High-Threshold, Low-Overhead and Single-Shot Decodable Fault-Tolerant Quantum Memory

Thomas R. Scruby,[1,*] Timo Hillmann,[2,3,†] and Joschka Roffe[4,5,‡]

[1] *Okinawa Institute of Science and Technology, Okinawa, Japan*

[2] *Department of Microtechnology and Nanoscience, Chalmers University of Technology, 412 96 Gothenburg, Sweden*

[3] *School of Physics, University of Sydney, Sydney, NSW 2006, Australia*

[4] *Quantum Software Lab, School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, United Kingdom*

[5] *Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, Berlin, 14195, Germany*

(Received 27 October 2025; revised 9 February 2026; accepted 6 March 2026; published 17 April 2026)

We present a family of quantum low-density parity-check codes, which we call radial codes, obtained from the lifted product of a specific subset of classical quasicyclic codes. The codes are defined using a pair of integers (r, s) and have parameters $[\![2r^2s, 2(r-1)^2, \leq 2s]\!]$, with numerical studies suggesting average-case distance linear in s. In simulations of circuit-level noise, we observe comparable error suppression to surface codes of similar distance while using approximately five times fewer physical qubits. This is true even when radial codes are decoded using a single-shot approach, which can allow for faster logical clock speeds and reduced decoding complexity. We describe an intuitive visual representation, canonical basis of logical operators and optimal-length stabilizer measurement circuits for these codes, and argue that their error correction capabilities, tunable parameters and small size make them promising candidates for implementation on near-term quantum devices.

DOI: 10.1103/67xf-zdjb

I. INTRODUCTION

A quantum error correcting code (QECC) is a strategy for redundantly encoding quantum information in a way that protects this information from environmental noise. The most commonly studied encodings use the collective state of a large number of qubits (n) to encode the state of a smaller number of qubits (k) so that one encoded state cannot be transformed into another without interacting with at least d of these qubits [1]. Due to the extreme fragility of quantum systems, QECCs have immense potential value in information-processing tasks, but their inherent inefficiency can also lead to significantly increased resource costs. In recent years much progress has been made in reducing these overheads, most notably with the development of so-called "good" quantum low-density parity-check (LDPC) codes [2–5] which have k and d both scaling linearly with n while also possessing a description in terms of sparse matrices (a desirable property for a number of reasons). While these codes require numbers of physical qubits well beyond the scale of current or near-term quantum computing devices, the techniques developed as part of their construction (in particular, the *lifted product* of classical codes [6]) have been observed also to be highly effective at creating efficient, small-scale codes [7–11].

In this work, we describe a family of quantum codes, which we call quantum radial codes, constructed using the lifted product. The codes have parameters comparable to the bivariate bicycle codes of Ref. [12] while also showing evidence of single-shot decodability under circuit-level noise. Additionally, they have an intuitive visual representation, a canonical basis of logical operators and optimal-length circuits for stabilizer measurement. Each code is defined using a pair of integers (r, s) and has parameters $[\![n, k, d]\!] = [\![2r^2s, 2(r-1)^2, \leq 2s]\!]$ and checks of weight $2r$. The independent tunability of k and d makes these codes highly flexible and potentially suited for a variety of applications. Table I summarizes properties of the specific instances studied in this work, which have check weights of at most eight.

The rest of this manuscript is structured as follows. In Sec. II we describe the classical codes used as input to the lifted product and then in Sec. III we study the quantum codes produced. In Sec. IV we numerically investigate the

*Contact author: t.r.scruby@gmail.com

†Contact author: timo.hillmann@rwth-aachen.de

‡Contact author: joschka@roffe.eu

TABLE I. Code parameters, thresholds, and Z word error rates at physical error rate $p = 10^{-3}$ for the quantum radial codes studied in this work. The net encoding rate counts both data qubits and ancilla qubits used for syndrome readout and for our codes is given by $k/2n$. The value of the Z word error rate for the $[\![352, 18, 20]\!]$ is obtained through a fit, see Fig. 9(b).

$[\![n, k, d]\!]$	Net encoding rate	Pseudo threshold	$p_Z(10^{-3})$
$[\![90, 8, 10]\!]$	2/45	0.4%	$\approx 6 \times 10^{-6}$
$[\![352, 18, 20]\!]$	$\approx 1/40$	0.6%	$\approx 3 \times 10^{-8}$

distances of these codes and discuss methods for ensuring higher average-case distance. Finally, in Sec. V we describe our techniques for simulating the performance of these codes under circuit-level noise and present evidence for their single-shot decodability in this setting. The scripts used to obtain all the numerical results in this work can be found in open source at Ref. [13].

II. CLASSICAL RADIAL CODES

We begin by defining what we call *classical radial codes*, which are a subset of the quasicyclic codes presented in Ref. [14]. These codes are defined using a pair of integers (r, s) and have parity-check matrices (PCMs) of the form

$$H = \begin{pmatrix} \lambda_s^{a_{(0,0)}} & \lambda_s^{a_{(0,1)}} & \cdots & \lambda_s^{a_{(0,r-1)}} \\ \lambda_s^{a_{(1,0)}} & \lambda_s^{a_{(1,1)}} & \cdots & \lambda_s^{a_{(1,r-1)}} \\ \vdots & \vdots & & \vdots \\ \lambda_s^{a_{(r-1,0)}} & \lambda_s^{a_{(r-1,1)}} & \cdots & \lambda_s^{a_{(r-1,r-1)}} \end{pmatrix}, \quad (1)$$

where λ_s^a is an element of the ring of circulants of length s. λ_s^a has a binary matrix representation as the $s \times s$ identity matrix with each row shifted to the right by a places, which we write as $\mathfrak{B}(\lambda_s^a)$. We will also use $\mathfrak{B}(H)$ to mean the matrix obtained by replacing each element of H with its binary representative. The code can also be described more concisely using just a matrix

$$A = \begin{pmatrix} a_{(0,0)} & a_{(0,1)} & \cdots & a_{(0,r-1)} \\ a_{(1,0)} & a_{(1,1)} & \cdots & a_{(1,r-1)} \\ \vdots & \vdots & & \vdots \\ a_{(r-1,0)} & a_{(r-1,1)} & \cdots & a_{(r-1,r-1)} \end{pmatrix}_s. \quad (2)$$

It is known that provided the elements of A satisfy

$$a_{(u_1,u_1')} - a_{(u_1,u_2')} - a_{(u_2,u_1')} + a_{(u_2,u_2')} \neq 0 \mod s, \quad (3)$$

the girth (minimum cycle length) in the Tanner graph of the associated code is at least six [14]. In addition, we will require that

(1) s is prime,
(2) $r \leq s$,
(3) the rows of A are linearly independent.

The motivation for these restrictions will be explained shortly, but first, it will be helpful to introduce a visual description of the structure of these codes.

For any code with parity-check matrix of the form in Eq. (1) we can arrange the bits and checks in a pattern of r concentric rings containing s spokes. This is done by enumerating the rings and spokes and then assigning to every row i of $\mathfrak{B}(H)$ a coordinate (u, v) where

$$\begin{aligned} u &= \text{ring number} = \lfloor i/s \rfloor, \\ v &= \text{spoke number} = i \mod s \end{aligned} \quad (4)$$

(where $\lfloor \cdot \rfloor$ is the floor operator) and similarly for the columns. This gives us one bit and one check for each independent coordinate $(0 \leq x < r, 0 \leq y < s)$. Each ring then contains s bits and s checks (one pair from each spoke) and each spoke contains r bits and r checks (one pair from each ring). When arranged in this way each check in the code contains exactly one bit from each ring and each bit is part of exactly one check from each ring. For clarity, we will always use u to refer to ring indices and v to refer to spoke indices. Additionally, we will use primed indices for bit coordinates and unprimed indices for check coordinates. Two examples are shown in Fig. 1, that is, in Fig. 1(a) we have a code

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}_3 \qquad \mathfrak{B}(H) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$
$$H = \begin{pmatrix} \lambda_3^0 & \lambda_3^0 \\ \lambda_3^0 & \lambda_3^0 \end{pmatrix} \quad (5)$$

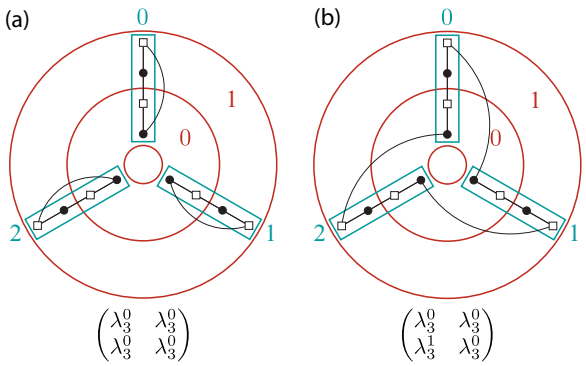


FIG. 1. Radial layouts for two examples of $(r, s) = (2, 3)$ classical codes with parity-check matrices given in Eqs. (5) and (6) respectively. Bits/checks are shown as circles/squares. Rings (and ring indices) are shown in red. Spokes (and spoke indices) are shown in blue.

where each spoke is a copy of the 2-bit cyclic repetition code, while in Fig. 1(b) we have a code

$$A = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}_3 \qquad \mathfrak{B}(H) = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$
$$H = \begin{pmatrix} \lambda_3^0 & \lambda_3^0 \\ \lambda_3^1 & \lambda_3^0 \end{pmatrix} \tag{6}$$

with connections between the spokes, resulting in a 6-bit cyclic repetition code.

By examining the parity-check matrices of these codes more closely we can see that the values $a_{u,u'}$ can be interpreted as a description of the connections between checks of ring u and bits of ring u'. Specifically, if $a_{u,u'} = x$ then each check of ring u is connected to the bit of ring u' that is x spokes ahead of it. When all $a_{u,u'} = 0$ [as in Fig. 1(a)] each check connects only to bits in its own spoke. Note that because the values of $a_{u,u'}$ depend only on the ring coordinates of the relevant checks and bits this representation of the Tanner graph is rotationally symmetric with order s.

We can now use this perspective to understand the significance of the conditions described above, starting with a derivation of Eq. (3). Consider a bit with coordinates (u_1', v_1') and two checks connected to this bit, which must have coordinates $(u_1, v_1' - a_{u_1,u_1'})$ and $(u_2, v_1' - a_{u_2,u_1'})$, with addition implicitly performed mod s. Under what circumstances can these checks also share a second bit, i.e., resulting in a length-4 cycle? The first check is connected to bits with coordinates $(u_2', (v_1' - a_{(u_1,u_1')}) + a_{(u_1,u_2')})$ and the second to bits with coordinates $(u_3', (v_1' - a_{(u_2,u_1')}) + a_{(u_2,u_3')})$. For two of these bits to be the same we need the ring coordinates to be equal, $u_2' = u_3'$, and also the spoke coordinates to be equal

$$v_1' - a_{(u_1,u_1')} + a_{(u_1,u_2')} = v_1' - a_{(u_2,u_1')} + a_{(u_2,u_2')}. \tag{7}$$

By requiring that this condition is not met for any u_1, u_2, u_1', u_2' we obtain Eq. (3) and guarantee a certain amount of expansion in the Tanner graph, since checks can only share at most one bit. To see an example, compare the Tanner graphs of Fig. 1(a), where A does not satisfy (3), and Fig. 1(b), where it does. With this in mind, the other three restrictions have the following effects.

s is prime: As mentioned above, the radial representation of the Tanner graph is rotationally symmetric with order s, and so any disjoint subgraphs contained in the graph must also have rotational symmetry with order that is some factor of s. By choosing s to be prime we thus ensure that all rotational symmetries of the graph are of order s or 1, and when combined with (3) this ensures that the Tanner graph consists of a single connected component. This is because a disjoint subgraph with order s symmetry (that is not just the entire graph) must contain every bit and check in some subset of the rings and no bits or checks from other rings, but because each bit/check is connected to one check/bit from each ring such a subgraph is not possible. On the other hand, if there is a disjoint subgraph with order 1 symmetry then there must be s disjoint copies of this subgraph which are mapped onto each other by the order s symmetry of the total graph. Each of these subgraphs can then contain at most r unique bit nodes (as there are only rs bits in the code), but each bit is connected to r checks and by Eq. (3) these checks are connected to $r(r-1)$ unique bits so there can be no disjoint subgraphs of this size. The only remaining option is for there to be no nontrivial disjoint subgraphs.

$r \leq s$: This is necessary because the r checks which share a given bit cannot have any other shared bits and must also be connected to one bit from each ring, so there must be at least r unique bits in each ring. We can also see that the distance of the code must be at least r, as any nonzero bit of a given codeword is connected to r checks and each of these must be connected to at least one other nonzero bit of this codeword for the checks to be satisfied.

The rows of A are linearly independent: Given any row H_i of H, the sum of all the rows of $\mathfrak{B}(H_i)$ is the all-1s vector. This means that there is one linear dependency in $\mathfrak{B}(H)$ for each pair of rows in H, giving $r-1$ linear dependencies. If we then choose the rows of A to be linearly independent then we find that with high probability [15] these are the only linear dependencies. We then have rs bits and $rs - (r-1)$ independent checks, giving $(r-1)$ encoded bits of information.

Finally, we can choose a basis for the codewords of these codes and show that the distance is $2s$. Enumerating the encoded bits from 0 to $r-2$, let the ith codeword correspond to setting all bits in the ith ring and the $(r-1)$th ring to be 1, and all other bits to be 0 (this is a codeword because each check in the code acts on exactly one bit from each ring). These $r-1$ codewords are all linearly independent and of weight $2s$, and because they overlap only on the $(r-1)$th ring, and this overlap always has size s, any linear combination of them always has weight at least $2s$. These codes then have parameters $[rs, r-1, 2s]$, with each node in the Tanner graph having degree r.

III. QUANTUM RADIAL CODES

Quantum radial codes (QRCs) are obtained from the lifted product of two classical radial codes. For a (ring-theoretic) ring R and two ring-valued matrices $H_1 \in R^{m_1 \times n_1}$ and $H_2 \in R^{m_2 \times n_2}$ the lifted product is defined as

$$H_X = \left[H_1 \otimes I_{m_2} | I_{m_1} \otimes H_2\right],$$
$$H_Z = \left[I_{n_1} \otimes H_2^* | H_1^* \otimes I_{n_2}\right], \tag{8}$$

where H^* is the conjugate transpose of H, obtained by transposing the matrix and taking the inverse of each element [6]. If H is a classical radial code then H^* will also be a classical radial code (in fact, it is just H with the bits and checks exchanged).

A. A small example

To understand the structure of QRCs we can start with a simple example, taking [from Fig. 1(b)]

$$H_1 = H_2 = H = \begin{pmatrix} \lambda_3^0 & \lambda_3^0 \\ \lambda_3^1 & \lambda_3^0 \end{pmatrix}, \tag{9}$$

which has conjugate transpose

$$H_1^* = H_2^* = H^* = \begin{pmatrix} \lambda_3^0 & \lambda_3^2 \\ \lambda_3^0 & \lambda_3^0 \end{pmatrix} \tag{10}$$

so that we obtain a quantum code with parity-check matrices

$$H_X = \left(\begin{array}{cccc|cccc|c} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \\ \lambda_3^0 & 0 & \lambda_3^0 & 0 & \lambda_3^0 & \lambda_3^0 & 0 & 0 & 0 \\ 0 & \lambda_3^0 & 0 & \lambda_3^0 & \lambda_3^1 & \lambda_3^0 & 0 & 0 & 1 \\ \lambda_3^1 & 0 & \lambda_3^0 & 0 & 0 & 0 & \lambda_3^0 & \lambda_3^0 & 0 \\ 0 & \lambda_3^1 & 0 & \lambda_3^0 & 0 & 0 & \lambda_3^1 & \lambda_3^0 & 1 \end{array}\right) \tag{11}$$

$$H_Z = \left(\begin{array}{cccc|cccc|c} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \\ \lambda_3^0 & \lambda_3^2 & 0 & 0 & \lambda_3^0 & 0 & \lambda_3^2 & 0 & 0 \\ \lambda_3^0 & \lambda_3^0 & 0 & 0 & 0 & \lambda_3^0 & 0 & \lambda_3^2 & 1 \\ 0 & 0 & \lambda_3^0 & \lambda_3^2 & \lambda_3^0 & 0 & \lambda_3^0 & 0 & 0 \\ 0 & 0 & \lambda_3^0 & \lambda_3^0 & 0 & \lambda_3^0 & 0 & \lambda_3^0 & 1 \end{array}\right). \tag{12}$$

Readers may find it helpful to refer to the full binary representations of these matrices (Appendix A) during the following discussion.

We can think of the quantum code as being built from two copies of H (orange and green) and two copies of H^* (red and blue). We will call the copies of H "X codes" and the copies of H^* "Z codes." Each qubit is associated with a bit in one of these four codes, while each stabilizer is associated with a check (X stabilizers with the checks of the X codes and Z stabilizers with those of the Z codes). The colored row/column indices show the associated classical code and ring within that code, of each qubit and stabilizer in the quantum code.

The support of each X stabilizer of the quantum code consists of the support of a ring u check in one of the X codes plus one qubit from ring u in each of the Z codes.

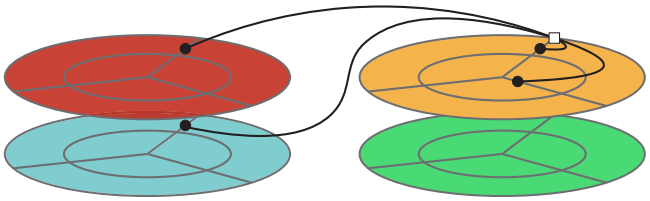


FIG. 2. Structure of the quantum radial code with PCMs (11) and (12). The code is made of four $(r, s) = (2, 3)$ classical radial codes, with two (red and blue) providing Z stabilizers and two (orange and green) providing X stabilizers. An example of a X stabilizer from ring 1 of the orange code (supported on a qubit from each ring of the orange code and a ring 1 qubit in each of the red and blue codes) is also shown.

Importantly, each stabilizer in ring u of a given X code is assigned a *unique* qubit from ring u of each Z code so that the product of all X stabilizers in any ring of an X code is supported on all qubits of that X code plus all ring u qubits in all Z codes. Similarly, each Z stabilizer has the support of a ring u check from a classical Z code plus one qubit from ring u in each of the X codes, and the assignments of these qubits are unique in the same way. All checks have weight 4, because they act on two X code qubits and two Z code qubits. A representation of this structure is shown in Fig. 2.

To see how many logicals we should expect we can count the number of relations between stabilizer generators. As explained above, the product of all X stabilizers from ring u of the orange code is supported on all qubits of the orange code and all ring u qubits in the red and blue codes. The product of all stabilizers from both rings of the orange code is then supported on all qubits of the red and blue codes and none of the qubits of the orange code. The same is true for the stabilizers of the green code so the product of all X stabilizers is identity on all qubits. The same argument can be made for the Z stabilizers. These are the only relations so we have only two encoded qubits.

We can also use the radial structure of the code to find a basis for these logical operators. We can see that any classical codeword from one of the Z codes lifts to a logical X operator in the quantum code because if this operator is supported only on, e.g., red qubits then it only intersects Z stabilizers associated with the checks of the red code, and by definition, it commutes with these stabilizers because its support is a codeword of the red code. We know that a basis for the codewords of a classical radial code is all bits in any pair of adjacent rings, and here we only have two rings so the logical operator is supported on all qubits of the red code. A similar operator is supported on all qubits of the blue code but these two are equivalent up to composition with stabilizers. There are also logical X operators supported on all ring u' qubits of the orange and green codes. We can see this because each Z stabilizer acts on one qubit in each of these rings, so it will intersect two qubits of this

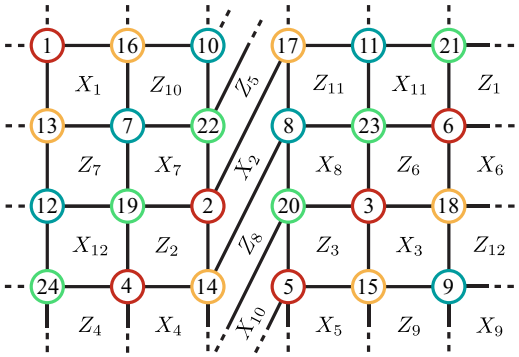


FIG. 3. Embedding of the radial code described by Eqs. (11) and (12) onto a twisted 2-torus, where it is equivalent to a surface code. Qubits are circles with coloring showing which of the classical radial codes in Fig. 2 they belong to. X and Z stabilizers are square faces.

operator (one orange and one green). The operators associated with $u' = 0$ and $u' = 1$ are once again equivalent up to stabilizers. Finally, we can identify a pair of logical Z operators, one supported on the X codes and one on the Z codes, in the same way. This is consistent with the fact that we expected two encoded qubits from counting relations and gives a basis of logical operators with weight $2s = 6$.

Before generalizing these results we observe that this example of a radial code has a different (and more familiar) representation as a surface code on a twisted torus (Fig. 3). Readers who wish to check this explicitly can refer to the binary PCMs in Appendix A. This makes it obvious that this code is actually distance 4 rather than 6. We will see later that this is not specific to this case, and that while we can generalize the logical basis described here the resulting upper bound on the distance is usually not tight.

B. The general case

A general QRC is built from r copies of H_1 and r copies of H_2^*. Each qubit and stabilizer is assigned a coordinate (c, u, v) where $0 \leq c < 2r$ indexes the code to which it belongs. We will also use the indices $0 \leq x, z < r$ to index X and Z codes separately.

H_X for a general QRC has the form

$$H_X = (H_X^{(Z)} \mid H_X^{(X)}), \tag{13}$$

where

$$H_X^{(Z)} = \begin{pmatrix} H_1^{(0,0)} I_r & H_1^{(0,1)} I_r & \cdots & H_1^{(0,r-1)} I_r \\ H_1^{(1,0)} I_r & H_1^{(1,1)} I_r & \cdots & H_1^{(1,r-1)} I_r \\ \vdots & \vdots & & \vdots \\ H_1^{(r-1,0)} I_r & H_1^{(r-1,1)} I_r & \cdots & H_1^{(r-1,r-1)} I_r \end{pmatrix} \tag{14}$$

describes the connections of X stabilizers to qubits of the Z codes (I_r is the $r \times r$ identity matrix) and

$$H_X^{(X)} = \begin{pmatrix} H_2 & 0 & \cdots & 0 \\ 0 & H_2 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & H_2 \end{pmatrix} \tag{15}$$

describes the connections to qubits of the X codes. In this block form each column of $H_X^{(Z)}/H_X^{(X)}$ is associated with all qubits of a Z/X code while each row is associated with all X stabilizers of an X code. We can then see that the support of an arbitrary X stabilizer from X code x, ring u, will be a single qubit from each ring of this X code (the same support as a check of H_2) and a single qubit from ring u of each Z code. This second point is due to the fact that each $H_1^{(i,j)} I_r$ can be expanded as

$$\begin{aligned} H_1^{(i,j)} I_r &= \lambda_s^{a^1_{(i,j)}} I_r \\ &= \begin{pmatrix} \lambda_s^{a^1_{(i,j)}} & 0 & \cdots & 0 \\ 0 & \lambda_s^{a^1_{(i,j)}} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \lambda_s^{a^1_{(i,j)}} \end{pmatrix}, \end{aligned} \tag{16}$$

where each row u is associated with all checks from the uth ring of X code x and each column u' is associated with all qubits from the u'th ring of Z code z. Because the matrix is diagonal X checks from ring u can only connect to Z code qubits from ring u, and because $\lambda_s^{a^1_{(i,j)}}$ has exactly one nonzero entry in each row and column each X check is connected to exactly one qubit from each Z code, and no two X stabilizers from the same X code x and ring u share a qubit in Z code z.

An equivalent analysis can be performed for Z stabilizers and so we obtain a direct generalization of the structure discussed in the previous section, namely that an X/Z stabilizer from code c, ring u is connected to a unique qubit in each ring of code c and a unique ring u qubit in each Z/X code. An illustration of this generalized structure is shown in Fig. 4.

The rest of the discussion in the previous section generalizes straightforwardly as well. Each codeword of a Z code (all bits in rings u',$u'+1$ of that code) lifts to a logical X operator of the quantum code and there are $r-1$ such independent logicals per Z code. The product of all X stabilizers from rings u,$u+1$ in any X code is supported

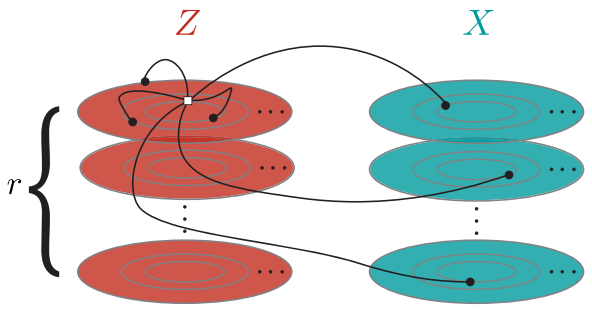


FIG. 4. Representation of the structure of a quantum radial code. The code is formed from r copies of a classical radial code H_1 (the X codes) and r copies of a classical radial code H_2^* (the Z codes). Each stabilizer is associated with a ring u check in one of these classical codes and has support on a qubit from each ring of that code and a ring u qubit from each code of the other type ($u = 0$ in the example above).

on all ring u and $u+1$ qubits of all Z codes, so there are $r-1$ independent choices of Z code and $(r-1)^2$ X logicals of this type. There are also logical X operators supported on all ring u qubits in a pair of X codes x and $x+1$. There are r different X codes giving $(r-1)$ independent pairs of codes for each choice of ring, and the product of all X stabilizers from any ring in X codes x and $x+1$ is supported on all qubits of these X codes, meaning we have $r-1$ independent choices of ring and $(r-1)^2$ independent logicals of this type also. An identical analysis can be performed for the logical Z operators, so we have $2(r-1)^2$ independent X and Z operators and $2(r-1)^2$ encoded qubits. The number of encoded qubits could also be checked by counting the number of relations between stabilizers, as was done in the previous example.

Finally, we observe that every logical operator in the basis described above is supported on a pair of rings, and that every ring contains s qubits, so each of these operators has weight $2s$. The number of physical qubits in the code is $2r^2s$ (as it is made of $2r$ classical radial codes, each of length rs) and so a general QRC has parameters $[\![2r^2s, 2(r-1)^2, \leq 2s]\!]$. We were unable to identify a method to reliably achieve this upper bound on the distance, but in the next section we study the scaling of d with s in the average case and present numerical evidence that the relation is linear. In a similar vein, it would be preferable if the check weights could be made constant rather than scaling as $2r$, but as the majority of the arguments provided above rely on the checks of the classical codes being supported on one bit of each ring this is not possible without significantly altering the construction. However, in this work we are only concerned with small-size instances of these codes and in these cases the stabilizer weights are no more than eight.

IV. CODE DISTANCES OF QUANTUM RADIAL CODES

As shown in the example of Fig. 3, the upper bound on the distance of $d \leq 2s$ that we obtained previously is not generally tight, and in most cases lower distance logical operators will exist. In Appendix B we examine this specific example in more detail and conclude that QRCs obtained from products of classical codes where $H_1 = H_2$ are more likely to exhibit low-weight logicals than those where $H_1 \neq H_2$. To test this hypothesis and to get a better idea of what kinds of distances we can expect from QRCs we use QDistRnd [16] to numerically estimate the distances of randomly generated codes for different r and s and for both $H_1 = H_2$ and $H_1 \neq H_2$ (i.e., independently generated H_1 and H_2). The accuracy of these distance estimates is discussed in Appendix C.

We also wish to emphasize that the instances we have studied here are relatively small and there is no guarantee that this linear behavior will continue to hold for larger codes. We note that, to the best of our knowledge, proving minimum distance bounds of lifted product codes over an Abelian group remains an open problem. While certain examples of bounds exist [6], these rely upon the input

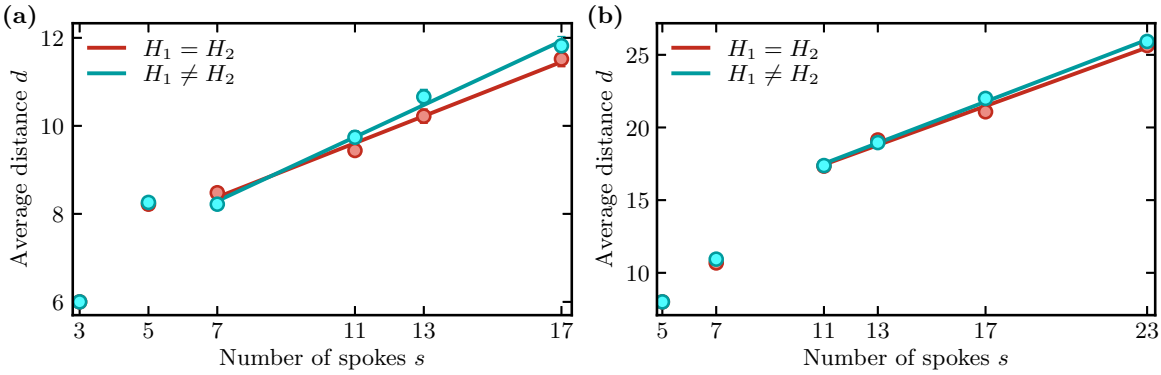


FIG. 5. Numerical estimates of average distances for quantum radial codes of different s for (a) $r = 3$ and (b) $r = 4$. 100 codes were generated for each pair (r, s). Error bars are the standard error on the mean and are typically smaller than the marker.

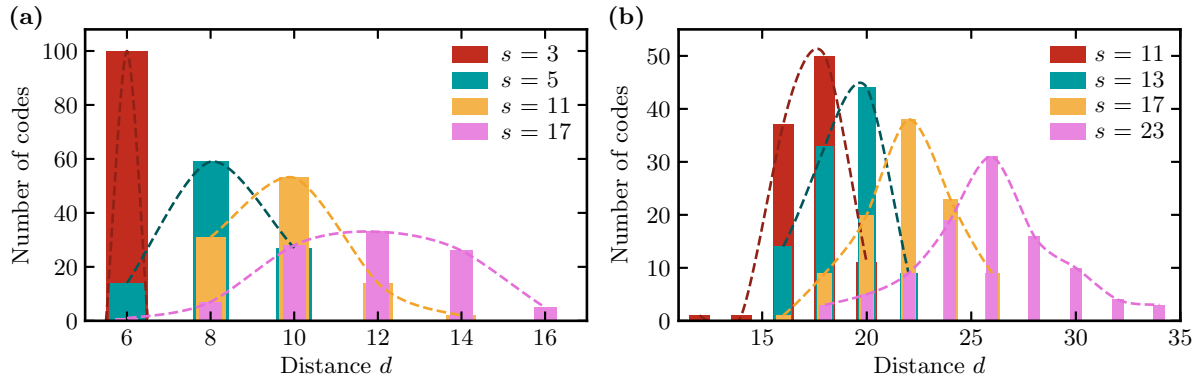


FIG. 6. Distribution of distances for quantum radial codes of different s for (a) $r = 3$ and (b) $r = 4$. The codes in these figures are a subset of the $H_1 \neq H_2$ codes from Fig. 5. Curves are intended to improve clarity where different distributions overlap and do not imply fitting to a model.

matrices having specific structures that do not extend to the radial code construction.

Additionally, we study the distributions of these distances around the average, as shown in Fig. 6. We observe that these distances appear to be normally distributed about the mean, with the variance increasing with increasing s. This is unsurprising, as there is a greater variety of possible classical radial codes to use as input when s is much larger than r.

A. Confinement properties

In addition to the code distance, the confinement [17] of a code becomes relevant in the presence of syndrome measurement errors for single-shot decoding. Formally, for a stabilizer code confinement is defined as

Definition 1 (Confinement, [17]). 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 $|e| \leq t$, it holds

$$f(|\sigma(e)|) \geq |e|^{\text{red}}. \tag{17}$$

In the above definition, $\sigma(\bullet)$ refers to the syndrome map and the superscript red refers to the condition that the weight of e must be reduced over the stabilizer group. Practically, we can characterize the confinement profile of a stabilizer code by searching for the minimum syndrome weights of errors of increasing weight $w \geq 1$ [18]. We use the software `dist-m4ri` [19] to compute the confinement profile of two specific examples of quantum radial codes up to $w \leq 8$ and compare it to the confinement profile of the bivariate bicycle codes in Ref. [12]. We also compare to the trivariate tricycle codes studied in Ref. [20]. We restrict to H_Z checks for all codes and note that trivariate tricycle codes have meta-checks in this sector. The results are summarized in Table II.

While our $[\![90, 8, 10]\!]$ quantum radial code has a confinement profile similar to the bivariate bicycle codes, nondecreasing only for irreducible errors of weight $w \leq 2$, our $[\![352, 18, 20]\!]$ code has improved confinement. Notably, the $[\![352, 18, 20]\!]$ quantum radial code satisfies $|\sigma(e)| \geq |e|^{\text{red}}$ for irreducible errors with weight $w \leq 6$. In that, it is more similar to examples of trivariate tricycle codes. While the syndrome that an error of weight w produces is larger, the condition $|\sigma(e)| \geq |e|^{\text{red}}$ is satisfied for irreducible errors with weight $w \leq 7$, $w \leq 5$, and $w \leq 6$ for the three examples in Table II.

In addition to the minimal syndrome weight for an irreducible error of weight w, we also show the average syndrome weight for an irreducible error of weight w in Fig. 7. Note that the average syndrome weight is typically nondecreasing up to larger values of w compared to the minimal syndrome weight in Table II.

TABLE II. Confinement profile for X errors (Z checks) for various examples of bivariate bicycle [12], quantum radial, and trivariate tricycle [20] codes. The confinement profile is obtained from the minimal syndrome weight of errors with weight $w = 1, \ldots, 8$ for all codes except the $[\![72, 12, 6]\!]$ (BB), where $w = 1, \ldots, 6$.

Code	Confinement profile
$[\![72, 12, 6]\!]$ (BB)	3, 4, 3, 4, 3, 4
$[\![90, 8, 10]\!]$ (BB)	3, 4, 3, 2, 3, 2, 3, 2
$[\![108, 8, 10]\!]$ (BB)	3, 4, 3, 2, 3, 2, 3, 2
$[\![144, 12, 12]\!]$ (BB)	3, 4, 3, 2, 3, 2, 3, 2
$[\![288, 12, 18]\!]$ (BB)	3, 4, 3, 2, 3, 4, 3, 4
$[\![360, 12, \leq 24]\!]$ (BB)	3, 4, 3, 2, 3, 4, 3, 4
$[\![90, 8, 10]\!]$ (QR)	3, 4, 3, 2, 3, 2, 3, 2
$[\![352, 18, 20]\!]$ (QR)	4, 6, 6, 6, 6, 6, 4, 4
$[\![72, 6, 6]\!]$ (TT)	6, 8, 10, 10, 10, 10, 10, 6
$[\![126, 6, 8]\!]$ (TT)	6, 8, 10, 12, 14, 12, 12, 6
$[\![288, 6, 10]\!]$ (TT)	6, 8, 10, 12, 14, 14, 12, 6

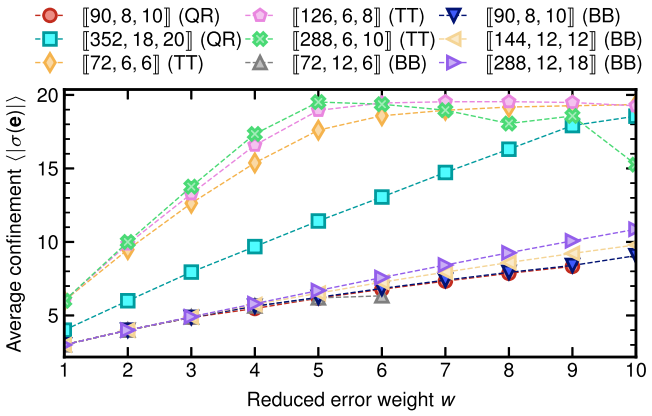


FIG. 7. Average confinement profile for X errors (Z checks) for various codes. It represents the average syndrome weight of irreducible errors of weight w. Dashed lines are a guide for the eye.

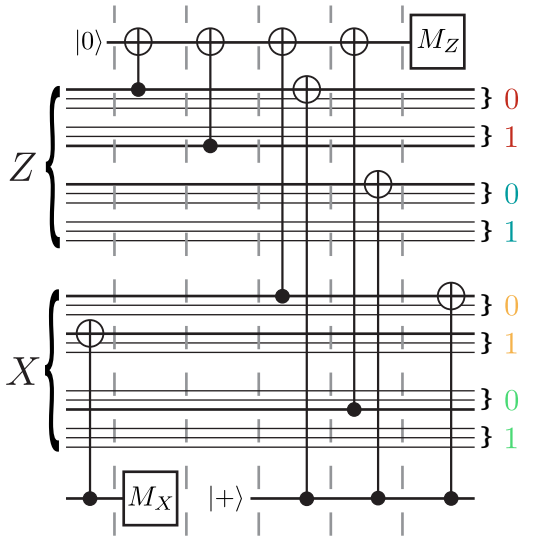


FIG. 8. Measurement circuit for two stabilizers from the $(2, 3)$ quantum radial code discussed in Sec. III A. The top qubit is the ancilla for the ring 0, spoke 0 Z stabilizer from the red classical code while the bottom qubit is the ancilla for the ring 0, spoke 0 X stabilizer from the orange classical code. The 24 qubits in the middle are the 24 data qubits of the quantum code and are ordered using their associated classical codes and rings as before. Bold qubit lines are part of the support of one or both stabilizers. Dashed gray lines show boundaries between timesteps.

Last, it is worth noting that the confinement profile for the *code* is not the same thing as the confinement profile for the *detector error model* (DEM) on which the decoding is performed (see Sec. V), although the two will be related. One important difference is that, in any circuit consisting of repeated measurements of the same set of stabilizers, a sequence of w consecutive measurement errors on the same ancilla qubit always results in a syndrome of weight two, regardless of the size of w.

V. SINGLE-SHOT DECODING UNDER CIRCUIT-LEVEL NOISE

A. Syndrome extraction circuits

We can use the previously described structure of QRCs to define an optimal-length (in the sense that ancilla qubits are never idle) schedule for parallelized stabilizer measurements. In this schedule, each stabilizer ancilla interacts first with its associated Z code qubits and then with its associated X code qubits, with X and Z stabilizers offset from each other by half a cycle. An example for two stabilizers of the example code from Sec. III A is shown in Fig. 8.

The general definition of such a measurement circuit is as follows. For a Z measurement ancilla associated to a stabilizer in Z code z and ring u the associated operation at each timestep of the circuit is

(1) $t = -1$: Initialize the ancilla in $|0\rangle$.
(2) $0 \leq t < r$: Apply a CX gate with the target on the ancilla and control on the relevant qubit.
(3) $r \leq t < 2r$: Apply a CX gate with the target on the ancilla and control on the relevant qubit in X code $(z + t) \mod r$, ring i.
(4) $t = 2r$: Measure the qubit in the Z basis.

For an X stabilizer measurement ancilla in X code x, ring u we instead have

(1) $t = r - 1$: Initialize the ancilla in $|+\rangle$.
(2) $r \leq t < 2r$: Apply a CX gate with control on the ancilla and target on the relevant qubit in Z code $(x + t) \mod r$, ring i.
(3) $2r \leq t < 3r$: Apply a CX gate with control on the ancilla and target on the relevant qubit in X code x, ring $(u + t) \mod r$.
(4) $t = 3r$: Measure the qubit in the X basis.

Because each Z code z, ring u data qubit is part of a unique stabilizer from each ring of Z code z and a unique ring u stabilizer from each X code (and the reverse for X code qubits) we can see that this schedule will never lead to collisions where two ancilla qubits try to interact with the same data qubit at the same time. Additionally, because each data qubit interacts first with all its associated Z measurement ancillas and then with all its associated X measurement ancillas the circuit is guaranteed to be valid and avoid the issues discussed in Appendix B of Ref. [21]. Finally, we can see that if we run the two halves of this circuit continuously (i.e., Z ancillas are initialized on each $t \mod 2r = -1$ and X ancillas on each $t \mod 2r = r - 1$) then these ancillas are never idle and each cycle of the circuit has the minimal possible length.

It is worth noting that this circuit is not guaranteed to be optimal from a fault-tolerance perspective, and there may be other measurement circuits for QRCs that lead to higher effective distance. We leave the study of such circuits as a problem for future work.

B. Simulation method and noise model

We use `stim` [22] to simulate the propagation of noise through these measurement circuits. We use a standard noise model where for each timestep,

(1) idle qubits experience single-qubit depolarizing errors with probability p_{idle},
(2) pairs of qubits acted on by CX gates experience two-qubit depolarizing errors immediately following the gate with probability p_{CX},
(3) qubits initialized in $|0\rangle$ / $|+\rangle$ experience single-qubit X/Z errors immediately following initialization with probability p_{reset},
(4) qubits measured in the Z/X basis experience single-qubit X/Z errors immediately before the measurement with probability p_{meas}.

We restrict our simulations to the setting where $p_{\text{idle}} = p_{\text{CX}} = p_{\text{reset}} = p_{\text{meas}}$ to facilitate comparison with other works. However, it is worth noting that such a setting is not necessarily experimentally realistic, and that in practice, e.g., idling errors may be negligible [10] or that measurements error rates may be much higher than two-qubit gate error rates [23].

C. Decoding strategy

The output from `stim` is a DEM—a bipartite graph of error nodes, corresponding to possible faults in the circuit, and detector nodes, corresponding to sets of measurement outcomes that can be used to infer the occurrence of these faults. An error node and a detector node are connected by an edge whenever the occurrence of the error modifies the outcome of the detector.

Under a phenomenological noise model, the standard approach to simulating single-shot decoding is to apply random Pauli errors to the qubits of the code, calculate the corresponding syndrome, flip random bits of this syndrome to represent the action of measurement errors, and then task the decoder to calculate a correction based on this syndrome [24]. Usually, there will be some discrepancy between the calculated and true error configurations, resulting in some residual error that is carried over into the next cycle of the simulation. This approach does not generalize directly to the setting of `stim` and DEMs where error configurations are generated for the entire circuit before any decoding is performed, with some errors triggering multiple detectors with different time coordinates. To represent the action of a single-shot decoding strategy in this setting we must therefore have a way to divide the DEM into distinct time slices which can be fed independently to the decoder and to propagate the calculated corrections forward in time so that future detector outcomes are appropriately modified [25].

In our work, we use an overlapping window [26–28] (sometimes referred to as a sliding window [29,30]) decoding approach that fulfills these requirements. In general, a (w, c)-overlapping window decoder is defined by a choice of decoding algorithm, window size $w \in \mathbb{N}$, and commit size $c \leq w \in \mathbb{N}$. The operation of the decoder is as follows. Consider a generic decoding cycle, before which a correction for the syndrome data from the first t syndrome extraction cycles has already been determined. w and c are then used to define two sets of consecutive syndrome extraction cycles, the *decoding window*, $\{t+1, t+2, \ldots, t+w\}$, and the *commit region*, $\{t+1, t+2, \ldots, t+c\}$. In each decoding cycle, the subset of the total syndrome associated with the extraction cycles within the decoding window is identified and passed to the decoder, and a correction is returned. The parts of this correction that lie outside the commit region are then discarded and only the correction within the commit region is applied (either in hardware or software). Finally, the syndrome of this modified correction is calculated and used to update the existing syndrome data for future extraction cycles. The decoding window is then advanced by c syndrome extraction cycles and the process begins again.

Algorithm 1 shows an implementation of this procedure based on the DEM obtained from `stim`. In each decoding cycle, the subgraph of the DEM lying within the decoding window is calculated using a call to current_inds. We make use of the fact that in a standard stabilizer code potential faults can only trigger detectors locally, that is, a fault during a syndrome extraction cycle can only influence detectors during that and the following cycle [31]. As a result, the matrix representation of the DEM is a banded matrix, which allows one to determine relevant indices for the current decoding and commit region by identifying the indices of the first and last nonzero elements for a subset of rows that corresponds to the detectors of the current decoding round.

The most direct generalization of single-shot decoding to this overlapping window case would be to take $w = c = 1$, but we can also consider looser generalizations where w and c are simply required to be both small and independent of the size or distance of the code. We propose "constant-depth decodable" as an alternative and more precise term for "single-shot decodable" in this case.

D. Numerical results

For the results presented in this section, and summarized in Fig. 9, we use the BP + OSD decoder of the *ldpc* Python

ALGORITHM 1. Pseudocode for overlapping window decoding.

```
Data: DEM, priors, syndrome s, window w,
      commit c, n_s
Result: total_corr
total_corr = zeros(size(DEM, 1))
for r in range(decodings) do
    c_inds, w_inds, s_c_inds, s_w_inds =
     current_inds(r, DEM, w, c, n_s)
    /* Get indices for slicing up DEM and
       syndrome                          */
    DEM_r = DEM[s_w_inds, :]
    decoder = init_decoder(DEM_r, priors)
    corr = decoder.decode(s[s_w_inds])
    if r ≠ decodings-1 then
        total_corr[c_inds] += corr[c_inds]
        /* Propagate correction to next
           round                         */
        s += DEM * total_corr
    else
        total_corr[w_inds] += corr[w_inds]
    /* Adjust priors such that the
       decoder does not correct these
       faults again                      */
    priors[c_inds] = decoder.perf_prior
return total_corr
```

package [32,33] in combination with a $(w, c) = (3, 1)$-overlapping window decoding strategy, although a $(1, 1)$ strategy was also observed to suggest a threshold, albeit with higher word error rates. For each window, we perform a maximum of 1000 iterations of the min-sum algorithm with a parallel schedule. If the decoder does not converge, the soft output of the decoder is used to inform the ordered statistics subroutine. We emphasize that these parameters are nonoptimal and that further increasing, for example, the maximum number of iterations for the min-sum algorithm yields additional improvements. Examples of how logical error rates vary with the number of BP iterations are shown in Appendix E. Here, we have limited simulations to 1000 iterations to obtain sufficient statistics in a reasonable time. We remark that, as its name suggests, the parallel schedule can be fully parallelized in practical applications as all necessary operations and calculations can be performed locally. The OSD routine in its current form is not efficiently parallelizable and achieving it is an active research question [34].

For our simulations, we choose a pair of radial codes with parameters $[\![90, 8, 10]\!]$ and $[\![352, 18, 20]\!]$, corresponding to (r, s) values of $(3, 5)$ and $(4, 11)$ respectively. Note that the $[\![90, 8, 10]\!]$ code saturates the distance bound discussed in Sec. IV, while the $[\![352, 18, 20]\!]$ code comes close. In Fig. 9(a) we can see how the Z word error rates in both of these codes vary as we increase the number of simulated decoding cycles under the error model and decoding strategy described above. In these simulations, we fix the physical error rate to be 2×10^{-3} and apply only order-0 OSD reprocessing. X word error rates are not shown here but are identical within the sample variance. We observe that after $\sim$10 cycles the error rate per cycle becomes constant, confirming the validity of our single-shot decoding approach. Further evidence for this can be seen in Fig. 9(b), which shows how the observed Z word error rate per syndrome cycle p_Z varies with the physical error rate p. Here we fix the number of decoding cycles to 15 and apply order-4 reprocessing with the combination-sweep search strategy (OSD-CS-4), and

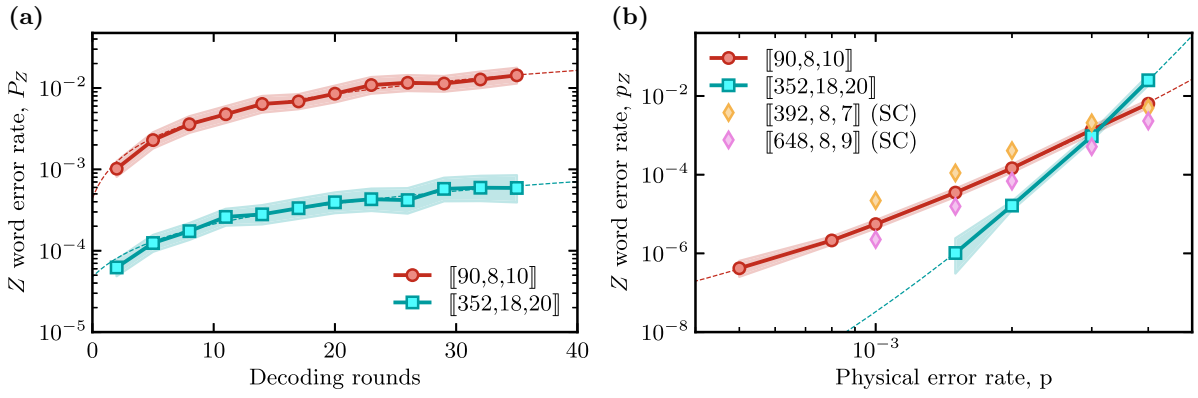


FIG. 9. Word error rates in a pair of radial codes as a function of (a) number of decoding cycles and (b) physical error rate. Simulations were performed using a circuit-level noise model and $(3, 1)$-overlapping window implementation of BP-OSD-0 in (a) and BP-OSD-CS-4 in (b). In (a) the physical error rate is fixed at $p = 2 \times 10^{-3}$ and dashed lines are linear fits showing a linearly increasing error rate per decoding round after a few decoding cycles. In (b) the number of decoding cycles is fixed at 15 and dashed lines are exponential fits with a quadratic exponent, showing exponential error suppression. Note that here we show the Z word error rate per syndrome cycle, in contrast to (a) which showed the total Z word error rate. Shown are also diamond markers for the Z error rate of eight patches of $d = 7$ and $d = 9$ rotated surface codes decoded over $3d$ syndrome cycles. Shading indicates hypotheses whose likelihoods are within a factor of 1000 of the maximum likelihood estimate.

observe exponential suppression of logical error rates with decreasing p. The larger code also outperforms the smaller one for low enough values of p.

It is worth emphasizing that the single-shot decodability of these codes is not merely a curiosity, and that the levels of error suppression displayed make them competitive with some of the best-known codes of similar size, as can be seen in the comparison against the surface code shown in Fig. 9(b). These data for the surface code were obtained from simulations of $3d$ rounds of the `surface_code:rotated_memory_z` experiment in `stim` [22] with physical error rate p for all error types. The noise model here is broadly similar to the one described above, but combines measurement and reset errors into a single error, ignores idling errors, and applies a depolarizing channel before each syndrome cycle. The resulting error rate $P_{Z,1}$ for a single logical qubit is then rescaled according to the expression

$$p_Z = 1 - (1 - P_{Z,1})^{k/3d}, \tag{18}$$

to obtain the Z word error rate per syndrome cycle for $k = 8$ logical qubits.

Additionally, we note that in some cases the parameters and error correction capabilities of the radial codes are comparable to those of the bivariate bicycle codes [12] (e.g., when comparing the $[\![90, 8, 10]\!]$ codes of both constructions), but that no valid single-shot decoding strategy has been demonstrated for the latter. On the other hand, large bivariate bicycle codes appear to significantly outperform large radial codes, perhaps due to the higher check weights of the radial codes and the difference in simulated decoding strategies.

VI. CONCLUSION

We have introduced a family of small quantum LDPC codes with high rate and distance and competitive error suppression capabilities under a single-shot decoding strategy. We proved an exact value for k and an upper bound on d for all codes in this family and showed numerically that while these codes do not achieve this bound in general, random instances can come very close to it with high probability when n is suitably small. In numerical studies of performance under circuit-level noise, small examples of these codes achieved similar levels of error suppression to surface codes of comparable distance while using $\sim$5 times fewer physical qubits. Due to their small size, flexibility and powerful error correction capabilities, we believe these codes would be well suited to small experiments and implementations on near-term hardware, especially on, e.g., neutral atom platforms [35] that naturally allow long-range connectivity.

A number of outstanding questions about these codes exist. First, we note that our results in conjunction with the confinement profile (Table II) suggest that the bivariate bicycle codes of Ref. [12] could also be single-shot decodable. Other open questions include whether more optimal measurement circuits exist for our codes—we have not evaluated the effective distance of the circuits used in this work—and also whether other modifications could be made to the construction to ensure higher distance. It would also be interesting to investigate the viability of different and more efficient decoding strategies in these codes, e.g., fully local or generalized matching decoders.

A more high-level question, given the large number of recent results involving lifted products of small classical codes, is which other classical code families might be suitable candidates for such a study and how these families might be identified. Rather than a brute-force search of the classical literature, it may be possible to identify particular desirable properties of classical codes that lead to useful quantum codes (e.g., single-shot decodability, high distance, etc.).

ACKNOWLEDGMENTS

T.R.S. acknowledges support from the JST Moonshot R&D Grant (Grant No. JPMJMS2061). T.H. acknowledges financial support from the Chalmers Excellence Initiative Nano and the Knut and Alice Wallenberg Foundation through the Wallenberg Centre for Quantum Technology (WACQT). J.R. is funded by an EPSRC Quantum Career Acceleration Fellowship (Grant Code UKRI1224). J.R. further acknowledges support from EPSRC (No. EP/T001062/1), EPSRC (No. EP/X026167/1), BMBF (RealistiQ), and DFG (No. CRC 183). Numerical results presented in this work were obtained using the HPC resources provided by the Scientific Computing and Data Analysis section of the Research Support Division at OIST, and also using resources at the Chalmers Centre for Computational Science and Engineering (C3SE) under Project 2024/1-9. The decoding part of this project resulted from discussions between T.R.S. and T.H. at the 2024 YITP Quantum Error Correction workshop in Kyoto. The authors also acknowledge valuable discussions with Armanda Quintavalle, Boren Gu, Oscar Higgott, Mike Vasmer, and Jens Eisert. The title of this paper is intended as a friendly joke and the authors hope it is received as such.

DATA AVAILABILITY

The data that support the findings of this article are openly available [13].

APPENDIX A: BINARY PCMs FOR A SMALL QRC

The following are the binary representations of the matrices (11) and (12) discussed in Sec. III A:

$$H_X = \left(\begin{array}{cccccccccccccccccccccc|c}
0 & & 1 & & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & & \\
1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0& \\
0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&\\
0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&0&\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&1&0&0&0&0&\\
0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&&\\
0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&0\\
0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&1
\end{array}\right) \begin{array}{c} \\ \\ 0 \\ \\ \\ \\ 1 \\ \\ 0 \\ \\ \\ \\ 1 \end{array} \tag{A1}$$

$$H_Z = \left(\begin{array}{cccccccccccccccccccccc|c}
0 & & 1 & & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & & \\
1&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0\\
0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&1&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\
0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0\\
0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0\\
0&0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1
\end{array}\right) \begin{array}{c} \\ \\ 0 \\ \\ \\ \\ 1 \\ \\ 0 \\ \\ \\ \\ 1 \end{array} \tag{A2}$$

APPENDIX B: STRATEGIES FOR PREVENTING LOW-DISTANCE LOGICALS

To understand what kind of features of QRCs can lead to logical operator representations with weight less than $2s$ we can return to the example discussed in Sec. III A, since, as we can see from Fig. 3, this code actually has a distance of 4. We start by considering the product of the two X stabilizers from spoke 0 of the orange code, which can be written as a sum of binary vectors as follows:

$$\begin{array}{cccccccccccccccccccccc}
 & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & & \\
X(0,0) & 1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0\\
X(1,0) & 0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&1&0&0&0&0&0&0&0\\
\hline
 & 1&0&0&1&0&0&1&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0
\end{array} \tag{B1}$$

We can see that when restricted to the X codes the support of this operator lies within a single ring of the orange code. The product of the same two operators from the green code gives

$$\begin{array}{cccccccccccccccccccccc}
 & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & & \\
X(0,0) & 0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
X(1,0) & 0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&0&0\\
\hline
 & 0&1&0&0&1&0&1&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0
\end{array} \tag{B2}$$

whose restriction to the X codes lies within a single ring of the green code. The product of these two stabilizers is then

$$\begin{array}{cccccccccccccccccccccc}
 & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & & \\
X(0,0)+X(1,0) & 1&0&0&1&0&0&1&0&0&1&0&0&1&1&0&0&0&0&0&0&0&0&0\\
X(0,0)+X(1,0) & 0&1&0&0&1&0&1&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
\hline
 & 1&1&0&1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0
\end{array} \tag{B3}$$

which, when restricted to the Z codes, is only supported on the red code. Recall that there is a logical X operator supported on all qubits of rings 0 and 1 in the red code and another operator supported on all qubits of ring 0 in the orange and green codes. The product of these two operators with the stabilizer shown above is then

$$\begin{array}{rcccccccccccccccccccccc} & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & & 0 & & 1 & \\ \overline{X_1}+\overline{X_2} & 1&1&1&1&1&1&0&0&0&0&0&0&1&1&1&0&0&0&1&1&1&0&0&0 \\ X_{\text{stab}} & 1&1&0&1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&1&1&0&0&0&0 \\ \hline & 0&0&1&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0 \end{array} \tag{B4}$$

which only has weight 4.

Our ability to reduce the weight of the logical operator in this way depended on three things. (a) It was possible to take products of X stabilizers associated with a single X code in such a way that the resulting stabilizer was supported on only a single ring of this X code (when restricted to the X codes). (b) It was possible to take products of X stabilizers associated with different X codes in such a way that the resulting stabilizer was supported on only a single Z code (when restricted to the Z codes). (c) We were able to do both (a) and (b) simultaneously.

In particular, the ability to achieve (c) came from the fact that $H_1 = H_2$ and so the structure of the X stabilizers within X codes (H_2) and between X and Z codes (H_1) were the same. This fact meant that in Eq. (B3) taking a product of stabilizers restricted to a single ring in the X codes naturally gave rise to a stabilizer restricted to a single code in the Z codes. Choosing H_1 and H_2 to be very different means that a product of generators that produces a very structured operator in the X codes will generally produce a very unstructured operator in the Z codes and vice versa.

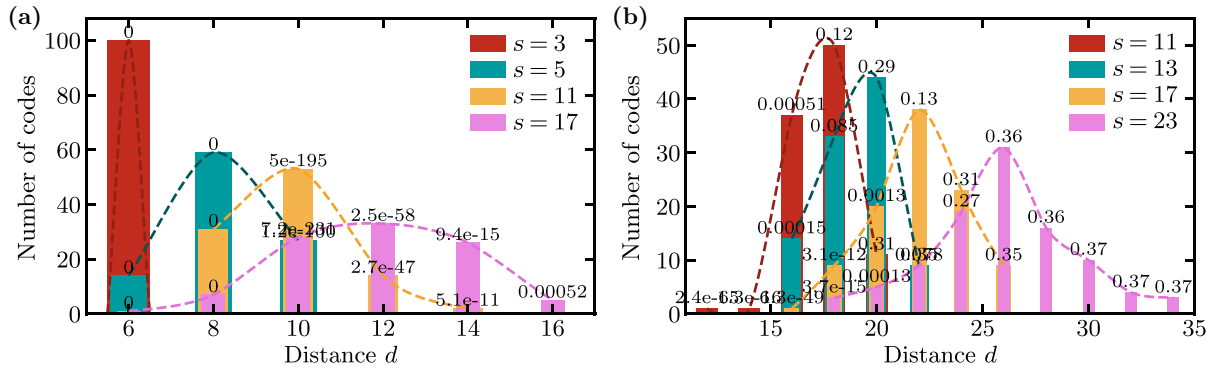
Numerical investigation (Fig. 5) suggests that choosing $H_1 \neq H_2$ does lead to an increase in average-case code distance in some regimes, but this increase is fairly small. Methods for reducing the likelihood of (a) and (b) were also developed but were not observed to lead to any distance improvements and so are not described here. An interesting question for future research is why the lifted product can so drastically reduce the distance of the input codes, and how this can (as much as possible) be prevented.

APPENDIX C: ACCURACY OF NUMERICALLY ESTIMATED CODE DISTANCES

As described in Ref. [16], the probability P_{fail} that the distance $\tilde{d}$ calculated by QDistRnd is greater than the true code distance is expected to be upper bounded by

$$P_{\text{fail}} < e^{-\langle n \rangle}, \tag{C1}$$

where $\langle n \rangle$ is the average number of times a codeword of weight $\tilde{d}$ was found,



Above we show versions of the plots from Fig. 6 with the average upper bound on failure probability displayed for each $(r, s, \tilde{d})$. We can see that for $r = 3$ the failure probability is negligible for all except the highest distance codes. For $r = 4$, on the other hand, this bound is much higher, on the order of 0.1 for almost all codes. However, we believe that these distance estimates are reliable for the following reasons: First, they are consistent with the pattern of distance distributions observed for the $r = 3$ codes, where the bound on failure probability is much lower. Second, the distance estimates are symmetrically distributed about the mean and failure probabilities for codes at the

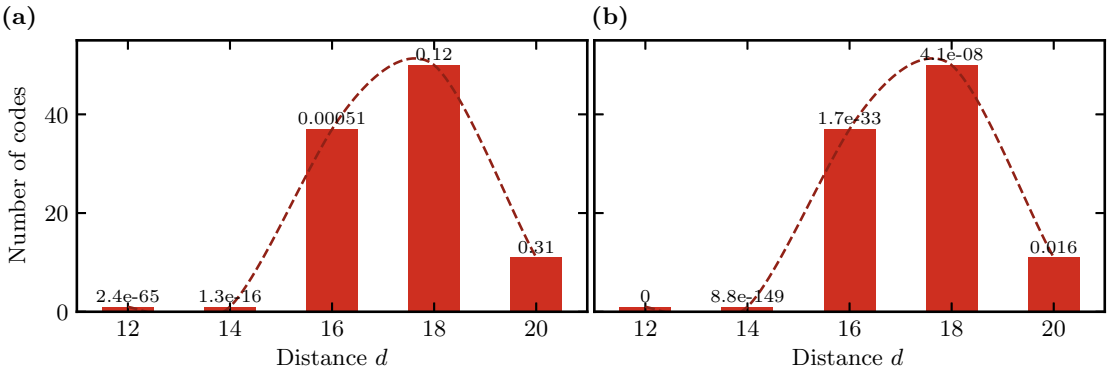


FIG. 10. Distance distributions for the $(4, 11)$ quantum radial code obtained from QDistRand for an increasing number of samples, i.e., 10^5 in (a) and 10^6 in (b). Annotated numbers on each bar indicate the upper bound on the probability that the distance $\tilde{d}$ calculated by QDistRand is greater than the true code distance.

low end of this distribution are fairly small. As there is no obvious reason why the distribution should be asymmetric (especially as all $r = 3$ distributions are symmetric) this suggests that the upper end of the distribution also has roughly the correct shape. Finally, increasing the number of samples in QDistRnd does not noticeably affect the distribution shapes. For instance, Fig. 10 shows the distributions of $(4, 11)$ codes with 10^5 (left) and 10^6 (right) samples.

The distributions are identical in both cases but the bounds on the failure probabilities of the latter are much lower than the former. We conjecture that, in the same way, taking additional samples would reduce the bounds on failure probabilities for all distributions in Fig. 6 without noticeably altering distribution shapes. Unfortunately, the number of samples and runtime required to test this for higher values of s would be prohibitively large.

APPENDIX D: DEFINITIONS OF CODES USED IN NUMERICS

The $[\![90, 8, 10]\!]$ code studied numerically in Sec. V is the lifted product of the classical codes defined by

$$A_1 = \begin{pmatrix} 3 & 2 & 1 \\ 4 & 1 & 4 \\ 1 & 2 & 3 \end{pmatrix}_5 \quad A_2 = \begin{pmatrix} 3 & 3 & 0 \\ 1 & 0 & 1 \\ 4 & 2 & 0 \end{pmatrix}_5 \tag{D1}$$

and for the $[\![352, 18, 20]\!]$ code,

$$A_1 = \begin{pmatrix} 10 & 10 & 1 & 6 \\ 4 & 7 & 5 & 2 \\ 8 & 10 & 6 & 9 \\ 1 & 6 & 0 & 6 \end{pmatrix}_{11} \quad A_2 = \begin{pmatrix} 9 & 5 & 8 & 3 \\ 5 & 4 & 1 & 0 \\ 0 & 4 & 6 & 10 \\ 2 & 8 & 4 & 2 \end{pmatrix}_{11}. \tag{D2}$$

APPENDIX E: DECODER OPTIMIZATIONS

It has been suggested that the performance of BP + OSD in surface codes is limited by the existence of short loops in the decoding graph [32,36], which prevent meaningful propagation of information once the iteration count becomes roughly equal to this loop length. However, such loops also exist in the decoding graphs of the codes described in this work, and indeed, in all decoding graphs for stabilizer codes which use repeated syndrome measurements (see Appendix B of Ref. [27]), and here we observe continued and substantial improvement in the performance of BP up to very high numbers of iterations ($\sim$10 000, see Fig. 11). We conjecture that the poor performance observed for BP in surface codes is due not just to the existence of short loops but to the combination of both short loops and pointlike syndromes.

The results in Appendix B of Ref. [36] suggest that the relevant part of the BP algorithm is the communication between factor nodes with syndrome -1 (this makes intuitive sense as these are the nontrivial features of the syndrome). We can think of each pair of these nodes (A, B) as being connected by a noisy channel, with the noise strength proportional to the separation distance between the nodes. Once this separation becomes roughly equal to the girth g of the graph, the channel becomes effectively completely noisy, preventing meaningful communication. However, if there exists a sequence of node pairs $(A, B), (B, C), \ldots (Y, Z)$ where the distance between consecutive nodes in each pair is less than g, information can still propagate through this path, albeit very slowly.

In the case of pointlike syndromes such paths are unlikely to exist, but they can occur very naturally in codes such as those studied here, where syndromes are extensive (i.e., the syndrome weight is nonconstant in the error weight). This slow propagation of information across long,

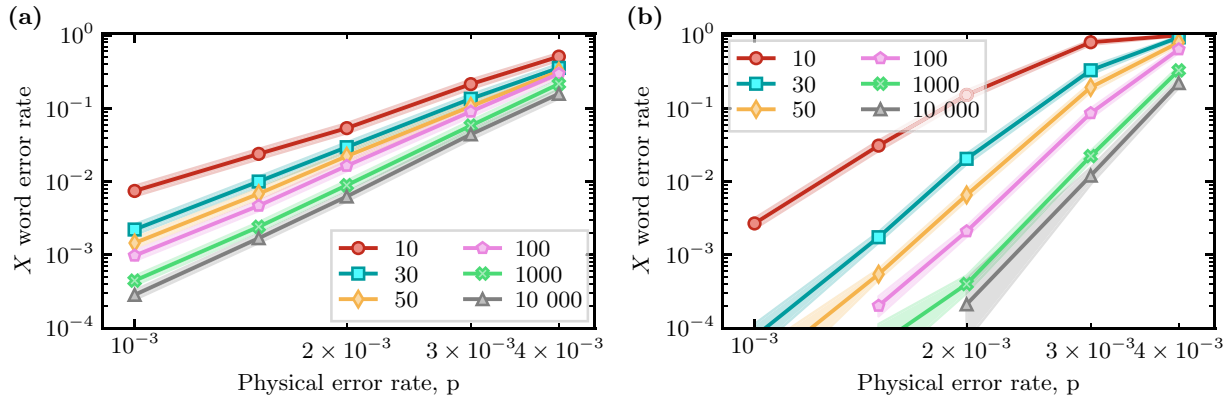


FIG. 11. Logical X error rate after 15 decoding rounds with the $(3, 1)$-overlapping window decoder in (a) for the $[\![90, 8, 10]\!]$ and in (b) for $[\![352, 18, 20]\!]$ code. Here we varied the number of maximum iterations in the min-sum decoder, shown in the legend, and fixed the reprocessing routine to OSD-0. A very large number of iterations are necessary before performance begins to plateau.

interconnected paths in codes with extensive syndromes potentially explains why a higher number of BP iterations can significantly improve error rates in these codes.

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

[2] N. P. Breuckmann and J. N. Eberhardt, Balanced product quantum codes, IEEE Trans. Inf. Theory **67**, 6653 (2021).

[3] P. Panteleev and G. Kalachev, in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022 (Association for Computing Machinery, New York, NY, USA, 2022), pp. 375–388.

[4] I. Dinur, M.-H. Hsieh, T.-C. Lin, and T. Vidick, Good quantum LDPC codes with linear time decoders, in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing STOC 2023* (Association for Computing Machinery, Orlando, FL, USA, 2023), pp. 905–918.

[5] A. Leverrier and G. Zémor, in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Computer Society, 2022), pp. 872–883.

[6] P. Panteleev and G. Kalachev, Quantum LDPC codes with almost linear minimum distance, IEEE Trans. Inf. Theory **68**, 213 (2022).

[7] P. Panteleev and G. Kalachev, Degenerate quantum LDPC codes with good finite length performance, Quantum **5**, 585 (2021).

[8] J. Roffe, L. Z. Cohen, A. O. Quintavalle, D. Chandra, and E. T. Campbell, Bias-tailored quantum LDPC codes, Quantum **7**, 1005 (2023).

[9] N. Raveendran, N. Rengaswamy, F. Rozpedek, A. Raina, L. Jiang, and B. Vasić, Finite rate QLDPC-GKP coding scheme that surpasses the CSS Hamming bound, Quantum **6**, 767 (2022).

[10] Q. Xu, J. P. Bonilla Ataides, C. A. Pattison, N. Raveendran, D. Bluvstein, J. Wurtz, B. Vasić, M. D. Lukin, L. Jiang, and H. Zhou, Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays, Nat. Phys. **20**, 1084 (2024).

[11] E. Sabo, L. G. Gunderman, B. Ide, M. Vasmer, and G. Dauphinais, Weight-reduced stabilizer codes with lower overhead, PRX Quantum **5**, 040302 (2024).

[12] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, High-threshold and low-overhead fault-tolerant quantum memory, Nature **627**, 778 (2024).

[13] Tools for generation and study of quantum radial codes, https://github.com/tRowans/radial-codes-public

[14] M. Fossorier, Quasi-cyclic low-density parity-check codes from circulant permutation matrices, IEEE Trans. Inf. Theory **50**, 1788 (2004).

[15] We thank Min Ye for this observation and for identifying an issue with a related claim in the previous version of this work. We note also that the instances where this is not the case are typically outside the parameter regime we will consider, although it is an interesting question for future work to understand the origin of these instances and identify a better set of conditions on the input matrices. Practically, we verify this condition on the number of linear dependencies explicitly for all generated codes.

[16] L. P. Pryadko, V. A. Shabashov, and V. K. Kozin, QDistRnd: A GAP package for computing the distance of quantum error-correcting codes, J. Open Source Software **7**, 4120 (2022).

[17] A. O. Quintavalle, M. Vasmer, J. Roffe, and E. T. Campbell, Single-shot error correction of three-dimensional homological product codes, PRX Quantum **2**, 020340 (2021).

[18] H.-K. Lin, P. K. Lim, A. A. Kovalev, and L. P. Pryadko, Abelian multi-cycle codes for single-shot error correction, arXiv:2506.16910 [quant-ph].

[19] L. P. Pryadko and W. Zeng, dist-m4ri: Distance of a classical or quantum CSS code, 2024, https://github.com/QEC-pages/dist-m4ri

[20] A. Jacob, C. McLauchlan, and D. E. Browne, Single-shot decoding and fault-tolerant gates with trivariate tricycle codes, arXiv:2508.08191 [quant-ph].

[21] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Towards practical large-scale quantum computation, Phys. Rev. A **86**, 032324 (2012).
[22] C. Gidney, Stim: A fast stabilizer circuit simulator, Quantum **5**, 497 (2021).
[23] C. Gidney, M. Newman, A. Fowler, and M. Broughton, A fault-tolerant honeycomb memory, Quantum **5**, 605 (2021).
[24] B. J. Brown, N. H. Nickerson, and D. E. Browne, Fault-tolerant error correction with the gauge color code, Nat. Commun. **7**, 12302 (2016).
[25] Of course, it is also possible to use the measurement outcomes from `stim` to decode on the Tanner graph for the quantum code rather than on the DEM. Such a strategy may work well in codes such as the 3D gauge color code [37] or 3D subsystem toric code [38] which use matching decoders, but will work very poorly for decoders such as BP which are highly dependent on the details of the graph.
[26] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).
[27] L. Berent, T. Hillmann, J. Eisert, R. Wille, and J. Roffe, Analog information decoding of Bosonic quantum low-density parity-check codes, PRX Quantum **5**, 020349 (2024).
[28] A. Gong, S. Cammerer, and J. M. Renes, Toward low-latency iterative decoding of QLDPC codes under circuit-level noise, arXiv:2403.18901 [quant-ph].
[29] L. Skoric, D. E. Browne, K. M. Barnes, N. I. Gillespie, and E. T. Campbell, Parallel window decoding enables scalable fault tolerant quantum computation, Nat. Commun. **14**, 7040 (2023).
[30] S. Huang and S. Puri, Increasing memory lifetime of quantum low-density parity check codes with sliding-window noisy syndrome decoding, Phys. Rev. A **110**, 012453 (2024).
[31] Note that this does not have to be true in more general codes, e.g., subsystem [39] and Floquet/dynamic [40,41] codes, where detecting regions can span multiple measurement cycles.
[32] J. Roffe, D. R. White, S. Burton, and E. Campbell, Decoding across the quantum low-density parity-check code landscape, Phys. Rev. Res. **2**, 043423 (2020).
[33] J. Roffe, LDPC: Python tools for low density parity check codes, 2022, https://pypi.org/project/ldpc/
[34] T. Hillmann, L. Berent, A. O. Quintavalle, J. Eisert, R. Wille, and J. Roffe, Localized statistics decoding: A parallel decoding algorithm for quantum low-density parity-check codes, Nat. Commun. **16**, 8214 (2025).
[35] D. Bluvstein *et al.*, Logical quantum processor based on reconfigurable atom arrays, Nature **626**, 58 (2024).
[36] O. Higgott and N. P. Breuckmann, Improved single-shot decoding of higher-dimensional hypergraph-product codes, PRX Quantum **4**, 020332 (2023).
[37] H. Bombín, Gauge color codes: Optimal transversal gates and gauge fixing in topological stabilizer codes, New J. Phys. **17**, 083002 (2015).
[38] A. Kubica and M. Vasmer, Single-shot quantum error correction with the three-dimensional subsystem toric code, Nat. Commun. **13**, 6272 (2022).
[39] O. Higgott and N. P. Breuckmann, Subsystem codes with high thresholds by Gauge fixing and reduced qubit overhead, Phys. Rev. X **11**, 031039 (2021).
[40] A. Townsend-Teague, J. M. de la Fuente, and M. Kesselring, Floquetifying the colour code, Electron. Proc. Theor. Comput. Sci. **384**, 265 (2023).
[41] M. Davydova, N. Tantivasadakarn, and S. Balasubramanian, Floquet codes without parent subsystem codes, PRX Quantum **4**, 020341 (2023).