{M}$ be the matrix formed by summing over all the matrices M found during these two steps. Define V^* and W^* as

$$V^* = V + \mathbf{M}\delta_C^T \quad \text{and} \quad W^* = W + \tilde{\delta}_0\mathbf{M}.$$

We now proceed to prove an upper bound for the weight of W^* first and then one for the weight of V^*. By combining these two bounds we obtain the desired confinement relation between the weight of the syndrome and the weight of the error.

BOUND ON THE WEIGHT OF W^*

1. By Claim 1, no row of W^* has weight more than t and therefore none of them belongs to $\ker \delta_C^T$ so that $\mathrm{row}(W^*\delta_C^T) = \mathrm{row}(W^*)$. Combining this with Eq. (C3) yields

$$\mathrm{row}(W^*) \subseteq \mathrm{row}(S_1). \tag{C5}$$

2. By Claim 1, the column support of W^* is contained in the column support of W, which is equal to the column support of S_2, by assumption on its weight. Summing these up,

$$\mathrm{col}(W^*) \subseteq \mathrm{col}(S_2). \tag{C6}$$

3. Combining Eqs. (C5) and (C6) yields

$$|S_1||S_2| \geq |W^*|.$$

BOUND ON THE WEIGHT OF V^*.

1. By rearranging the syndrome Eq. (SE), we can write $\tilde{\delta}_0 V^* = S_1 + W^*\delta_C^T$. Equations (C3) and (C4) therefore entail

$$\mathrm{row}(\tilde{\delta}_0 V^*) \subseteq \mathrm{row}(S_1), \tag{C7}$$

and

$$\mathrm{col}(\tilde{\delta}_0 V^*) \subseteq \mathrm{col}(S_1). \tag{C8}$$

2. By Claim 1, the row support of V^* is contained in the row support of V, which has cardinality at most t. In particular, all the columns of V^* have weight at most t and therefore we can use the confinement property of the map $\tilde{\delta}_0$ columnwise (see Lemma 9). In other words, for each column j of V^*, the following holds:

$$\frac{|(\tilde{\delta}_0 V^*)^j|^2}{4} \geq |(V^*)^j|.$$

Combining this with Eq. (C7) yields

$$\frac{|\mathrm{row}(S_1)|^2}{4} \geq |(V^*)^j|. \tag{C9}$$

3. By Claim 1, no column of V^* has weight more than t and therefore none of them belongs to $\ker \tilde{\delta}_0$ so that $\mathrm{col}(V^*) = \mathrm{col}(\tilde{\delta}_0 V^*)$. By Eq. (C9) this entails

$$\mathrm{col}(V^*) \subseteq \mathrm{col}(S_1).$$

In other words, V^* has at most $|\mathrm{col}(S_1)|$ nonzero columns and combining this with Eq. (SE) yields

$$\frac{|\mathrm{row}(S_1)|^2}{4}\,|\mathrm{col}(S_1)| \geq |V^*|, \tag{C10}$$

which entails

$$\frac{1}{4}|S_1|^3 \geq |V^*|.$$

Since $|S| = |S_1| + |S_2|$ and $|(V, W)| = |V| + |W|$, we can add the bounds found for V^* and W^*. Observing that $(a+b)^3 \geq (a^3 + a^2b + ab)$ for integer a, b, we obtain that (v^*, w^*) is a vector equivalent to (v, w) [i.e., it satisfies the syndrome Eq. (SE)] for which it holds

$$\frac{1}{4}|s|^3 \geq |(v^*, w^*)|. \tag{C11}$$

In conclusion, since $|(v^*, w^*)| \geq |(v, w)| = |(v, w)|^{\mathrm{red}}$, we prove that $\check{\delta}_1$ has confinement with respect to the function $f(x) = x^3/4$.

It is sometimes possible to find a better confinement function f for the map $\check{\delta}_1$ when $\tilde{\delta}_0$ has (t, g) confinement, for instance, as per Corollary 1. In fact, in such a case, Eq. (C9) becomes

$$g(|\tilde{\delta}_0 V^{*j}|) \geq |(V^*)^j|,$$

and combining this with Eq. (C7) yields

$$g(\mathrm{row}(S_1)) \geq |(V^*)^j|.$$

Thanks to Eq. (C10) we obtain

$$g(\mathrm{row}(S_1))|\mathrm{col}(S_1)| \geq |V^*|,$$

which, for g increasing, entails

$$g(|S_1|)|S_1| \geq |V^*|.$$

Summing up,

$$|S_1||S_2| + g(|S_1|)|S_1| \geq |V^*| + |W^*|. \tag{C12}$$

In other words, depending on the confinement function g for $\tilde{\delta}_0$, Eq. (C12) can be used to find better confinement function f for $\breve{\delta}_1$. For instance, if $g(x) = \alpha x$ is linear then $f(x) = \hat{\alpha}x^2$, for $\hat{\alpha} = \max\{\alpha, 1\}$ is a confinement function for $\breve{\delta}_1$. To sum up, whenever at least one of the seed matrices $\delta_A, \delta_B, \delta_C$ has linear confinement, the associated 3D product code has quadratic confinement.

We do not rule out the existence of a direct relationship between the confinement function for the seed matrices δ_A, δ_B, and δ_C and the confinement function for the corresponding $\breve{\delta}_1$ map of their 3D product code. In fact, we do believe that the cubic factor of Lemma 10 is an artefact of our proof and not a tight bound. For instance, when considering the 3D toric or surface code we find a quadratic relationship between the error size and syndrome size that follows a surface area-perimeter law.

APPENDIX D: ON SOME PROPERTIES OF EXAPANDER CODES

Here we prove that the family of expander codes considered in Ref. [20] has the three properties stated in the main text, namely,

(i) they have full-rank parity-check matrices;
(ii) they have $(t, 3x)$ confinement with $t \in \Omega(d)$;
(iii) for every small error $|e| \leq 3$, $\sigma(e) > 1$.

Property (i) is true by assumption made by the authors in Ref. [20]. In order to prove property (ii) we use Corollary 9 of Ref. [20], which states that the code family considered has robustness. Robustness for a code is very similar to confinement but uses a slightly different notion of reduced weight, which, for an operator e, is defined as

$$|e|_{\mathcal{S}}^{\text{red}} := \min\{|e + s| : \ s \text{ is a stabilizer}\}.$$

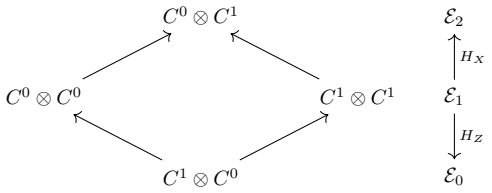
Our definition of reduced weight instead minimizes over all Pauli operators with the same syndrome and therefore it considers both stabilizers and logical operators. Because for the reduced weight we minimize over a bigger set, the following holds:

$$|e|_{\mathcal{S}}^{\text{red}} \geq |e|^{\text{red}}. \tag{D1}$$

Confinement follows combining the statement of Corollary 9 in Ref. [20] for errors such that $|e|_{\mathcal{S}}^{\text{red}} < d$ and Eq. (D1):

$$3|\sigma(e)| \geq |e|^{\text{red}}.$$

In order to prove property (iii) we need to make use of the hypergraph product structure of the expander codes in Ref. [20]. Briefly, the code family is defined by taking the two-product of the length-1 chain complex $\delta : C_0 \longrightarrow C_1$ with itself, where δ is an expander matrix (see also Ref. [70]). More precisely, the expander codes in Ref. [20] are CSS codes defined on the chain complex:



where

$$H_Z = \begin{pmatrix} \delta \otimes \mathbb{1} & \mathbb{1} \otimes \delta^T \end{pmatrix},$$
$$H_X = \begin{pmatrix} \mathbb{1} \otimes \delta & \delta^T \otimes \mathbb{1} \end{pmatrix}.$$

The matrix δ is chosen in a family of LDPC expander matrices with full rank and constant column and row weight w_c and w_r bigger than two.

We prove property (iii) for X errors e and the syndrome map $\sigma(e) = H_Z\bar{e}$, where $\bar{e}$ is the binary vector representation of the Pauli operator e; the proof for Z errors and syndrome map H_X follows by duality with minor changes.

Let e be a weight-1 X operator and $\bar{e}$ its representation as a unit vector. Then $\sigma(e) = H_Z\bar{e}$ is a column of the matrix H_Z, namely column j if $\bar{e}$ has jth coordinate equal to 1. Since the seed matrix δ has constant column and row degree bigger than 2, so has the matrix H_Z and therefore $|\sigma(e)| \geq 2$.

Consider now a weight-2 error operator e. By reshaping of vectors into matrices we can write e as

$$(L, R)$$

for some binary matrices L of size $n \times n$, R of size $m \times m$, and such that $|L| + |R| = 2$, where δ has size $m \times n$. Following this notation, the syndrome S of E can be written as

$$S = \delta L + R\delta.$$

We have three cases to be distinguished.

(a) $|L| = 2$. If the support of L is not contained in one column, i.e., $L_{i_1,j_1} = L_{i_2,j_2} = 1$ and $j_1 \neq j_2$, then for the syndrome $S = \delta L$ the following holds:

$$S^{j_1} = \delta^{i_1} \quad \text{and} \quad S^{j_2} = \delta^{i_2},$$

i.e., the syndrome matrix S has at least two nonzero columns and therefore weight at least 2. If instead the support of L is contained in one column, $L_{i_1,j} =$

$L_{i_2,j} \neq 0$ then the syndrome S is zero but for the j th column:

$$S^j = \delta^{i_1} + \delta^{i_2}.$$

In this case, whenever δ has distance at least 3, because it has constant column weight w_c, the following holds:

$$\begin{aligned} |\delta^{i_1} + \delta^{i_2}| &= |\delta^{i_1}| + |\delta^{i_2}| - 2|\delta^{i_1} \wedge \delta^{i_2}| \\ &\geq 2w_c - 2(w_c - 1) \\ &\geq 2. \end{aligned}$$

Where the second to last inequality holds because if δ defines code of distance bigger than 3, then it must have all distinct columns and different vectors of constant weight w_c overlap in at most $w_c - 1$ positions.

(b) $|L| = |R| = 1$. Suppose $L_{i_1,j_1} = R_{i_2,j_2} = 1$. The syndrome S has support contained in one column and one row, in the shape of a cross as follows:

$$\begin{aligned} S_{k,j_1} &= \delta_{k,i_1}, & k \neq i_2 \\ S_{i_2,k} &= \delta_{j_2,k}, & k \neq j_1 \\ S_{i_2,j_1} &= \delta_{i_2,i_1} + \delta_{j_2,j_1}, & \\ S_{ij} &= 0, & \text{otherwise.} \end{aligned}$$

It then follows for the weight of the syndrome S that

$$|S| \geq w_c + w_r - 1,$$

which is bigger that 1 by assumption on the column and row weight of δ.

(c) The case $|R| = 2$ can be proven as done in (a) for $|L| = 2$ by exchanging the role of columns and rows.

To sum up, whenever $|e| = 2$, $|\sigma(e)| > 2$.

Last, consider a weight-3 error e. As done for weight-2 errors, we can write e as (L, R) for some binary matrices L and R. Again, we need to distinguish between the possible weight combinations of $|L| + |R| = 3$. We now prove

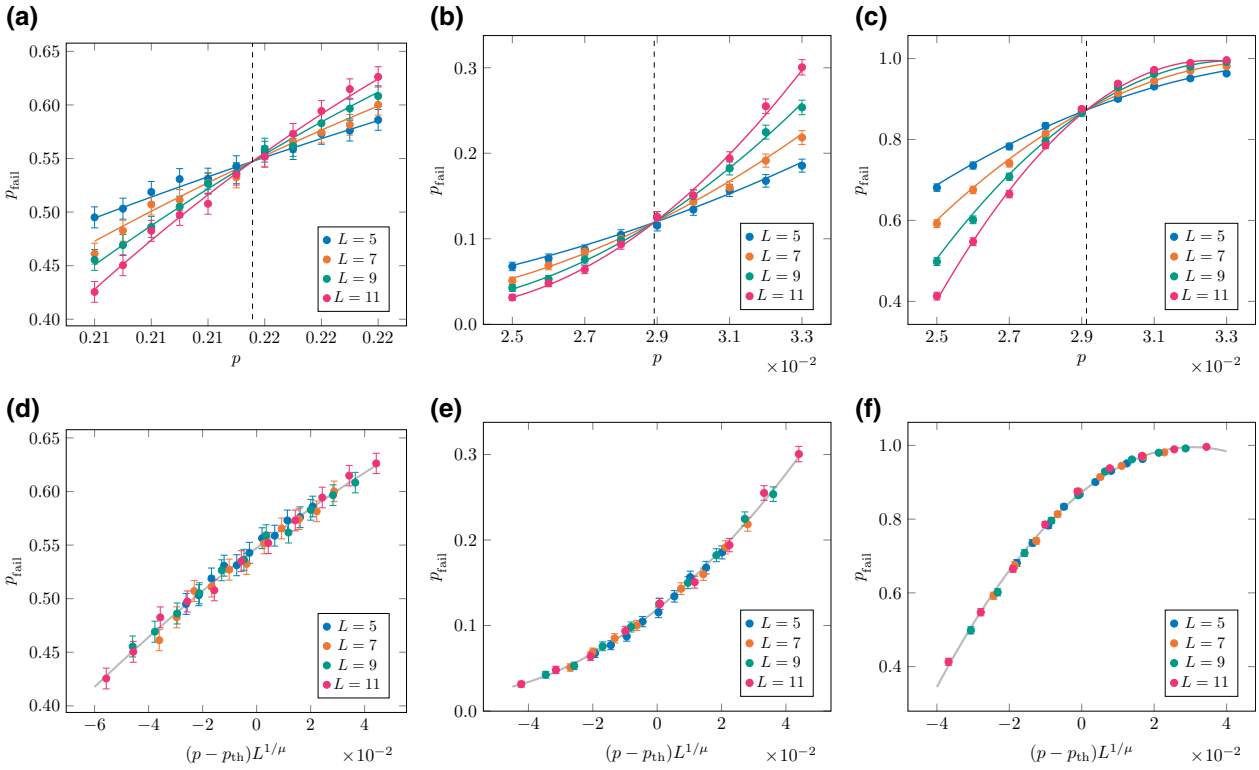


FIG. 9. Threshold fits for the 3D toric code using MWPM and BP+OSD to decode. In (a), we plot the logical error rate p_{fail} as a function of the phase-flip error rate p, for values of p close to the threshold. The colored lines show the fit given by Eq. (E1), with parameters $a_0 = 0.547$, $a_1 = 1.92$, $a_2 = -4.04$, $\mu = 1.04$, and $p_{\text{th}} = 0.216$ (dashed gray line). In (d), we show the same data using the rescaled variable $x = (p - p_{\text{th}})L^{1/\mu}$. Subfigures (b) and (e) show equivalent data for one round of single-shot error correction, with fit parameters $a_0 = 0.119$, $a_1 = 3.04$, $a_2 = 22.9$, $\mu = 1.01$, and $p_{\text{th}} = 0.0289$. Subfigures (c) and (f) show equivalent data for 16 rounds of single-shot error correction, with fit parameters $a_0 = 0.873$, $a_1 = 7.99$, $a_2 = -130$, $\mu = 1.10$, and $p_{\text{th}} = 0.0291$. The error bars show the 95% confidence intervals $p_{\text{fail}} = \hat{p}_{\text{fail}} \pm 1.96\sqrt{p_{\text{fail}}(1 - p_{\text{fail}})/\eta}$, where $\eta \geq 10^4$ is the number of Monte Carlo trials.

the case for $|L| = 3$ and support of L contained in one column. The other cases are either a dual argument of this one (i.e., for $|R| = 3$ supported on one row) or follows easily adapting the proof for $|e| = 2$.

Let e be a weight-3 error operator with reshaping (L, R) such that $|L| = 3$ and $L_{i_1,j} = L_{i_2,j} = L_{i_3,j} = 1$, for some column index j. In this case, the syndrome matrix S has support contained in its j th column:

$$S^j = \delta^{i_1} + \delta^{i_2} + \delta^{i_3},$$

and therefore,

$$|\sigma(e)| = |S| = |S^j| = |\delta^{i_1} + \delta^{i_2} + \delta^{i_3}|.$$

In order to prove $|\sigma(e)| = |S^j| > 2$, we need to use the expansion properties of δ and more specifically Lemma 3 of Ref. [20], (see also Ref. [70]). We first introduce some notation. We refer to the rows of δ as checks and to its columns as bits; we say that a bit j is in the support of the check i if and only if $\delta_{ij} = 1$. Given a set of bits $B \subseteq \{1, \ldots, n\}$ we say that the check $i \in \{1, \ldots, m\}$

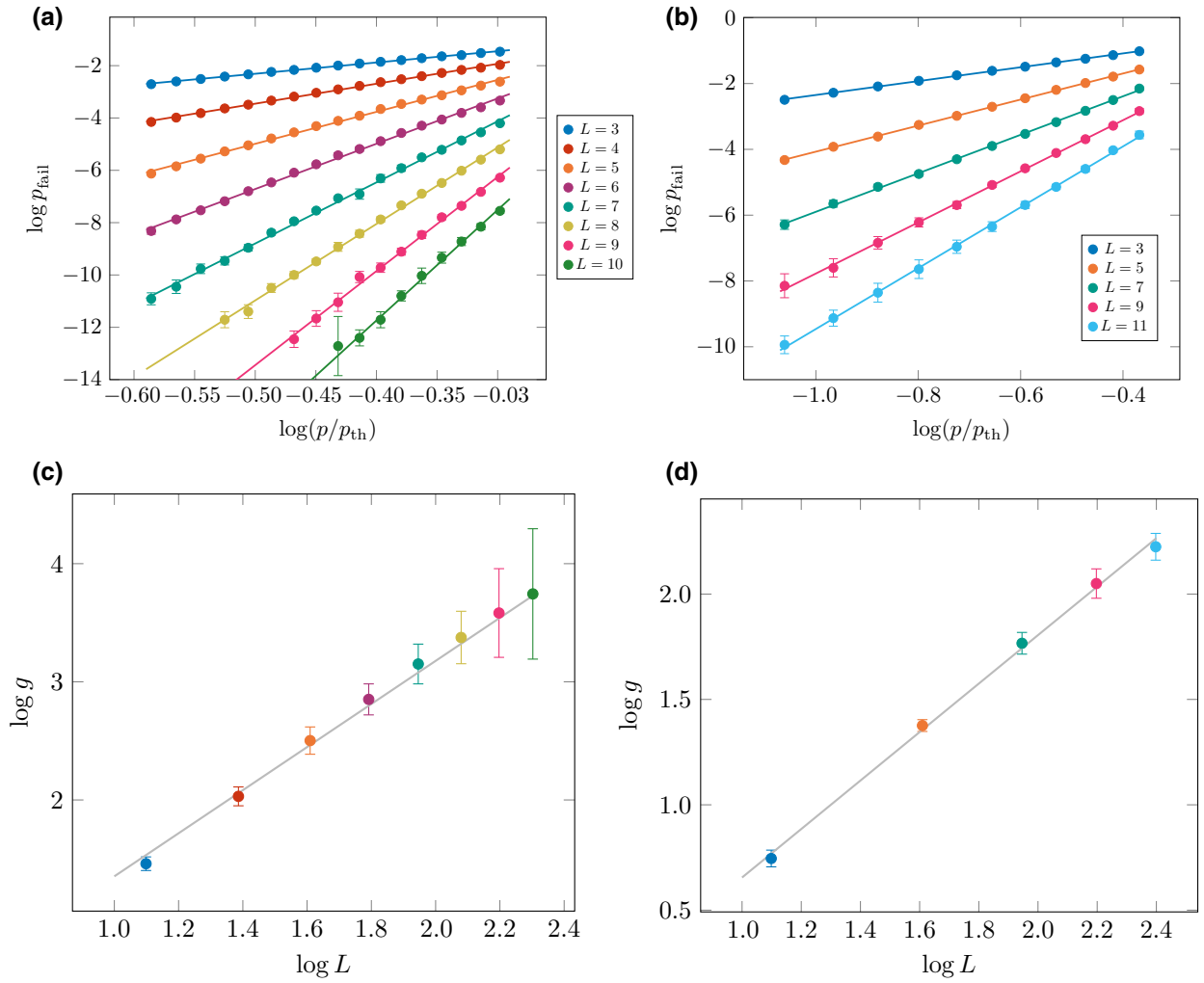


FIG. 10. Illustration of the fitting procedure for finding the coefficients describing the suppression of the logical error rate for phase-flip error rates substantially below threshold. (a),(c) Data for code-capacity noise (no measurement errors), and (b),(d) show data for eight rounds of single-shot error correction. In both cases, we first plot $\log p_{\text{fail}}$ as a function of $\log(p/p_{\text{th}})$ for differing values of L, observing trends that agree with the straight line prediction of Eq. (E3) [(a),(b)]. We note that for the single-shot case there is an odd-even effect so we include only the data for odd L. We extract the gradients $g(L)$ from the corresponding straight line fits in (a),(b) (gray lines), and plot the logarithms of these values against $\log L$ [(c),(d)]. The data fit well to the linear ansatz given in Eq. (E5), which allows us to estimate the parameters α and β, which control the suppression of the logical error rate as per Eq. (E2). For code-capacity noise, we estimate $\alpha = 0.546(33)$ and $\beta = 1.91(3)$, and for eight rounds of single-shot error correction, we estimate $\alpha = 0.610(37)$ and $\beta = 1.15(3)$. The error bars in (a),(b) show the 95% confidence intervals $\log p_{\text{fail}} = \log \hat{p}_{\text{fail}} \pm \frac{1.96}{p_{\text{fail}}} \sqrt{p_{\text{fail}}(1 - p_{\text{fail}})/\eta}$, where $\eta \geq 10^4$ is the number of Monte Carlo trials. We include data points with at least 25 failures. The error bars in (c),(d) show the 95% confidence intervals given by the LINEARMODELFIT function of Mathematica.

is a unique neighbor of B if and only if one and only one bit in B belongs to the support of the check i. We indicate with $\Gamma_u(B)$ the set of unique neighbors of B. Lemma 3 in Ref. [20] states that, for the considered class of matrices δ:

$$|\Gamma_u(B)| \geq \frac{2}{3} w_c |B|.$$

Combining this with the observation that $|\Gamma_u(\{i_1, i_2, i_3\})|$ is a lower bound on $|S^j|$ and plugging in $|B| = 3$, we find

$$|S^j| \geq |\Gamma_u(\{i_1, i_2, i_3\})| \geq 2w_c.$$

To sum up, whenever an error (L, R) of weight 3 has support on either one column of L or one row of R, by expansion its syndrome has weight strictly bigger than 1. When instead a weight-3 error has support spread among more than one column and row it is enough to use the hypergraph product structure of the code family, as done for weight-2 errors, to find that its syndrome need to have weight at least 2.

APPENDIX E: FITTING DETAILS

To obtain our threshold estimates, we use the standard critical exponent method [59]. Specifically, in the vicinity of the threshold, we fit our data to the following ansatz:

$$a_0 + a_1 x + a_2 x^2, \tag{E1}$$

where the rescaled variable $x = (p - p_{th})L^{1/\mu}$. Examples of this fit are shown in Fig. 9.

We use the fitting method described in Ref. [12] to understand the behavior of the 3D toric code logical error rate for error rates p significantly below threshold. Recall from Sec. V that we use the following ansatz:

$$p_{\text{fail}}(L) \propto (p/p_{\text{th}})^{\alpha L^{\beta}}, \tag{E2}$$

we take the logarithm of both sides to obtain

$$\log p_{\text{fail}} = \log f\,(L) + \alpha L^{\beta} \log(p/p_{\text{th}}). \tag{E3}$$

For different values of L, we plot $\log p_{\text{fail}}$ as a function of $\log(p/p_{\text{th}})$ and fit to a straight line to obtain gradients

$$g(L) = \frac{\partial \log p_{\text{fail}}}{\partial u} = \alpha L^{\beta}, \tag{E4}$$

where $u = \log(p/p_{\text{th}})$. Finally, take the logarithm of both sides of the above to give

$$\log g = \log \alpha + \beta \log L. \tag{E5}$$

We then plot $\log g$ as a function of $\log L$ and fit to a straight line to get α and β. Figure 10 illustrates the above fitting procedure for code-capacity noise (no measurement errors) and for eight rounds of single-shot error correction.

[1] Joschka Roffe, Quantum error correction: An introductory guide, Contemporary Phys. **60**, 226 (2019).

[2] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

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

[4] R. Raussendorf, J. Harrington, and K. Goyal, Topological fault-tolerance in cluster state quantum computation, New J. Phys. **9**, 199 (2007).

[5] A. Bolt, G. Duclos-Cianci, D. Poulin, and T. M. Stace, Foliated Quantum Error-Correcting Codes, Phys. Rev. Lett. **117**, 070501 (2016).

[6] Naomi Nickerson and Héctor Bombín, Measurement based fault tolerance beyond foliation, arXiv:1810.09621 (2018).

[7] Hector Bombin, 2D quantum computation with 3D topological codes, arXiv:1810.09571 (2018).

[8] Benjamin J. Brown, A fault-tolerant non-Clifford gate for the surface code in two dimensions, arXiv:1903.11634 (2019).

[9] Michael Newman, Leonardo Andreta de Castro, and Kenneth R. Brown, Generating fault-tolerant cluster states from crystal structures, arXiv:1909.11817 (2019).

[10] Héctor Bombín, Single-Shot Fault-Tolerant Quantum Error Correction, Phys. Rev. X **5**, 031043 (2015).

[11] Héctor Bombín, Resilience to Time-Correlated Noise in Quantum Computation, Phys. Rev. X **6**, 041034 (2016).

[12] Benjamin J. Brown, Naomi H. Nickerson, and Dan E. Browne, Fault-tolerant error correction with the gauge color code, Nat. Commun. **7**, 1 (2016).

[13] Kasper Duivenvoorden, Nikolas P. Breuckmann, and Barbara M. Terhal, Renormalization group decoder for a four-dimensional toric code, IEEE Trans. Inf. Theory **65**, 2545 (2018).

[14] Nikolas P. Breuckmann and Vivien Londe, Single-shot decoding of linear rate LDPC quantum codes with high performance, arXiv:2001.03568 (2020).

[15] Aleksander Kubica, Ph.D. thesis, Caltech, 2018.

[16] Aleksander Kubica and John Preskill, Cellular-Automaton Decoders with Provable Thresholds for Topological Codes, Phys. Rev. Lett. **123**, 020501 (2019).

[17] Michael Vasmer, Dan E. Browne, and Aleksander Kubica, Cellular automaton decoders for topological quantum codes with noisy measurements and beyond, arXiv:2004.07247 (2020).

[18] Earl T. Campbell, A theory of single-shot error correction for adversarial noise, Quantum Sci. Technol. **4**, 025006 (2019).

[19] Jean-Pierre Tillich and Gilles Zémor, Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength, IEEE Trans. Inf. Theory **60**, 1193 (2014).

[20] Anthony Leverrier, Jean-Pierre Tillich, and Gilles Zémor, in *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on* (IEEE, 2015), p. 810.

[21] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier, in *Proc. STOC* (ACM, 2018), p. 521.

[22] Omar Fawzi, Antoine Grospellier, and Anthony Leverrier, Constant overhead quantum fault tolerance with quantum expander codes. to appear in FOCS 2018, (2018).

[23] Austin G. Fowler, Time-optimal quantum computation, arXiv:1210.4626 (2012).

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

[25] Poulami Das, Christopher A. Pattison, Srilatha Manne, Douglas Carmean, Krysta Svore, Moinuddin Qureshi, and Nicolas Delfosse, A Scalable Decoder Micro-architecture for Fault-Tolerant Quantum Computing, arXiv:2001.06598 (2020), p. 1.

[26] Antoine Grospellier and Anirudh Krishna, Numerical study of hypergraph product codes, arXiv:1810.03681 (2018).

[27] Antoine Grospellier, Lucien Grouès, Anirudh Krishna, and Anthony Leverrier, Combining hard and soft decoders for hypergraph product codes, arXiv:2004.11199 (2020).

[28] Sergey Bravyi and Matthew B. Hastings, in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing* (ACM, 2014), p. 273.

[29] Benjamin Audoux and Alain Couvreur, On tensor products of CSS codes, arXiv:1512.07081 (2015).

[30] Arun B. Aloshious and Pradeep Kiran Sarvepalli, Decoding toric codes on three dimensional simplical complexes, arXiv:1911.06056 (2019).

[31] Nikolas P. Breuckmann and Xiaotong Ni, Scalable neural network decoders for higher dimensional quantum codes, Quantum **2**, 68 (2018).

[32] Yukiyasu Ozeki and Nobuyasu Ito, Multicritical dynamics for the ±J Ising model, J. Phys. A: Math. General **31**, 5451 (1998).

[33] Takuya Ohno, Gaku Arakawa, Ikuo Ichinose, and Tetsuo Matsui, Phase structure of the random-plaquette $\mathbb{Z}_2$ gauge model: Accuracy threshold for a toric quantum memory, Nucl. Phys. B **697**, 462 (2004).

[34] Martin Hasenbusch, Francesco Parisen Toldin, Andrea Pelissetto, and Ettore Vicari, Magnetic-glassy multicritical behavior of the three-dimensional ±J Ising model, Phys. Rev. B - Condens. Matter Mater. Phys. **76**, 184202 (2007).

[35] Koujin Takeda and Hidetoshi Nishimori, Self-dual random-plaquette gauge model and the quantum toric code, Nucl. Phys. B **686**, 377 (2004).

[36] Aleksander Kubica, Michael E. Beverland, Fernando Brandão, John Preskill, and Krysta M. Svore, Three-Dimensional Color Code Thresholds via Statistical-Mechanical Mapping, Phys. Rev. Lett. **120**, 180501 (2018).

[37] Alexey A. Kovalev and Leonid P. Pryadko, Fault tolerance of quantum low-density parity check codes with sublinear distance scaling, Phys. Rev. A **87**, 020304 (2013).

[38] Sergey Bravyi and Matthew B. Hastings, in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing* (ACM, 2014), p. 273.

[39] Allen Hatcher, *Algebraic Topology. 2002* (Cambridge UP, Cambridge, 2002), p. 606.

[40] Weilei Zeng and Leonid P. Pryadko, Higher-Dimensional Quantum Hypergraph-Product Codes with Finite Rates, Phys. Rev. Lett. **122**, 230501 (2019).

[41] Jack Edmonds, Paths, trees, and flowers, Canadian J. Math. **17**, 449 (1965).

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

[43] Nikolas P. Breuckmann and Barbara M. Terhal, Constructions and noise threshold of hyperbolic surface codes, IEEE Trans. Inf. Theory **62**, 3731 (2016).

[44] Benjamin J. Brown and Dominic J. Williamson, Parallelized quantum error correction with fracton topological codes, Phys. Rev. Res. **2**, 013303 (2020).

[45] Aleksander Kubica and Nicolas Delfosse, Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders, arXiv:1905.07393 (2019).

[46] Namely, the dual lattice of the one described in Appendix B.

[47] Vladimir Kolmogorov, Blossom V: A new implementation of a minimum cost perfect matching algorithm, Math. Program. Comput. **1**, 43 (2009).

[48] David J. C. MacKay and Radford M. Neal, Near shannon limit performance of low density parity check codes, Electron. Lett. **33**, 457 (1997).

[49] F. R. Kschischang, B. J. Frey, and H. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. Inf. Theory **47**, 498 (2001).

[50] David Poulin and Yeojin Chung, On the iterative decoding of sparse quantum codes, Quantum Info. Comput. **8**, 987 (November 2008).

[51] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, Fifteen years of quantum ldpc coding and improved decoding strategies, IEEE Access **3**, 2492 (2015).

[52] Ye-Hua Liu and David Poulin, Neural Belief-Propagation Decoders for Quantum Error-Correcting Codes, Phys. Rev. Lett. **122**, 200501 (2019).

[53] Alex Rigby, J. C. Olivier, and Peter Jarvis, Modified belief propagation decoders for quantum low-density parity-check codes, Phys. Rev. A **100**, 012330 (Jul 2019).

[54] M. Li and T. J. Yoder, in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2020), p. 109.

[55] Pavel Panteleev and Gleb Kalachev, Degenerate quantum LDPC codes with good finite length performance, arXiv:1904.02703 (2019).

[56] Joschka Roffe, David R. White, Simon Burton, and Earl Campbell, Decoding across the quantum low-density parity-check code landscape, Phys. Rev. Res. **2**, 043423 (Dec 2020).

[57] Joschka Roffe, BP+OSD - a decoder for sparse quantum codes. *github.com*.

[58] More precisely, rows of L_M are vectors in $\mathcal{C}_2$ that correspond to elements of the second cohomology group $\mathcal{H}_2^*$; hence their weight is lower bounded by $d_2^* = \min\{d_a^T d_b^T, d_a^T d_c^T, d_b^T d_c^T\}$, see Ref. [40].

[59] J. Harrington, Ph.D. thesis, Caltech, 2004.

[60] Chenyang Wang, Jim Harrington, and John Preskill, Confinement-higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory, Ann. Phys. **303**, 31 (2003).

[61] Sagar Vijay, Jeongwan Haah, and Liang Fu, Fracton topological order, generalized lattice gauge theory, and duality, Phys. Rev. B **94**, 235157 (2016).

[62] Arpit Dua, Isaac H. Kim, Meng Cheng, and Dominic J. Williamson, Sorting topological stabilizer models

in three dimensions, Phys. Rev. B **100**, 155137 (2019).
[63] www.computeontario.ca.
[64] www.computecanada.ca.
[65] Antoine Grospellier, Ph.D. thesis, Sorbonne universités, 2019.
[66] When the code is a CSS code we consider the group generated by the X stabilizers and the Z stabilizers separately. $\mathcal{S}$ will thus refer either to $\mathcal{S}_X$ or $\mathcal{S}_Z$.
[67] For example, if e_1 and e_2 are both X operators, $E_1 + E_2$ is the symmetric difference of the sets E_1 and E_2.
[68] $\binom{n}{k} \simeq 2^{nh(k/n)}$, where $h(x) = x\log_2(1/x) + (1-x)\log_2[1/(1-x)]$ is the binary entropy function.
[69] A real function is said subadditive if for any $x, y \in \mathbb{R}$, $g(x) + g(y) \geq g(x+y)$.
[70] Michael Sipser and Daniel A. Spielman, Expander codes, IEEE Trans. Inf. Theory **42**, 1710 (1996).

Chapter 6

Outlook and Open problems

In this work, we have addressed some of the most fundamental issues in the design of a full-stack hypergraph-product-codes quantum computer. We believe that the study of hypergraph product codes could provide promising new approaches to the quantum error correction paradigm, particularly informing novel techniques that use to good advantage constant rate codes. We discussed the limitations – and henceforth possible directions – of our work in the relevant Sections but beyond those, there are a few open problems that are of particular interest to us.

First and foremost, we lack a comprehensive framework for encoded operations that fully exploits the logical dimension of the code. Current proposals [1, 2] rely on auxiliary patches of planar code and we believe this is sub-optimal. Transversal gates seem to lose much of their appeal when the underlying code has more than one logical qubit. In fact, they naturally tend to act on the entire logical group of the code and it is still unknown how to address single logical qubits without supplementary gadgets. Possibly completely new paradigms are needed.

A related, unexplored, problem is logical circuit compilation. The logical action of fault-tolerant gates on the encoded space will likely be tied to a generating set for the logical Pauli group. Once a logical basis is chosen, we expect different logical qubits to have inherently different properties in respect of minimum weight but also ease of implementation of some logical operations over others. Presumably, some but not all logical operations could be implemented in parallel. For all these reasons, ad hoc logical circuit compilation could be needed to make the most of the specific structure of the encoded space and the chosen universal set of fault-tolerant operations.

Decoding: decoding is not a solved problem. Efficient decoding algorithms with linear running time are known in theory, but the hidden constants are often too big in practice. Bringing the computational cost of decoding down could be the turning point for usable quantum computers [3, 4, 5, 6]. The current approach is to use a cheap 'pre-decoder' to correct the most common and trivial errors before the syndrome information is sent to a general-purpose decoder. Such kind of approach has not been studied for codes with $k > 1$ and hypergraph product codes more specifically.

Strictly related to the decoding problem is the design of fault-tolerant syndrome extraction circuits that are compatible with the current hardware [7, 8]. This will not only affect the overall performance of the computation but also shape and constrain the decoding problem that needs to be solved.

On a more speculative level, we would like to further investigate the relationship between confinement, as defined in Chapter 5, and energy barrier. Stabilizer codes can be mapped into the degenerate ground state of a Hamiltonian where Pauli errors correspond to excited states. The energy barrier of a code is then the minimum energy cost to implement a non-trivial logical operator, namely moving from one ground state to an orthogonal one, via a sequence of single-qubit Pauli errors [9, 10, 11]. Since confinement is a relation between the syndrome weight – roughly the energy cost – and the error weight, the two concepts are coupled. We know that a code with (macroscopic) confinement has a macroscopic energy barrier and we conjecture that the converse is also true. Our community agrees on the close link between confinement, energy barrier, single-shot error correction and self-correction. The exact nature of this link is still not fully understood.

Can we build this bridge?

A scalable quantum computer is one whose computational power can be increased on demand, but at the same time the resources used – manufacturing cost, physical space and energy usage – do not grow exponentially. To actually increase the computational power of a machine, qubits' quality and the ability to perform gates between them have to be preserved when increasing the number of available qubits [12, 13, 14, 15]. A figure of merit for the quality of the qubits is their relaxation times expressed as exponential decay constants. The longitudinal relaxation time measures the exponential decay of the probability that a qubit initialised in the $|1\rangle$ state is found in the $|0\rangle$ state. The transverse relaxation time measures the decay of a qubit state in some superposition $\alpha|0\rangle + \beta|1\rangle$. Equally important are the gate and measurement times, both in absolute terms and relative to the qubits' relaxation time. Long qubit relaxation times as well as fast gates and measurements naturally allow for longer computations. In addition, we need gates and measurement error rates to be as low as possible. Last, as the number of qubits grows, the number of possible multiple-qubit entangling gates – e.g. CNOTs – between different qubits has to substantially grow too. In other words, we say that the number of *connected* qubits has to grow.

A key issues in code design is indeed that the physical arrangement of qubits and consequent ability to perform gates has to be compatible with the chosen code family and unfortunately good qLDPC codes are not local in two-dimensions [9, 16]. Here by D-dimensional local quantum codes we mean a code family in which data qubits and auxilia qubits for stabilizer measurements have a layout in the D-dimensional real space such that each stabiliser's support only contains neighbouring qubits in the layout. Bravyi, Poulin and Terhal [16] proved that a $[\![n, k, d]\!]$ code in D dimensions obeys $kd^{\frac{2}{D-1}} \sim n$ and Baspin and Krishna [17] further proved that long-range connectivity is indeed essential to surpass this trade-off between parameters. Delfosse, Beverland and Tremblay [18] have shown that, when we are limited to local circuits for syndrome measurement, the number of auxilia needed grows with the number of physical qubits – as n or n^2, depending on the depth of the circuits. In particular in 2-dimensions, when $d \sim \sqrt{n}$, the number of encoded logical qubits has to be constant – as for the planar code [16]. Hypergraph product codes circumnavigate the no-go result of Ref. [16] by dropping locality constraints; as per Refs. [18, 19], a scalable implementation on hardware would require going past a two-dimensional physical layout of qubits with nearest-neighbour interactions.

To summarise, good codes need long-range connectivity but many current hardware implemen-

tations – e.g. superconducting qubits – do not accommodate for this. Superconducting qubits are constrained in terms of connectivity and do not have long relaxation times, in the order of microseconds, but have fast gates, in the order of nanoseconds [13]. On the other hand, trapped ion and neutral atom qubits, for example, are quite versatile in terms of their connectivity, and have longer relaxation times, in the seconds range, but slow gates, in the order of microseconds [14, 15]. The interplay between the different requirements for the construction of a useful quantum computer is a complex technological challenge. As of today, we do not know which qubit architecture will be favoured but we are optimistic it will overcome the constraints of a planar qubit layout to allow diverse quantum codes – hypergraph product codes and beyond.

References

[1] Lawrence Z Cohen et al. "Low-overhead fault-tolerant quantum computing using long-range connectivity". In: *Science Advances* 8.20 (2022).

[2] Anirudh Krishna and David Poulin. "Fault-tolerant gates on hypergraph product codes". In: *Physical Review X* 11.1 (2021), p. 011023.

[3] Nicolas Delfosse. "Hierarchical decoding to reduce hardware requirements for quantum computing". In: *arXiv preprint arXiv:2001.11427* (2020).

[4] Samuel C Smith, Benjamin J Brown, and Stephen D Bartlett. "A local pre-decoder to reduce the bandwidth and latency of quantum error correction". In: *arXiv preprint arXiv:2208.04660* (2022).

[5] Gokul Subramanian Ravi et al. "Have your QEC and Bandwidth too!: A lightweight cryogenic decoder for common/trivial errors, and efficient bandwidth+ execution management otherwise". In: *arXiv preprint arXiv:2208.08547* (2022).

[6] Luka Skoric et al. "Parallel window decoding enables scalable fault tolerant quantum computation". In: *arXiv preprint arXiv:2209.08552* (2022).

[7] Nicolas Guillaume Delfosse, Maxime Tremblay, and Michael Edward Beverland. *Short-depth syndrome extraction circuits in 2d quantum architectures for hypergraph product codes*. US Patent App. 17/219,331. 2022.

[8] Maxime A Tremblay, Nicolas Delfosse, and Michael E Beverland. "Constant-overhead quantum error correction with thin planar connectivity". In: *Physical Review Letters* 129.5 (2022), p. 050504.

[9] Sergey Bravyi and Barbara Terhal. "A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes". In: *New Journal of Physics* 11.4 (2009), p. 043029.

[10] Barbara M Terhal. "Quantum error correction for quantum memories". In: *Reviews of Modern Physics* 87.2 (2015), p. 307.

[11] Benjamin J Brown et al. "Quantum memories at finite temperature". In: *Reviews of Modern Physics* 88.4 (2016), p. 045005.

[12] Thaddeus D Ladd et al. "Quantum computers". In: *nature* 464.7285 (2010), pp. 45–53.

[13] Morten Kjaergaard et al. “Superconducting qubits: Current state of play”. In: *Annual Review of Condensed Matter Physics* 11 (2020), pp. 369–395.

[14] Colin D Bruzewicz et al. “Trapped-ion quantum computing: Progress and challenges”. In: *Applied Physics Reviews* 6.2 (2019), p. 021314.

[15] Mark Saffman. “Quantum computing with atomic qubits and Rydberg interactions: progress and challenges”. In: *Journal of Physics B: Atomic, Molecular and Optical Physics* 49.20 (2016), p. 202001.

[16] Sergey Bravyi, David Poulin, and Barbara Terhal. “Tradeoffs for reliable quantum information storage in 2D systems”. In: *Physical review letters* 104.5 (2010), p. 050503.

[17] Nouédyn Baspin and Anirudh Krishna. “Connectivity constrains quantum codes”. In: *Quantum* 6 (2022), p. 711.

[18] Nicolas Delfosse, Michael E Beverland, and Maxime A Tremblay. “Bounds on stabilizer measurement circuits and obstructions to local implementations of quantum LDPC codes”. In: *arXiv preprint arXiv:2109.14599* (2021).

[19] Armands Strikis and Lucas Berent. “Quantum LDPC Codes for Modular Architectures”. In: *arXiv preprint arXiv:2209.14329* (2022).