This is a repository copy of *Transversal diagonal logical operators for stabiliser codes*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/204277/

Version: Published Version

Article:
Webster, M.A. orcid.org/0000-0002-5300-1643, Quintavalle, A.O. orcid.org/0000-0001-5101-5673 and Bartlett, S.D. orcid.org/0000-0003-4387-670X (2023) Transversal diagonal logical operators for stabiliser codes. New Journal of Physics, 25 (10). 103018. ISSN 1367-2630

https://doi.org/10.1088/1367-2630/acfc5f

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG IOP Institute of Physics

PAPER • OPEN ACCESS

Transversal diagonal logical operators for stabiliser codes

To cite this article: Mark A Webster *et al* 2023 *New J. Phys.* **25** 103018

View the article online for updates and enhancements.

You may also like

- Loss-tolerant teleportation on large stabilizer states
Sam Morley-Short, Mercedes Gimeno-Segovia, Terry Rudolph et al.

- Error suppression via complementary gauge choices in Reed-Muller codes
Christopher Chamberland and Tomas Jochym-O'Connor

- Unfolding the color code
Aleksander Kubica, Beni Yoshida and Fernando Pastawski

This content was downloaded from IP address 51.6.246.78 on 16/10/2023 at 14:25

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics
Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
26 April 2023

REVISED
5 September 2023

ACCEPTED FOR PUBLICATION
22 September 2023

PUBLISHED
12 October 2023

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

PAPER

Transversal diagonal logical operators for stabiliser codes

Mark A Webster[1,2], Armanda O Quintavalle[3] and Stephen D Bartlett[1,*]

[1] Centre for Engineered Quantum Systems, School of Physics, University of Sydney, Sydney, NSW 2006, Australia
[2] Sydney Quantum Academy, Sydney, NSW, Australia
[3] Department of Physics and Astronomy, University of Sheffield, Sheffield S3 7RH, United Kingdom
[*] Author to whom any correspondence should be addressed.

E-mail: stephen.bartlett@sydney.edu.au

Keywords: quantum error correction, quantum computation, quantum information

Abstract
Storing quantum information in a quantum error correction code can protect it from errors, but the ability to transform the stored quantum information in a fault tolerant way is equally important. Logical Pauli group operators can be implemented on Calderbank-Shor-Steane (CSS) codes, a commonly-studied category of codes, by applying a series of physical Pauli X and Z gates. Logical operators of this form are fault-tolerant because each qubit is acted upon by at most one gate, limiting the spread of errors, and are referred to as transversal logical operators. Identifying transversal logical operators outside the Pauli group is less well understood. Pauli operators are the first level of the Clifford hierarchy which is deeply connected to fault-tolerance and universality. In this work, we study transversal logical operators composed of single- and multi-qubit diagonal Clifford hierarchy gates. We demonstrate algorithms for identifying all transversal diagonal logical operators on a CSS code that are more general or have lower computational complexity than previous methods. We also show a method for constructing CSS codes that have a desired diagonal logical Clifford hierarchy operator implemented using single qubit phase gates. Our methods rely on representing operators composed of diagonal Clifford hierarchy gates as diagonal XP operators and this technique may have broader applications.

1. Overview

Quantum error correction has become a very active area of research because of its potential to mitigate noise in complex quantum devices. Recent experimental results have validated the storage of quantum information in the codespace of a quantum error correction code as a practical way of protecting it from noise (see [1–3]). Many of these initial demonstrations have made use of Calderbank-Shor-Steane (CSS) codes [4], a well-studied class of quantum error correction codes that are relatively simple to analyse and implement.

To implement algorithms on quantum computers, we also need to transform the stored quantum information in a fault-tolerant way. One method of implementing fault-tolerant logical operations on CSS codes is to use transversal logical operators. Transversal logical operators have depth-one circuit implementations involving single or multi-qubit gates. Such implementations are considered fault-tolerant because an error on one physical qubit can only spread to a limited number of other qubits when applying the logical operator. Whilst the Eastin–Knill theorem rules out the existence of a quantum error correcting code with a set of transversal operators that is universal [5], determining the transversal gates of a quantum error correction code is key to designing a fault-tolerant architecture.

Deeply connected to fault tolerance and universality is the Clifford hierarchy [6] of unitary operators. The first level of the Clifford hierarchy is the Pauli group $\mathcal{CH}_1 := \langle iI, X, Z\rangle$. Conjugation of Paulis by operators at level $t+1$ results in an operator at level t. The level $t+1$ operators $A \in \mathcal{CH}_{t+1}$ are then defined recursively as those for which $ABA^{-1} \in \mathcal{CH}_t$ for all $B \in \mathcal{CH}_1$. Level 2 Clifford hierarchy gates include the single-qubit Hadamard and $S := \sqrt{Z}$ gates, as well as the two-qubit controlled-Z (CZ) gates. Level 3 gates include the single-qubit $T := \sqrt{S}$ gate as well as the multi-qubit controlled-S (CS) and controlled-controlled-Z (CCZ) gates. A set of gates that includes all level-2 gates and at least one level-3 gate is universal [7].

© 2023 The Author(s). Published by IOP Publishing Ltd on behalf of the Institute of Physics and Deutsche Physikalische Gesellschaft

Logical Pauli group operators can be implemented transversally on CSS codes and identifying these is relatively straightforward. Identifying transversal logical operators at higher levels of the Clifford hierarchy is more challenging and existing methods are of exponential complexity in either the number of physical or logical qubits in the code. Some classes of CSS codes with high degrees of symmetry are known to have non-Pauli transversal logical operators. Examples using single-qubit diagonal gates include the seven-qubit Steane code [8], two-dimensional (2D) color codes [9] and triorthogonal codes [10]. Examples of CSS codes which have logical operators made from single and multi-qubit gates include the 2D toric code [11], codes with ZX-symmetries [12] and symmetric hypergraph product codes [13].

In this paper, we present a suite of method and algorithms for identifying diagonal transversal logical operators on any CSS code, without any knowledge of any symmetries of the code. The building blocks of our logical operators are physical single- or multi-qubit diagonal gates, at a given level t of the Clifford hierarchy. Our methods scale as a polynomial in the number of physical and/or logical qubits in the code, with one exception. We also give a method for constructing a CSS code that has a transversal implementation of a desired diagonal logical Clifford hierarchy operator using single-qubit gates. Our new algorithms use the XP formalism, introduced in [14], which is a powerful tool for representing the logical operator structure of a stabiliser code.

1.1. Existing work on transversal logical operators

We briefly review previous methods for identifying diagonal logical operators of arbitrary CSS codes, and methods for constructing CSS codes with a desired transversal logical operator. In [15], a method is given to find all logical operators at level 2 of the Clifford hierarchy for a CSS code by mapping it to a classical code over $GF(4)$. This method involves calculating the automorphism group of the classical code, which has exponential complexity in the number of qubits in the stabiliser code [16].

There has also been a significant amount of work on logical operators constructed from single- and multi-qubit diagonal Clifford hierarchy gates. In [17], operators composed of diagonal Clifford hierarchy gates on one or two qubits are shown to be representable as symmetric matrices over $\mathbb{Z}_N$, referred to as quadratic form diagonal (QFD) gates. Necessary and sufficient conditions for a QFD gate to act as a logical operator on a CSS code are then presented. In [18], a method of generating circuits using multi-qubit gates which implement arbitrary logical operators at level 2 of the Clifford hierarchy is presented. A method for generating CSS codes with transversal diagonal logical operators at increasing levels of the Clifford hierarchy is presented in [19], along with a method to increase the Z-distance of such codes. In [14], we demonstrated an algorithm for finding all diagonal logical operators composed of single-qubit phase gates which, for CSS codes, involves taking the kernel modulo N of a matrix with $n + 2^k$ columns where n and k are the number of physical and logical qubits respectively.

1.2. Contribution of this work

In this work, we present efficient methods to identify and test diagonal logical operators on CSS codes using both single and multi-qubit diagonal Clifford hierarchy gates as building blocks. These methods generalise to non-CSS stabiliser codes. We also present a technique for generating CSS codes with implementations of any desired diagonal Clifford hierarchy logical operator using single-qubit phase gates.

We first consider operators composed of single-qubit phase gates at level t of the Clifford hierarchy. We show that these can be represented as diagonal XP operators of precision $N = 2^t$. For logical operators of this form, we demonstrate the following algorithms that apply to any CSS code and at any desired level of the Clifford hierarchy:

1. **Finding a generating set of diagonal logical identity operators for the code**: an XP operator may act as a logical identity, but may not be an element of the stabiliser group of a CSS code. The logical identities are used as inputs to several other algorithms (section 3.1)
2. **Search for an implementation of a desired logical controlled-phase operator on the code**: useful for checking if a given CSS code has a transversal implementation of a particular logical operator and for checking the results of other algorithms (section 3.2);
3. **Determining if a given diagonal operator acts as a logical operator on the code**: this method is of linear complexity in the number of independent X-checks whereas existing methods are of exponential complexity (section 3.3);
4. **Finding a generating set of diagonal logical operators on the code**: the generating set gives us a complete understanding of the diagonal logical operator structure of a CSS code, and can be used on CSS codes with a large number of physical and logical qubits at any desired level of the Clifford hierarchy (section 3.4);

Table 1. Comparison of search and test algorithms for diagonal logical operators. The space complexity of the algorithm is expressed in terms of the dimensions of the key matrices used. The time complexity is based on the number of times we calculate a Howell matrix form—these dominate the complexity of the algorithms. We calculate the number of matrix operations and multiply this by a time complexity of $\mathcal{O}(mn^2)$ for finding the Howell form of an $m \times n$ matrix.

Algorithm	Matrix dimensions	Matrix operations	Time complexity
1. Diagonal logical identity group generators	$\mathcal{O}((k+r)^t \times n)$	$\mathcal{O}(1)$	$\mathcal{O}((k+r)^t n^2)$
2. Search by logical action	$\mathcal{O}((k+r)^t \times n)$	$\mathcal{O}(1)$	$\mathcal{O}((k+r)^t n^2)$
3. Logical operator test*	$\mathcal{O}(n \times n)$	$\mathcal{O}(r)$	$\mathcal{O}(rn^3)$
4. Diagonal logical operator group generators*	$\mathcal{O}(n \times n)$	$\mathcal{O}(r)$	$\mathcal{O}(rn^3)$
5. Determine action of diagonal logical operator	$\mathcal{O}(k^t \times n)$	$\mathcal{O}(1)$	$\mathcal{O}(k^t n^2)$
6. Depth-one logical operators**	$\mathcal{O}(n^t \times n^t)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n n^{3t})$

Note that entries annotated with* require the diagonal logical identities of algorithm 1 as input. Entries annotated with** require the diagonal logical operators of algorithm 4 as input.

5. **Expressing the action of a diagonal logical operator as a product of logical controlled-phase gates**: the action of a logical operator can be difficult to interpret, particularly for codes with a large number of logical qubits. This method greatly simplifies the interpretation of logical actions (section 3.5).

We then show that multi-qubit diagonal Clifford hierarchy gates acting on a codesepace can be represented as diagonal XP operators acting on a larger Hilbert space via an embedding operator (section 4.3). We demonstrate algorithms for:

6. **Finding depth-one implementations of logical operators composed of diagonal Clifford hierarchy gates**: on small CSS codes, this allows us to identify and verify the depth-one logical operators of [11–13] with no knowledge of the symmetry of the code (section 4.4);
7. **Canonical implementations of a desired logical controlled-phase operator composed of multi-qubit controlled-phase gates**: this allows us to write closed-form expressions for arbitrary diagonal Clifford hierarchy logical operators section 5.1;
8. **Construction of CSS codes which have an implementation of a desired logical controlled-phase operator composed of single qubit phase gates**: the canonical logical operator implementation allows us to construct families of CSS codes which have transversal implementations of a desired diagonal Clifford hierarchy logical operator section 5.4.

Apart from the depth-one search algorithm, the eight algorithms have complexity that is polynomial in the parameters n, k, r of the CSS code (see below). As a result, they can be applied to 'large' codes that have so far been out of reach of existing methods. There are no restrictions on the level of the Clifford hierarchy or maximum support size of the physical gates used in the methods.

A summary of the characteristics and computational complexity of search and test algorithms is presented in table 1. Complexity is expressed in terms of the following variables:

- Required level of the Clifford hierarchy t;
- Number of physical qubits n in the CSS code;
- Number of logical qubits k in the CSS code;
- Number of independent X-checks r in the CSS code;

The space complexity of the algorithm is expressed in terms of the dimensions of the key matrices used. The time complexity is based on the number of times we calculate a Howell matrix form—these dominate the complexity of the algorithms. We calculate the number of matrix operations and multiply this by a time complexity of $\mathcal{O}(mn^2)$ for finding the Howell form of an $m \times n$ matrix.

The algorithms have been implemented in a Python GitHub repository accessible under the GNU General Public License. A range of sample codes are also available for testing in this repository, including Reed–Muller codes, hyperbolic surface codes, triorthogonal codes and symmetric hypergraph product codes.

2. Background

This section reviews the necessary background material for this work. We first introduce the Clifford hierarchy of diagonal operators and introduce a vector representation of these. We then outline notation and fundamental properties of CSS codes. Next, we define what we mean by a diagonal logical operator on a CSS code. We then present an example illustrating the types of diagonal logical operators we consider in this work for the well-known $[[4,2,2]]$ code. We then review the XP stabiliser formalism and some fundamental properties of the XP operators, which we will use to represent logical operators composed of diagonal Clifford hierarchy gates. We explain the logical operator group structure in the XP formalism, which is somewhat different than in the Pauli stabiliser formalism.

2.1. Diagonal Clifford hierarchy operators

Here we review the properties of operators in the diagonal Clifford hierarchy. We will use diagonal gates at level t of the Clifford hierarchy on n qubits as the building blocks for logical operators. The diagonal Clifford hierarchy operators at each level form a group generated by the following operators [20]:

- Level 1: Pauli Z gate on qubit $i : 0 \leqslant i < n$ denoted Z_i;
- Level 2: controlled-Z (CZ_{ij}) and $S_i := \sqrt{Z_i}$;
- Level 3: CCZ_{ijk}, CS_{ij} and $T_i := \sqrt{S_i}$;
- Level $t+1$: square roots and controlled versions of operators from level t.

At each level, we refer to the generators as **level-t controlled-phase gates**. Where an operator is an element of the diagonal Clifford hierarchy group at level t, we say that it is **composed of level-t controlled-phase gates**.

The single-qubit **phase gate** at level t is of form $\mathrm{diag}(1, \exp(2\pi i/N))$ where $N := 2^t$. If an operator is an element of the group generated by single-qubit phase gates at level t, we say it is **composed of level-t phase gates**.

The matrix form of any diagonal transversal logical operator of a CSS code must have entries of form $\exp(q\pi i/2^t)$ for integers q, t, as shown in [21]. Such matrices are elements of the diagonal Clifford hierarchy group at some level, and so considering logical operators composed of controlled-phase gates yields all possible diagonal transversal logical operators on a CSS code.

2.2. Vector representation of controlled-phase operators

We now introduce a vector representation of controlled-phase operators that underpins our analytical methods. Fix a level t of the Clifford hierarchy (section 2.1) and let $N := 2^t$. Let $\omega := \mathrm{e}^{\pi\,\mathrm{i}/N}$ be a $(2N)$th root of unity. The operator $\mathrm{CP}_N(q, \mathbf{v})$, where $q \in \mathbb{Z}_{2N}$ and $\mathbf{v}$ is a binary vector of length n, is defined as follows by its action on a computational basis vectors $|\mathbf{e}\rangle$ for $\mathbf{e} \in \mathbb{Z}_2^n$:

$$\mathrm{CP}_N(q,\mathbf{v})\,|\mathbf{e}\rangle := \begin{cases} \omega^q|\mathbf{e}\rangle & \text{if } \mathbf{v} \preccurlyeq \mathbf{e}; \\ |\mathbf{e}\rangle & \text{otherwise.} \end{cases} \tag{1}$$

The relation $\preccurlyeq$ is a partial order for binary vectors based on their support (the set of indices where the vector is non-zero). The expression $\mathbf{v} \preccurlyeq \mathbf{e}$ indicates $\mathrm{supp}(\mathbf{v}) \subseteq \mathrm{supp}(\mathbf{e}) \iff \mathbf{ev} = \mathbf{v}$ where vector multiplication is componentwise. For an integer $0 \leqslant i < n$, we will also write $i \preccurlyeq \mathbf{v}$ if $\mathbf{v}[i] = 1$. The phase applied can be expressed more concisely as follows:

$$\mathrm{CP}_N(q,\mathbf{v})\,|\mathbf{e}\rangle = \omega^{q\cdot p_{\mathbf{v}}(\mathbf{e})}|\mathbf{e}\rangle \text{where } p_{\mathbf{v}}(\mathbf{e}) := \prod_{i \preccurlyeq \mathbf{v}} \mathbf{e}[i]\,. \tag{2}$$

Each generator of the diagonal Clifford hierarchy can be written in vector form. To see this, we note that the phase gate at level t can be written as $P := \mathrm{diag}(1, \omega^2)$. The phase operator acting on qubit i can be written in vector form as $P_i = \mathrm{CP}_N(2, \mathbf{b}_i^n)$ where $\mathbf{b}_i^n$ is the length n binary vector, which is all zero apart from component i which is one. Similarly, the operator $\mathrm{CP}_{ij} = \mathrm{CP}_N(2, \mathbf{b}_{ij}^n)$ where $\mathbf{b}_{ij}^n$ is zero apart from components i and j. The operators of form $\mathrm{CP}_N(2^{\mathrm{wt}(\mathbf{v})}, \mathbf{v})$ with $1 \leqslant \mathrm{wt}(\mathbf{v}) \leqslant t$ are the generators of the level-t controlled-phase operators presented in section 2.1.

Example 2.1 (vector representation of level 3 controlled-phase operators). This example illustrates the vector representation of level 3 diagonal Clifford hierarchy operators. At level $t = 3$ the generators have vector representations as follows:

$$T_i = \mathrm{CP}_8(2, \mathbf{b}_i^n) \tag{3}$$

$$CS_{ij} = \mathrm{CP}_8\left(4, \mathbf{b}_{ij}^n\right) \tag{4}$$

$$CCZ_{ijk} = \mathrm{CP}_8\left(8, \mathbf{b}_{ijk}^n\right). \tag{5}$$

We also include $\omega I = \mathrm{CP}_8(1, 0)$ as a generator at the third level of the hierarchy as phases of this form occur in the commutation relation for controlled-phase operators—see equations (30) and (31).

2.3. CSS codes

Here we introduce some key notation and results for CSS codes. Our notation for CSS codes is somewhat different to that in the literature and is used because it simplifies the statement of our results. Although we focus on CSS codes in this work, the methods are applicable to any stabiliser code as set out in appendix C. For our purposes, a CSS code on n qubits is specified by an $r \times n$ binary matrix S_X the rows of which we refer to as the X-**checks** and a $k \times n$ binary matrix L_X whose rows are referred to as the X-**logicals**. We assume that the rows of S_X and L_X are independent binary vectors—otherwise we can use linear algebra modulo 2 to ensure this. The Z-**checks** can be calculated by taking the kernel modulo 2 of the X-checks and X-logicals, i.e.

$$S_Z := \ker_{\mathbb{Z}_2}\begin{pmatrix} S_X \\ L_X \end{pmatrix}. \tag{6}$$

In equation (6), the notation $\ker_{\mathbb{Z}_2}$ refers to the basis in reduced row echelon form of the kernel modulo 2 of a binary matrix. We form **stabiliser generators** $\mathbf{S}_X, \mathbf{S}_Z$ from the rows of S_X and S_Z in the obvious way—if $\mathbf{x}$ is a row of S_X then the corresponding stabiliser generator is $\prod_{0\leqslant i<n} X_i^{\mathbf{x}[i]}$. The **codespace** is the simultaneous $+1$ eigenspace of the **stabiliser group** $\langle \mathbf{S}_X, \mathbf{S}_Z\rangle$ and is a subspace of $\mathcal{H}_2^n$. The codespace is spanned by 2^k **canonical codewords** which are indexed by binary vectors $\mathbf{v}$ of length k and are defined as follows:

$$|\mathbf{v}\rangle_{\mathrm{L}} := \sum_{\mathbf{u}\in\mathbb{Z}_2^r} |\mathbf{e_{uv}}\rangle := \sum_{\mathbf{u}\in\mathbb{Z}_2^r} |\mathbf{u}S_X + \mathbf{v}L_X\rangle. \tag{7}$$

In the above expression, matrix operations are modulo 2. For simplicity, we are not concerned with normalising codeword states. It may be possible to make a different **choice of basis** for the span $\langle L_X\rangle$ over $\mathbb{Z}_2$. The choice of basis affects the labelling of the canonical codewords by binary vectors $\mathbf{v}$ of length k, but does not otherwise change the set of canonical codewords.

2.4. Logical operators of CSS codes

We now describe what we mean by a logical operator on a CSS code. Let $\mathcal{C}: \mathcal{H}_2^k \to \mathcal{H}_2^n$ be the **encoding operator** which takes computational basis vectors to canonical codewords of equation (7) i.e. $\mathcal{C}|\mathbf{v}\rangle = |\mathbf{v}\rangle_{\mathrm{L}}$ for $\mathbf{v}\in\mathbb{Z}_2^k$. Now let B be a unitary operator acting on k qubits. We say that an operator $\overline{B}$ acting on n qubits is a **logical** B **operator** if

$$\overline{B}\mathcal{C} = \mathcal{C}B. \tag{8}$$

A unitary operator B is **diagonal** if we can write $B := \mathrm{diag}(\mathbf{c})$ for some complex-valued vector $\mathbf{c}$ of length 2^k representing the phase applied to each computational basis vector, i.e. $B|\mathbf{v}\rangle = \mathbf{c_v}|\mathbf{v}\rangle$ for $\mathbf{v}\in\mathbb{Z}_2^k$ and $\mathbf{c_v}\in\mathbb{C}$. If $\overline{B}$ is a diagonal logical operator, then B is diagonal as well, though the converse is not necessarily true. From equations (8) and (7), we have:

$$\overline{B}\mathcal{C}|\mathbf{v}\rangle = \overline{B}|\mathbf{v}\rangle_{\mathrm{L}} = \overline{B}\sum_{\mathbf{u}\in\mathbb{Z}_2^r}|\mathbf{e_{uv}}\rangle = \sum_{\mathbf{u}\in\mathbb{Z}_2^r}\overline{B}|\mathbf{e_{uv}}\rangle \tag{9}$$

$$= \mathcal{C}B|\mathbf{v}\rangle = \mathbf{c_v}|\mathbf{v}\rangle_{\mathrm{L}} = \sum_{\mathbf{u}\in\mathbb{Z}_2^r}\mathbf{c_v}|\mathbf{e_{uv}}\rangle. \tag{10}$$

As a result, we can check if $\overline{B}$ is a logical B operator by doing the following:

1. For each $\mathbf{v}\in\mathbb{Z}_2^k$, calculate $\mathbf{c_v}\in\mathbb{C}$ such that $B|\mathbf{v}\rangle = \mathbf{c_v}|\mathbf{v}\rangle$;
2. For each $\mathbf{u}\in\mathbb{Z}_2^r$, check that $\overline{B}|\mathbf{e_{uv}}\rangle = \mathbf{c_v}|\mathbf{e_{uv}}\rangle$.

This method of checking whether a diagonal unitary is a logical operator involves $\mathcal{O}(2^{r+k})$ steps; we present a method in section 3.3 with linear complexity in r.

We say that an operator $\overline{B}$ is a **logical identity** if $\overline{B}|\mathbf{v}\rangle_{\mathrm{L}} = |\mathbf{v}\rangle_{\mathrm{L}}$ for all $\mathbf{v}\in\mathbb{Z}_2^k$—that is, it fixes each canonical codeword and hence each element of the codespace. If $\overline{B}$ is diagonal, as a consequence of equation (10), it is a logical identity if and only if $\overline{B}|\mathbf{e_{uv}}\rangle = |\mathbf{e_{uv}}\rangle$ for all $\mathbf{u}\in\mathbb{Z}_2^r, \mathbf{v}\in\mathbb{Z}_2^k$.

Whether a diagonal operator is a logical identity or a logical operator is independent of the choice of basis for the span $\langle L_X\rangle$ (see section 2.3). However, the logical action of the operator depends on the labelling the canonical codewords and so is dependent on the choice of basis for $\langle L_X\rangle$.

Example 2.2 (transversal logical operators of [[4,2,2]] code). We use the [[4,2,2]] code to illustrate the types of transversal logical operators we consider in this work. Using the notation introduced in section 2.3, the X-checks and X-logicals of the code are:

$$S_X := \begin{pmatrix} 1111 \end{pmatrix} \tag{11}$$

$$L_X := \begin{pmatrix} 0101 \\ 0011 \end{pmatrix}. \tag{12}$$

In this case, there are $r = 1$ X-checks and $k = 2$ X-logicals. There are $2^k = 4$ canonical codewords which we calculate using equation (7):

$$\begin{aligned} |00\rangle_L &:= |0000\rangle + |1111\rangle \\ |01\rangle_L &:= |0011\rangle + |1100\rangle \\ |10\rangle_L &:= |0101\rangle + |1010\rangle \\ |11\rangle_L &:= |0110\rangle + |1001\rangle. \end{aligned} \tag{13}$$

We can calculate the single Z-check as follows:

$$S_Z := \ker_{\mathbb{Z}_2} \begin{pmatrix} S_X \\ L_X \end{pmatrix} = \begin{pmatrix} 1111 \end{pmatrix}. \tag{14}$$

Readers can verify that $Z^{\otimes 4}$ acts as a logical identity by checking that $Z^{\otimes 4}|\mathbf{v}\rangle_L = |\mathbf{v}\rangle_L$ for each of the canonical codewords.

The following are examples of transversal diagonal logical operators composed of controlled-phase gates at level 2 whose actions can be verified by applying the method of section 2.4:

1. **Single-qubit phase gates** controlled-Z: $\overline{CZ_{01}} = S_0^3 S_1 S_2 S_3^3$
2. **Multi-qubit controlled-phase gates** S operator on both logical qubits: $\overline{S_0 S_1} = S_1 S_2 CZ_{03}$

2.5. The XP formalism

The XP formalism is a generalisation of the Pauli stabiliser formalism, and we will show that diagonal Clifford hierarchy operators can be represented as diagonal XP operators. In the XP formalism, we fix an integer **precision** $N \geqslant 2$ and let $\omega = \exp(\pi i/N)$ be a $(2N)$th root of unity. We define a diagonal phase operator $P = \mathrm{diag}(1, \omega^2)$ which is a $1/N$ rotation around the Z axis and consider the group of XP operators $\mathcal{XP}_N^n$ that is generated by $\omega I, X_i, P_i$ where P_i is a P operator applied to qubit i. By setting $N := 2^t$, it is easy to see that the P_i correspond to the level t phase gates of section 2.1, and so any operator composed of single-qubit phase gates can be represented as a diagonal XP operator. For example, setting $t = 1$ results in $N = 2, \omega = i$ and $P = Z$ so $\mathcal{XP}_2^n$ is the Pauli group on n qubits.

The XP formalism has a fundamental commutation relation that allows us to move P operators to the right of X operators:

$$PX = \omega^2 \mathrm{XP}^{-1}. \tag{15}$$

All XP operators have a unique **vector representation** with a phase component $p \in \mathbb{Z}_{2N}$, an X-component $\mathbf{x} \in \mathbb{Z}_2^n$ and a Z-component $\mathbf{z} \in \mathbb{Z}_N^n$. The Z-component is modulo N, for instance, because $P^N = I$. The XP operator formed from these components is:

$$\mathrm{XP}_N(p|\mathbf{x}|\mathbf{z}) := \omega^p \prod_{0 \leqslant i < n} X_i^{\mathbf{x}[i]} P_i^{\mathbf{z}[i]}. \tag{16}$$

Diagonal XP operators are those with a zero X-component. The vector form of XP operators allows us to perform algebraic operations efficiently via componentwise addition and multiplication of vectors—examples are given in table 4 of [14]. In particular, the **action of an XP operator** on a computational basis element $|\mathbf{e}\rangle$ where $\mathbf{e} \in \mathbb{Z}_2^n$ is determined as follows:

$$\mathrm{XP}_N(p|\mathbf{x}|\mathbf{z})|\mathbf{e}\rangle = \omega^{p+2\mathbf{e}\cdot\mathbf{z}}|\mathbf{e} \oplus \mathbf{x}\rangle \tag{17}$$

where $N = 2^t$, we can determine the lowest level of the Clifford hierarchy at which a diagonal operator $B := \mathrm{XP}_N(0|0|\mathbf{z})$ occurs. Let $g := \mathrm{GCD}(N, \mathbf{z})$ be the GCD of N and each component of $\mathbf{z}$. As $N = 2^t, g$ is a power of 2 and $B = \mathrm{XP}_{N/g}(p/g|0|\mathbf{z}/g)$. Accordingly, B occurs at level $t - \log_2(g)$ of the diagonal Clifford hierarchy.

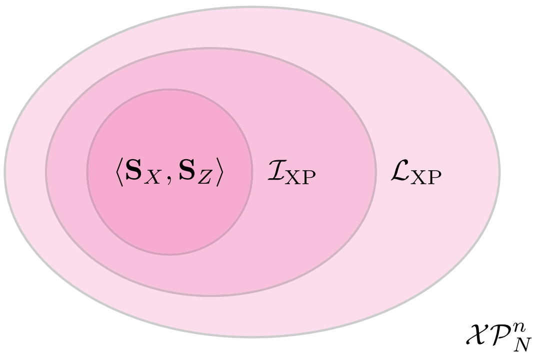

$\langle \mathbf{S}_X, \mathbf{S}_Z \rangle$
$\mathcal{I}_{\text{XP}}$
$\mathcal{L}_{\text{XP}}$
$\mathcal{XP}_N^n$

Figure 1. Relationship between XP operator groups: here, $\mathcal{XP}_N^n$ is the group of all XP operators of precision N on n qubits. The stabiliser group $\langle \mathbf{S}_X, \mathbf{S}_Z \rangle$ of a CSS code is a subgroup of the logical XP identity group $\mathcal{I}_{\text{XP}}$ which fixes all elements of the codespace which, in turn, is a subgroup of the logical operators of XP form $\mathcal{L}_{\text{XP}}$.

Example 2.3 (determining Clifford hierarchy level of XP operators). Let $t = 3$ and $B = \text{XP}_8(0|0|4444)$, so that $g = \text{GCD}(8, 4) = 4$. Hence $B = \text{XP}_2(0|0|1111) = Z^{\otimes 4}$ and occurs at level $t - \log_2(4) = 3 - 2 = 1$ of the Clifford hierarchy.

2.6. Logical identity and logical operator groups in the XP formalism

We now look at the logical group structure of a CSS code in the XP formalism with reference to the definitions of logical operators in section 2.4. In the stabiliser formalism, a Pauli operator acts as a logical identity if and only if it is in the stabiliser group $\langle \mathbf{S}_X, \mathbf{S}_Z \rangle$. In the XP stabiliser formalism, an XP operator may act as a logical identity but not be in the stabiliser group—we will see an instance of this in example 3.1. The **logical XP identity group**, $\mathcal{I}_{\text{XP}}$, are the XP operators of precision N which fix each element of the codespace. The stabiliser group is a subgroup of $\mathcal{I}_{\text{XP}}$ but may not be equal to it.

The **logical XP operator group**, $\mathcal{L}_{\text{XP}}$, are the XP operators of precision N that are logical B operators for some unitary B acting on k qubits. Logical XP operators may have actions outside the Pauli group, and the logical $\overline{CZ_{01}}$ operator of example 2.2 is an instance of such an operator. Logical identities are elements of $\mathcal{L}_{\text{XP}}$ that have a trivial action. The logical groups in the XP formalism are summarised in figure 1.

3. Logical operators composed of single-qubit phase gates

In this section, we present methods for identifying and testing logical operators composed of single-qubit phase gates at a given level t of the Clifford hierarchy. Operators of this form can be identified with diagonal XP operators of precision $N = 2^t$. The algorithms in this section are of polynomial complexity in the code parameters n, k, r (section 2.3), so they can be used on CSS codes with a large number of physical or logical qubits.

This section is structured as follows. We first show how to calculate generators for the diagonal logical identity XP group. This is an important first step for a number of our algorithms. We then demonstrate an algorithm that searches for a diagonal XP operator with a desired logical action. Next, we set out an efficient method for testing if a given diagonal XP operator is a logical operator on a CSS code. We then show how to use this test to find all diagonal logical operators of XP form. Finally, we show how to express the action of a diagonal logical XP operator in terms of a product of logical controlled phase operators. We use the hypercube code of [22, 23] which has a rich logical operator structure an example throughout this section. We also demonstrate the use of the algorithms on larger codes such as hyperbolic color codes [24], poset codes [25] and triorthogonal codes [10].

3.1. Diagonal logical XP identity group generators

Calculating generators for the logical identity group of a CSS code is an important first step for several of the algorithms discussed in this paper. An algorithm for determining the logical identity group is set out in section 6.2 of [14]. Here, we present a simplified version for CSS codes.

Due to the discussion in section 2.4, a diagonal logical identity operator fixes all $|\mathbf{e}_{\mathbf{uv}}\rangle$ in the canonical codewords of equation (7). Now let $N := 2^t$ and let $B := \text{XP}_N(2p|0|\mathbf{z})$ be a diagonal XP operator. Using equation (17), the action of B on the computational basis vector $|\mathbf{e}_{\mathbf{uv}}\rangle$ is $B|\mathbf{e}_{\mathbf{uv}}\rangle = \omega^{2p+2\mathbf{e}_{\mathbf{uv}}\cdot\mathbf{z}}|\mathbf{e}_{\mathbf{uv}}\rangle$. Considering the action of B on $|\mathbf{e}_{00}\rangle = |0\rangle$, we see that $p = 0 \mod 2N$. As $\omega^{2N} = 1$, B applies a trivial phase to $|\mathbf{e}_{\mathbf{uv}}\rangle$ if and only if $\mathbf{e_{uv}} \cdot \mathbf{z} = 0 \mod N$. We can find all such solutions by taking the kernel of a suitably constructed matrix modulo N. This is done via the **Howell matrix form** [26] which is a generalisation of the reduced row echelon form for modules over rings such as $\mathbb{Z}_N$. The notation $\ker_{\mathbb{Z}_N}(E_M)$ means the Howell basis of the kernel of the matrix E_M modulo N.

Algorithm 1. Logical identity group generators.

```
Input:
  1. The $X$-checks $S_X$ and $X$-logicals $L_X$ of a CSS code (section 2.3);
  2. The desired level of the Clifford hierarchy $t$ (section 2.1).
Output: a matrix $K_M$ whose rows are the $Z$-components of a set of generators for the diagonal logical identity XP group
of precision $N = 2^t$ (section 2.6).

Method:
  1. Let $E_M$ be the binary matrix whose rows are the $\mathbf{e_{uv}} := \mathbf{u}S_X + \mathbf{v}L_X$ of equation (7);
  2. Let $N := 2^t$ and calculate $K_M := \ker_{\mathbb{Z}_N}(E_M)$ in Howell matrix form;
  3. Return $K_M$.
```

Because E_M has 2^{r+k} rows, the complexity of the logical identity algorithm is highly sensitive to the number of X-checks r and logical qubits k. However, due to proposition E.13 of [14], we only need to consider $\mathbf{e_{uv}}$ where $\mathrm{wt}(\mathbf{u}) + \mathrm{wt}(\mathbf{v}) \leqslant t$ to determine the logical identity group up to level t of the Clifford hierarchy. Hence, we only require $\left[{r+k \atop t}\right] := \sum_{0\leqslant j\leqslant t} \binom{r+k}{j}$ rows from E_M. Hence, the dimensions of the key matrix E_M scale as $\mathcal{O}((k+r)^t \times n)$. As we require only a single kernel calculation for the algorithm the time complexity as defined in table 1 is $\mathcal{O}((k+r)^t n^2)$.

Example 3.1 (logical identity algorithm—hypercube code). In this example, based on [22, 23] and illustrated in Figure 2, qubits reside on the eight vertices of a cube. The single X-check is the all-ones vector indicating an X operator on all vertices of the cube:

$$S_X = \begin{pmatrix}11111111\end{pmatrix}. \tag{18}$$

The three X-logicals are weight four vectors associated with three faces meeting at a point which we write in the notation of section 2.3 as follows:

$$L_X = \begin{pmatrix}01010101\\00110011\\00001111\end{pmatrix}. \tag{19}$$

We calculate the Z-checks by applying equation (6) and find that the Z-checks also correspond to faces:

$$S_Z := \ker_{\mathbb{Z}_2}\begin{pmatrix}S_X\\L_X\end{pmatrix} = \begin{pmatrix}10010110\\01010101\\00110011\\00001111\end{pmatrix}. \tag{20}$$

This process is exactly the same as finding the diagonal logical identities at level $t = 1$ as outlined in section 3.1. In this case, E_M has $r + k = 1 + 3 = 4$ rows and the logical identities are the kernel of E_M modulo 2. Now applying the logical identity algorithm at level $t = 3$, E_M has 15 rows representing the sum modulo 2 of up to three rows from S_X and L_X. Taking the kernel of E_M modulo $N = 2^3 = 8$, we find:

$$K_M := \ker_{\mathbb{Z}_8}(E_M) = \begin{pmatrix}22222222\\04040404\\00440044\\00004444\end{pmatrix}. \tag{21}$$

The rows of K_M are the Z-components of diagonal XP operators which act as logical identities, and form a generating set of all such operators of precision N. For instance, the operator $XP_8(0|0|22222222) = S^{\otimes 8}$ acts as a logical identity, but is not in the stabiliser group $\langle \mathbf{S}_X, \mathbf{S}_Z\rangle$. An interactive version of this example is in the linked Jupyter notebook.

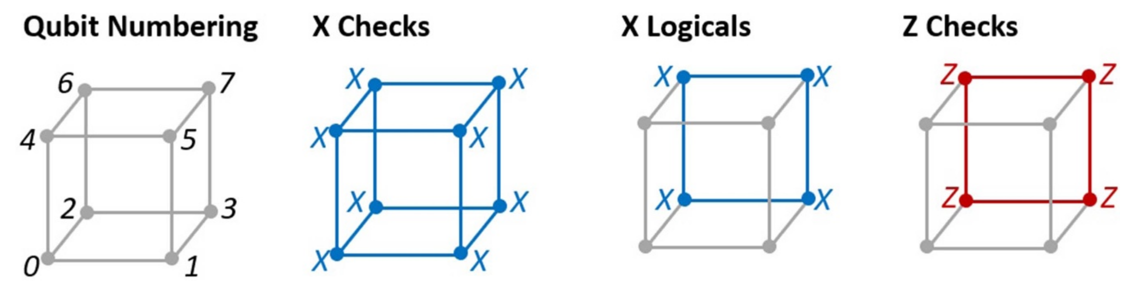


Figure 2. Hypercube code of dimension 3: qubits reside on the vertices of a cube. The blue-coloured X-logicals are associated with the 2D faces, whilst the X-check is associated with the single 3D volume. The red-coloured Z-checks are associated with the 2D faces.

3.2. Algorithm 2. search for diagonal XP operator by logical action

We now demonstrate a method that searches for diagonal logical operators of XP form with a desired action. Aside from verifying if a CSS code has a transversal implementation of a particular logical operator, this is a useful method for cross-checking other algorithms.

Algorithm 2. Search for diagonal XP operator by logical action.

```
Input:
  1. The $X$-checks $S_X$ and $X$-logicals $L_X$ of a CSS code (section 2.3);
  2. A level-$t$ controlled-phase operator $B$ on $k$ qubits (section 2.1) such that $B|0\rangle = |0\rangle$.
Output: a diagonal XP operator of precision $N = 2^t$ which acts as a logical $B$ operator or FALSE if this is not possible.
Method:
  1. For $\mathbf{v} \in \mathbb{Z}_2^k$ calculate the phase $\mathbf{q_v} \in \mathbb{Z}_N$ such that $B|\mathbf{v}\rangle = \omega^{2\mathbf{q_v}}|\mathbf{v}\rangle$;
  2. Form the matrix $E_\mathrm{B}$ that has rows of form $(-\mathbf{q_v}|\mathbf{e_{uv}})$ where $\mathbf{e_{uv}} := \mathbf{u}S_X + \mathbf{v}L_X$;
  3. Calculate the kernel $K_\mathrm{B} := \ker_{\mathbb{Z}_N}(E_\mathrm{B})$;
  4. If there is an element $(1|\mathbf{z}) \in K_\mathrm{B}$ then $\mathbf{z}$ is the $Z$-component of a logical $B$ operator $\overline{B} := XP_N(0|0|\mathbf{z})$. This is
     because $(1|\mathbf{z}) \cdot (-\mathbf{q_v}|\mathbf{e_{uv}}) = 0 \mod N \iff \mathbf{e_{uv}} \cdot \mathbf{z} = \mathbf{q_v} \mod N$ for all $\mathbf{e_{uv}}$, which corresponds to the action
     of a logical $B$ operator on the codewords $|\mathbf{v}\rangle_\mathrm{L}$.
```

The above algorithm requires that $B|0\rangle = |0\rangle$. If this is not the case, let $B|0\rangle = \omega^p|0\rangle$, run the algorithm using $B' := \omega^{-p}B$ and adjust for phase on the result. The results of the algorithm are dependent on the choice of basis for the span $\langle L_X\rangle$ (see section 2.3).

The logical action search algorithm involves finding the kernel of a matrix E_B of dimension $2^{r+k} \times (n+1)$. Hence the complexity of the algorithm is sensitive to the number of logical qubits k and independent X-checks r, but can be reduced as follows. Due to proposition B.1, where $N = 2^t$ the dot product $\mathbf{e_{uv}} \cdot \mathbf{z}$ can always be written as a $\mathbb{Z}_N$ linear combination of terms of form $\mathbf{e_{u'v'}} \cdot \mathbf{z}$ where $\mathrm{wt}(\mathbf{u}') + \mathrm{wt}(\mathbf{u}') \leqslant t$. Hence, we only need to consider $\mathbf{e_{uv}}$ where $\mathrm{wt}(\mathbf{u}) + \mathrm{wt}(\mathbf{v}) \leqslant t$ and $\mathbf{q_v}$ where $\mathrm{wt}(\mathbf{v}) \leqslant t$. The number of rows required in E_B is therefore $\left[{k+r \atop t}\right]$ where $\left[{r \atop t}\right] := \sum_{0 \leqslant j \leqslant t} \binom{r}{j}$. Hence, the dimensions of the key matrix E_B scale as $\mathcal{O}((k+r)^t \times n)$ and as we require only a single kernel calculation for the algorithm the time complexity as defined in table 1 is $\mathcal{O}((k+r)^t n^2)$.

Example 3.2 (search for diagonal XP operator by logical action). The linked Jupyter notebook illustrates the operation of the search algorithm on the hypercube code of example 3.1. Users can enter the desired logical operator to search for in text form—for example `CZ[1,2]`, `S[1]` or `CCZ[0,1,2]`. The script either returns a diagonal XP operator with the desired logical action, or `FALSE` if there is no such operator. We find logical operators $\overline{CZ_{12}} = \mathrm{XP}_8(0|0|02060602)$ and $\overline{CCZ_{012}} = \mathrm{XP}_8(0|0|13313113)$ but no solutions for transversal logical S operators.

3.3. Logical operator test for diagonal XP operators

We now present an efficient method for determining whether a given diagonal XP operator acts as a logical operator on a CSS code, which relies on a commutator property of logical operators. This is used to find a generating set of all diagonal logical XP operators of a given precision and to check the results of other algorithms.

Due to proposition E.2 of [14], an XP operator B acts as a logical operator on the codespace if and only if the group commutator with any logical identity A is again an element of the logical identity group $\mathcal{I}_{\text{XP}}$ (see 2.6). That is:

$$[[A,B]] := ABA^{-1}B^{-1} \in \mathcal{I}_{\text{XP}}, \forall A \in \mathcal{I}_{\text{XP}}. \tag{22}$$

When $B := \text{XP}_N(0|0|\mathbf{z})$ is diagonal and $A := \text{XP}_N(0|\mathbf{x}|0)$ is non-diagonal, by applying the COMM rule of table 4 in [14] we have:

$$[[A,B]] = \text{XP}_N(2\mathbf{x}\cdot\mathbf{z}|0|-2\mathbf{xz}). \tag{23}$$

As B is a diagonal operator, we only need to consider commutators with non-diagonal elements of the logical identity group. In proposition B.2 we show that this reduces to finding $\mathbf{z} \in \mathbb{Z}_N^n$ such that for all X-checks $\mathbf{x}$, both $\mathbf{x}\cdot\mathbf{z} = 0 \mod N$ and $2\mathbf{xz} \in \langle K_{\text{M}}\rangle_{\mathbb{Z}_N}$ where K_{M} is a generating set of Z-components of the diagonal logical identities as defined in section 3.1 and $\langle K_{\text{M}}\rangle_{\mathbb{Z}_N}$ the row span of K_{M} over $\mathbb{Z}_N$.

As $2\mathbf{xz}$ and N are both divisible by 2, we apply the method of section 2.5 and see that the group commutator must be at most a level $t-1$ Clifford hierarchy operator. For instance, for $t=2, N=4$ logical operators must commute up to level $t=1, N=2$ logical identities which are the Z-checks (see example 3.1). This observation either eliminates the need to calculate the logical identities (for $t \leqslant 2$) or reduces the complexity of calculating them (the number of rows in the matrix E_{M} of section 3.1 is a polynomial of degree t).

The dimensions of the matrix K_{M} scale as $\mathcal{O}(n\times n)$ and checking whether a vector is in the span of $\langle K_{\text{M}}\rangle$ involves the calculation of a Howell normal form. We require $\mathcal{O}(r)$ characteristic matrix operations for the algorithm where r is the number of independent X-checks, so the time complexity is $\mathcal{O}(rn^3)$. We may need to first run the diagonal logical identity algorithm of section 3.1 at level $t-1$.

Algorithm 3. Logical operator test for diagonal XP operators.

Input:
1. The X-checks S_X of a CSS code (section 2.3);
2. The matrix K_{M} corresponding to the Z-components of the level $t-1$ diagonal logical identity generators (section 3.1);
3. A diagonal XP operator $B = \text{XP}_N(0|0|\mathbf{z})$ on n qubits of precision $N = 2^t$ (section 2.5).

Output: `TRUE` if B acts as a logical operator on the code or `FALSE` otherwise.

Method:
```
1. For each row $\mathbf{x}$ of $S_X$:
    (a) Check if $\mathbf{x}\cdot\mathbf{z} = 0 \mod N$; and
    (b) Check if $2\mathbf{xz}$ is in the rowspan of $K_{\text{M}}$ over $\mathbb{Z}_N$;
    (c) If either is not the case, return FALSE.
2. Return TRUE.
```

Example 3.3 (logical operator test). In this example, we apply the logical operator test to the logical $\overline{CZ_{12}}$ found for the hypercube code in example 3.2. As $\overline{CZ_{12}} := \text{XP}_8(0|0|02060602)$, we let $\mathbf{z} = 02060602$. Let $\mathbf{x} = 11111111$ corresponding to the single X-check. We calculate the group commutator $C := (2\mathbf{x}\cdot\mathbf{z}|0|-2\mathbf{xz})$. We find that $\mathbf{x}\cdot\mathbf{z} = 16 = 0 \mod 8$ and $-2\mathbf{xz} = 04040404 \mod 8$. Referring to example 3.1, we see that this vector is a row of K_{M}. As both $\mathbf{x}\cdot\mathbf{z} = 0 \mod 8$ and $-2\mathbf{xz} \in \langle K_{\text{M}}\rangle_{\mathbb{Z}_N}$, C is a logical identity. Accordingly, we have verified that $\overline{CZ_{12}}$ is a diagonal logical operator on the code. Applying the method of section 2.5, we note that $\overline{CZ_{12}}$ is at level 2 of the Clifford hierarchy and the group commutator C is at level 1.

3.4. Diagonal logical XP operator group generators

We now show how to apply the test for diagonal logical XP operators of section 3.3 to find all diagonal logical operators of XP form for a CSS code.

Algorithm 4. Diagonal logical XP operator group generators.

```
Input:
  1. The $X$-checks $S_X$ of a CSS code (section 2.3);
  2. The desired level of the Clifford hierarchy $t$ (section 2.1);
  3. The matrix $K_\mathrm{M}$ corresponding to the $Z$-components of the level $t-1$ diagonal logical identity generators
     (section 3.1).
Output: a matrix $K_\mathrm{L}$ over $\mathbb{Z}_N$ representing the $Z$-components of a generating set of diagonal logical operators
        of XP form (section 2.6).
Method:
  1. For each $X$-check $\mathbf{x} \in S_X$, find solutions $\mathbf{z} \in \mathbb{Z}_N^n$ such that both $\mathbf{x} \cdot \mathbf{z} = 0$ and $2\mathbf{xz} \in \langle K_\mathrm{M} \rangle_{\mathbb{Z}_N}$. Details of solving
     within these constraints are set out in section B.3. Denote the solutions $\mathrm{Comm}_N(K_\mathrm{M}, \mathbf{x})$;
  2. Find the intersection of all such solution sets $K_\mathrm{L} := \bigcap_{\mathbf{x} \in S_X} \mathrm{Comm}_N(K_\mathrm{M}, \mathbf{x})$. The method for determining
     intersections of spans over $\mathbb{Z}_N$ is covered in appendix A.4 of [14];
  3. Return $K_\mathrm{L}$.
```

The rows of K_L correspond to the Z-components of a generating set of the logical XP group (section 2.6), which includes the logical identity XP group. Determining the logical action of the operators is discussed in section 3.5.

The dimensions of the key matrices $\mathrm{Comm}_N(K_\mathrm{M}, \mathbf{x})$ scale as $\mathcal{O}(n \times n)$ and determining intersections of spans involves the calculation of a matrix kernel. We require $\mathcal{O}(r)$ characteristic matrix operations for the algorithm where r is the number of independent X-checks giving a time complexity of $\mathcal{O}(rn^3)$.

3.5. Determine action of diagonal logical XP operator

Here we demonstrate an algorithm expressing the action of a diagonal logical XP operator in terms of logical controlled-phase operators. This is important because the algorithm in section 3.4 does not yield any information on the action of the resulting diagonal logical operators.

Algorithm 5. Determine action of diagonal logical XP operator.

```
Input:
  1. The $X$-logicals $L_X$ of a CSS code (section 2.3) with $k$ logical qubits;
  2. A diagonal XP operator $\overline{B}$ of precision $N := 2^t$ that acts as a logical operator on the code (section 2.4).
Output: a diagonal Clifford hierarchy operator $B$ on $k$ qubits representing the logical action of $\overline{B}$.
Method:
  1. Let $V := \{\mathbf{v} \in \mathbb{Z}_2^n : \mathrm{wt}(\mathbf{v}) \leqslant t\}$;
  2. For each $\mathbf{v} \in V$, calculate $\mathbf{q_v}$ such that $\overline{B}|\mathbf{v}L_X\rangle = \omega^{\mathbf{q_v}}|\mathbf{v}L_X\rangle$;
  3. Loop over each $\mathbf{v} \in V$ ordered by weight. For any $\mathbf{v} \preccurlyeq \mathbf{u} \in V \setminus \{\mathbf{v}\}$, update $\mathbf{q_u} := (\mathbf{q_u} - \mathbf{q_v}) \mod 2N$;
  4. Return $B := \prod_{\mathbf{v} \in V} CP_N(\mathbf{q_v}, \mathbf{v})$ in terms of the vector form of controlled-phase operators of section 2.2.
```

The above algorithm involves calculating $\mathcal{O}(k^t)$ phase components $\mathbf{q_v}$, and this is sufficient due to proposition B.1. Hence the size of the matrix required to calculate the phase components is $\mathcal{O}(k^t \times n)$. The algorithm involves a single pass through the list of phases, so we require $\mathcal{O}(1)$ matrix operations for the algorithm giving time complexity of $\mathcal{O}(k^t n^2)$. A naive approach which calculates the phase applied to each codeword would involve calculating $\mathcal{O}(2^k)$ such phase components, and would be impractical for CSS codes with a large number of logical qubits.

The results of the algorithm are dependent on the choice of basis for the span $\langle L_X \rangle$ (see section 2.3).

Example 3.4 (action of diagonal logical XP operators—hypercube codes). In this example, we apply the method of section 3.4 to the hypercube code of example 3.1 at level $t = 3$. The output of the method of section 3.4 is a set of length 8 vectors over $\mathbb{Z}_8$ corresponding to Z-components of diagonal logical XP operators. Using the method of section 3.5, we obtain the following list of logical actions corresponding to the Z-components:

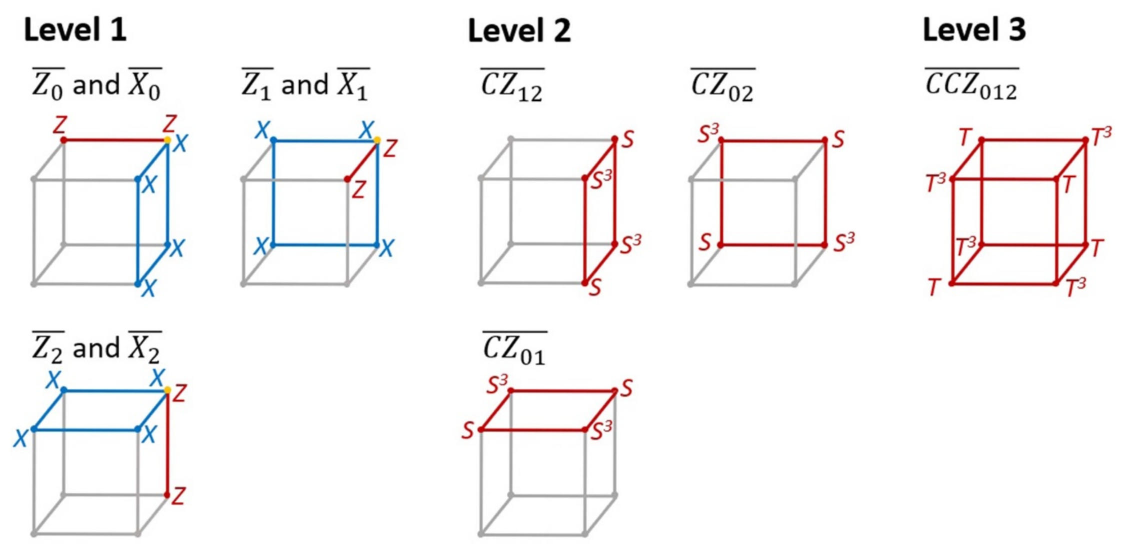

Level 1
$\overline{Z_0}$ and $\overline{X_0}$
$\overline{Z_1}$ and $\overline{X_1}$
$\overline{Z_2}$ and $\overline{X_2}$
Level 2
$\overline{CZ_{12}}$
$\overline{CZ_{02}}$
$\overline{CZ_{01}}$
Level 3
$\overline{CCZ_{012}}$

Figure 3. Diagonal logical operators of hypercube code of dimension 3: logical XP operators returned by the method of section 3.4 are plotted on the cube. We note that Clifford hierarchy level 1 logical Z operators have support on 1D edges, level 2 *CZ* operators have support on 2D faces and level 3 *CCZ* operators have support on the entire 3D cube.

z	Logical action	Clifford level
00000044	Z_0	1
00000404	Z_1	1
00040004	Z_2	1
00002662	CZ_{01}	2
00260062	CZ_{02}	2
02060602	CZ_{12}	2
13313113	CCZ_{012}	3

In figure 3 we display the resulting logical operators on the cube and notice that that the Clifford hierarchy level of the logical operator corresponds to the dimension of the support of the operator. An interactive version of this example is available in the linked Jupyter notebook.

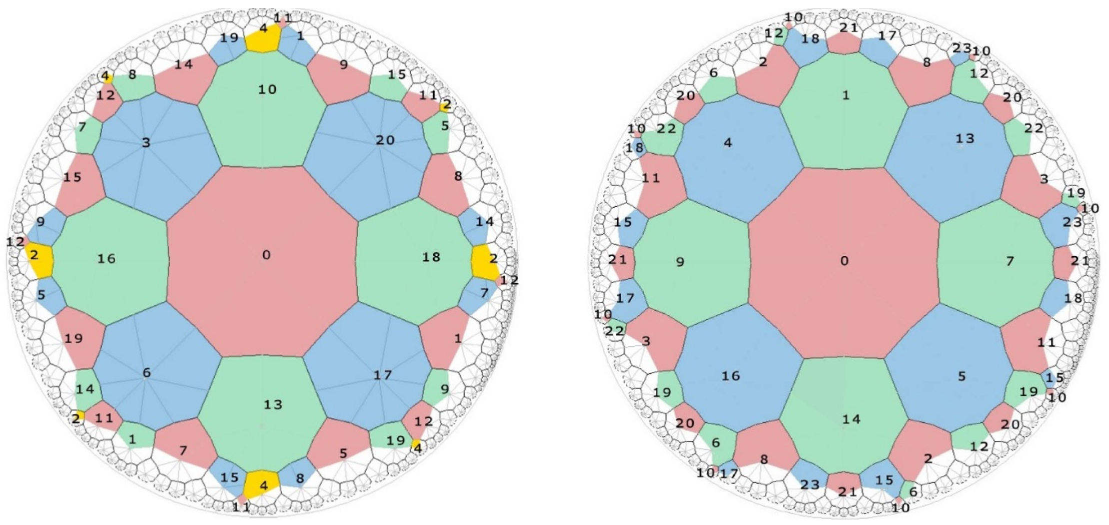
Example 3.5 (hyperbolic quantum color codes and poset codes). In the linked Jupyter notebook, we illustrate the application of the method of section 3.4 to codes that have a large number of logical qubits. We choose examples of self-orthogonal codes that are known to have transversal implementations of diagonal level 2 Clifford hierarchy logical operators.

Hyperbolic quantum colour codes [24] involve constructing codes from tessellations of the 2D hyperbolic plane. The tessellations are formed from polygons with an even number of edges, and each vertex is shared by three such polygons. We place a qubit on each vertex of the tessellation. For each polygonal face, we have an X-check corresponding to the adjacent vertices. The Z-checks are the same as the X-checks. Applying the method of section 3.4, we find that the codes have a transversal level 2 logical operator with action which can be expressed as a product of controlled-Z operators. We illustrate two examples of such tesselations in figure 4.

There are various methods in the literature for constructing classical self-orthogonal codes and these can also be used to make quantum codes with Z-checks which are the same as the X-checks which we expect to have transversal level 2 diagonal logical operators. In [25], self-orthogonal codes are constructed from partially ordered sets (posets). Analysing poset codes using our methods, we see that they have transversal level 2 logical operators with actions which can be expressed as products of S and CZ operators.

Example 3.6 (triorthogonal codes). For triorthogonal codes [10], there is always a logical operator of form $\overline{T^{\otimes k}} := UT^{\otimes n}$ where U is a product of CZ and S operators and k is the number of logical qubits of the code. In the linked Jupyter notebook, we apply the method of section 3.4 to find generating sets of diagonal logical operators for the 38 triorthogonal code classes in table II of [27]. In this example, we consider codes with $k = 3$ logical qubits (this choice can be modified by the user).

Applying our method, we see that the logical operator structure of triorthogonal codes varies widely. In some cases, the code has a transversal logical $\overline{T_i}$ operator for each logical qubit $0 \leqslant i < k$. For most of the



(a) [[56,14,6]] Code (b) [[64,20,4]] Code

Figure 4. {8,3} Hyperbolic colour codes: above are tessellations corresponding to two hyperbolic colour codes from example 3.5. The [[56,14,6]] code is not globally three-colourable as there is no valid colour assignment for faces 2 and 4 in the diagram above. Each code has a transversal level 2 diagonal logical operator whose action is a product of logical *CZ* operators.

codes, we find a logical $\overline{T^{\otimes k}}$ operator of XP form. The exceptions are codes which require the application of *CZ* operators to form $\overline{T^{\otimes k}}$, and so would not be identified by our method. We do not see any instances of logical *CCZ* or *CS* operators.

4. Transversal logical operators composed of multi-qubit controlled-phase gates

In the previous section, we have shown how to find a generating set of all logical operators of a CSS code that can be constructed from single-qubit phase gates at any level of the Clifford hierarchy. This relied on representing operators composed of single-qubit phase gates as diagonal XP operators. In this section, we show how to find all transversal (depth-one) logical operators of a CSS code composed of multi-qubit controlled-phase gates. The method relies on representing controlled-phase operators acting on a codespace as diagonal XP operators acting on a larger Hilbert space via an embedding operator.

The structure of this section is as follows. We first introduce phase-rotation gates and discuss some of their elementary properties. We then prove a duality result that transforms controlled-phase operators to phase-rotation operators and vice versa. Hence phase-rotation gates are an alternative generating set for diagonal Clifford hierarchy gates. We then describe an embedding operator from the codespace into a larger Hilbert space such that phase-rotation operators in the codespace correspond to diagonal XP operators in the embedded codespace. As a result, any diagonal Clifford hierarchy operator can be represented as a diagonal XP operator in the embedded codespace.

Finally, we demonstrate an algorithm that searches for transversal logical operators composed of single- and multi-qubit controlled-phase gates for a given CSS code. Such implementations are depth one and use operators with bounded support size and so have fault-tolerant properties. Logical operators of this type have recently been studied in [12, 13, 28] and we provide examples of the application of the algorithm to codes in these papers.

4.1. Phase-rotation operators

Phase-rotation operators are single or multi-qubit diagonal gates that form an alternative generating set for the diagonal Clifford hierarchy operators of section 2.1. Phase-rotation operators are defined as follows. Let $A := \mathrm{XP}_2(0|0|\mathbf{v})$ be a tensor product of Z operators and let $\omega := \exp(\pi i/N)$. Let $A_{\pm 1} := (I \pm A)/2$ be the projectors onto the ± 1 eigenspaces of A and let $q \in \mathbb{Z}_{2N}$. The phase-rotation operator is:

$$\mathrm{RP}_N(q,\mathbf{v}) = \exp\left(\frac{q\pi i}{N} A_{-1}\right). \tag{24}$$

This form is similar to the Pauli product rotations of [29] and operators of this type arise as fundamental gates in nuclear magnetic resonance (NMR) [30] and quantum dot systems [31]. In proposition A.4, we show that the action of $\mathrm{RP}_N(q,\mathbf{v})$ on the computational basis element $|\mathbf{e}\rangle$ for $\mathbf{e} \in \mathbb{Z}_2^n$ is:

$$\mathrm{RP}_N(q,\mathbf{v})|\mathbf{e}\rangle = \begin{cases}\omega^q|\mathbf{e}\rangle & \text{if } \mathbf{e}\cdot\mathbf{v} \mod 2 = 1\\ |\mathbf{e}\rangle & \text{otherwise.}\end{cases} \tag{25}$$

We can express the phase applied more concisely as follows:

$$\mathrm{RP}_N(q,\mathbf{v})|\mathbf{e}\rangle = \omega^{q\cdot s_{\mathbf{v}}(\mathbf{e})}|\mathbf{e}\rangle \text{where } s_{\mathbf{v}}(\mathbf{e}) := \bigoplus_{i\preccurlyeq\mathbf{v}} \mathbf{e}[i]. \tag{26}$$

Single qubit phase gates of precision N in this notation are of form $P_i = \mathrm{RP}_N(2,\mathbf{b}_i^n)$ where $\mathbf{b}_i^n$ is the length n binary vector which is all zero, apart from component i which is one.

Where the precision and number of qubits are fixed, we use a more concise notation for phase-rotation operators analogous to the notation for controlled-phase operators. For example, on $n=3$ qubits, the following are examples of precision $N=8$ operators: $\mathrm{RRZ}_{012} := \mathrm{RP}_8(8,111), \mathrm{RS}_{01} := \mathrm{RP}_8(4,110)$, $T_0 = \mathrm{RP}_8(2,100)$.

4.2. Duality of controlled-phase and phase-rotation operators

In proposition A.5, we prove a duality result that allows us to convert vector form controlled-phase operators to products of phase-rotation operators and vice versa:

$$\mathrm{CP}_N\left(2^{\mathrm{wt}(\mathbf{v})},\mathbf{v}\right) = \prod_{0\neq\mathbf{u}\preccurlyeq\mathbf{v}} \mathrm{RP}_N\left(2\cdot(-1)^{\mathrm{wt}(\mathbf{u})-1},\mathbf{u}\right); \tag{27}$$

$$\mathrm{RP}_N(2,\mathbf{v}) = \prod_{0\neq\mathbf{u}\preccurlyeq\mathbf{v}} \mathrm{CP}_N\left(2\cdot(-2)^{\mathrm{wt}(\mathbf{u})-1},\mathbf{u}\right). \tag{28}$$

In section 2.2, we saw that operators of form $\mathrm{CP}_N(2^{\mathrm{wt}(\mathbf{v})},\mathbf{v})$ with $\mathrm{wt}(\mathbf{v}) \leqslant t$ and $N := 2^t$ generate the level t diagonal Clifford hierarchy operators. As a consequence of the duality result, phase-rotation operators of form $\mathrm{RP}_N(2,\mathbf{v})$ where $\mathrm{wt}(\mathbf{v}) \leqslant t$ are an alternative generating set. In the linked Jupyter notebook we show that $\mathrm{RS}_{01} = CZ_{01}S_1S_2 = \mathrm{RS}_{01}^3 Z_1 Z_2$ by applying the duality result twice—hence phase-rotation operators may have more than one vector representation.

4.3. Embedded code method

The embedded code method involves constructing an embedding operator on the codespace of a CSS code such that phase-rotation operators in the original codespace correspond to diagonal XP operators in the embedded codespace. The embedding technique is similar to the one used to represent weighted hypergraph states in section 5.4 of [14]. We first define the embedding operator in terms of its action on computational basis states, then show how to extend it to phase rotation operators and strings of Pauli X operators. As an example, we show how the embedding operator transforms repetition codes.

4.3.1. Action of embedding operator on computational basis states and CSS codespaces

Let M_t^n be the matrix whose rows are the binary vectors of length n of weight between 1 and t. Let V be a matrix whose rows are a subset of the rows of M_t^n. We define the embedding operator $\mathcal{E}_V : \mathcal{H}_2^n \to \mathcal{H}_2^{|V|}$ that has the following action on computational basis vectors $|\mathbf{e}\rangle, \mathbf{e}\in\mathbb{Z}_2^n$:

$$\mathcal{E}_V|\mathbf{e}\rangle = |\mathbf{e}V^{\mathrm{T}} \mathrm{mod}\ 2\rangle. \tag{29}$$

Now let S_X, L_X be the X-checks and X-logicals of a CSS code $\mathbf{C}$ on n qubits (see section 2.3). The image of the codespace of $\mathbf{C}$ under $\mathcal{E}_V$ is the codespace of the embedded code $\mathbf{C}_V$ defined as follows:

- X-checks $S_X^V := S_X V^{\mathrm{T}}$
- X-logicals $L_X^V := L_X V^{\mathrm{T}}$
- Z-checks $S_Z^V := \ker_{\mathbb{Z}_2}\begin{pmatrix} S_X^V \\ L_X^V\end{pmatrix}$.

Providing V is full rank, the X-checks and X-logicals of the embedded code are independent (for instance if V includes all rows of I_n). We will show that phase-rotation operators acting on the codespace correspond to diagonal XP operators in the embedded codespace. Because operators of form $\mathrm{RP}_N(2,\mathbf{v})$ for $\mathbf{v}\in M_t^n$ and $N=2^t$ generate all controlled-phase operators of level t on n qubits (see section 4.2), choosing $V = M_t^n$ allows any such operator to be represented. By limiting V to a subset of M_t^n, we can place restrictions on the phase-rotation operators we wish to work with in the embedded codespace. For instance, we can allow only nearest neighbour interactions for a lattice-based code or cater for ZX symmetries and qubit partitions as discussed in [12, 13].

4.3.2. Action of embedding operator on phase-rotation and Pauli X operators

We now demonstrate an extension of the embedding operator $\mathcal{E}_V$ to phase-rotation and Pauli X operators which acts as a group homomorphism. A group homomorphism must respect commutation relations, and this is much simpler to achieve for phase-rotation operators than for controlled-phase operators. In proposition A.7, we prove the following commutation relation for controlled-phase and Pauli X operators:

$$\mathrm{CP}_N(q,\mathbf{v})X_i = \begin{cases} X_i\mathrm{CP}_N(-q,\mathbf{v})\,\mathrm{CP}_N(q,\mathbf{v}\oplus\mathbf{b}_i^n) & \text{if } \mathbf{v}[i]=1 \\ X_i\mathrm{CP}_N(q,\mathbf{v}) & \text{otherwise.}\end{cases} \tag{30}$$

In equation (30), $\mathbf{b}_i^n$ is the binary vector of length n which is zero apart from entry i which is one. Extending this to arbitrary strings of X operators we obtain the following:

$$\mathrm{CP}_N(q,\mathbf{v})\,\mathrm{XP}_2(0|\mathbf{x}|0) = \mathrm{XP}_2(0|\mathbf{x}|0)\prod_{0\preccurlyeq\mathbf{u}\preccurlyeq\mathbf{xv}}\mathrm{CP}_N\left(q\cdot(-1)^{\mathrm{wt}(\mathbf{xv})+\mathrm{wt}(\mathbf{u})},\mathbf{v}\oplus\mathbf{u}\right). \tag{31}$$

In proposition A.6, we prove the much simpler commutation relation for phase-rotation operators which corresponds closely to the commutation relation for XP operators in equation (15):

$$\mathrm{RP}_N(q,\mathbf{v})X_i = \begin{cases}\omega^q X_i\mathrm{RP}_N(-q,\mathbf{v}) & \text{if } \mathbf{v}[i]=1\\ X_i\,\mathrm{RP}_N(q,\mathbf{v}) & \text{otherwise.}\end{cases} \tag{32}$$

The relation in equation (32) also implies that for any $V\subset M_t^n$, we have closure under conjugation with any Pauli X string, which is not the case for controlled-phase operators.

Now consider the group $\mathcal{XRP}_N^V$ generated by operators of form ωI, X_i and $\mathrm{RP}_N(2,\mathbf{v})$ for $\mathbf{v}$ a row of V. Elements of $\mathcal{XRP}_N^V$ can be written in terms of components $p\in\mathbb{Z}_{2N}$, $\mathbf{x}\in\mathbb{Z}_2^n$ and the vector $\mathbf{q}\in\mathbb{Z}_N^{|V|}$ indexed by rows of V such that:

$$\mathrm{XRP}_N^V(p|\mathbf{x}|\mathbf{q}) := \omega^p\prod_{0\leqslant i<n}X_i^{\mathbf{x}[i]}\prod_{\mathbf{v}\in V}\mathrm{RP}_N(2\mathbf{q}[\mathbf{v}],\mathbf{v}). \tag{33}$$

We define an embedding map for XRP operators with respect to V as follows:

$$\mathcal{E}_V\left(\mathrm{XRP}_N^V(p|\mathbf{x}|\mathbf{q})\right) := XP_N\left(p|\mathbf{x}V^{\mathrm{T}}|\mathbf{q}\right). \tag{34}$$

In proposition B.4, we show that the embedding operator $\mathcal{E}_V$ respects group operations and so acts as a group homomorphism. As a result, we can use the diagonal logical identity and logical operator algorithm in sections 3.1 and 3.4 to find logical operators in the embedded codespace. The results can be interpreted as phase-rotation operators in the original codespace. One application of this method is to better understand what kinds of coherent noise a CSS code is inherently protected against as in [32]. The logical identity group of the embedded code represents the correlated noise that the code protects against up to configurable constraints (for example connectivity and the level of Clifford hierarchy).

Another consequence of the embedded code technique is that for CSS codes, any logical operator $\overline{B}$ which can be written as a product of single or multi-qubit phase rotation gates has a logical action in the Clifford hierarchy, even where the precision N is not a power of 2. To see this, note that the proof of proposition B.4 does not rely on N being a power of 2. Hence, we can construct the embedded code using the method outlined above. The embedded code is a CSS code with a transversal logical operator $\overline{B}_V$ composed of single-qubit phase gates of precision N. Due to [21], $\overline{B}_V$ is a product of Clifford hierarchy phase and has logical action within the Clifford hierarchy. The result follows because the logical actions of $\overline{B}$ and $\overline{B}_V$ are the same.

Example 4.1 (embedding the repetition code). In this example we show how to construct an embedded code based on the repetition code. For example, let S_X be the check matrix of the classical repetition code on three bits and let L_X be a weight one vector. This forms a CSS code $\mathbf{C}$ with:

$$S_X := \begin{pmatrix}110\\011\end{pmatrix}, \tag{35}$$

$$L_X := \begin{pmatrix}001\end{pmatrix}. \tag{36}$$

Let $V := M_2^3$ be the matrix whose rows are binary vectors of length 3 and weight 1 or weight 2. The embedded code $\mathbf{C}_V$ is defined by setting $S_X^V := S_X V^\mathrm{T}$ and $L_X^V := L_X V^\mathrm{T}$ so that:

$$V^\mathrm{T} := \begin{pmatrix} 100110 \\ 010101 \\ 001011 \end{pmatrix}, \tag{37}$$

$$S_X^V := S_X V^\mathrm{T} = \begin{pmatrix} 101101 \\ 011110 \end{pmatrix}, \tag{38}$$

$$L_X^V := L_X V^\mathrm{T} = \begin{pmatrix} 001011 \end{pmatrix}. \tag{39}$$

Applying the method of section 3.4, we find that the embedded code has a logical S operator given by $\overline{S}_V := \mathrm{XP}_4(0|0|113133) = S_0 S_1 S_2^3 S_3 S_4^3 S_5^3$. In the original codespace, this corresponds to the following product of phase-rotation gates (section 4.1):

$$\overline{S} := \mathrm{RP}_4(2,100)\,\mathrm{RP}_4(2,010)\,\mathrm{RP}_4(6,001)\,\mathrm{RP}_4(2,110)\,\mathrm{RP}_4(6,101)\,\mathrm{RP}_4(6,011). \tag{40}$$

In the linked Jupyter notebook, users can verify that using a repetition code on d bits and $V = M_t^d$ the matrix whose rows are binary vectors of length d of weight between 1 and t, the embedded code has a transversal logical phase gate at level t of the Clifford hierarchy.

4.4. Algorithm 6. depth-one logical operators

We now show how to find the transversal logical operators composed of single and multi-qubit diagonal Clifford hierarchy gates (i.e. depth-one circuit implementations where each physical qubit is involved in at most one gate) for a CSS code. It relies on the method of representing phase-rotation operators on a codespace as XP operators in an embedded codespace of section 4.3.

Algorithm 6. Depth-one logical operators.

Input:

1. The X-checks S_X and X-logicals L_X of a CSS code (section 2.3);
2. The desired level t of the Clifford hierarchy (section 2.1).

Output: a depth-one implementation of a logical controlled-phase operator at level t, or `FALSE` if there is no such implementation.

Method:

```
1. Use the embedding $V = M_t^n$—all binary vectors of length $n$ of weight between 1 and $t$;
2. For the embedded code $\mathbf{C}_V$ (section 4.3), calculate $K_\mathrm{L}$ the rows of which are the $Z$-components of a generating
   set of the diagonal logical XP operator group (section 3.4);
3. For each row of $K_\mathrm{L}$, determine the logical action and the level of the Clifford hierarchy (section 3.5);
4. From the rows of $K_\mathrm{L}$, choose a vector $\mathbf{z}$ corresponding to a logical operator at level $t$ of the Clifford hierarchy.
    If there is no such operator, return FALSE. Otherwise, perform the following steps:
   (a) Remove $\mathbf{z}$ from $K_\mathrm{L}$;
   (b) For each element $\mathbf{q}$ of the rowspan of $K_\mathrm{L}$ over $\mathbb{Z}_N$, check if $\mathbf{z}' := (\mathbf{q}+\mathbf{z}) \mod N$ represents a depth-one
       operator at level $t$ of the Clifford hierarchy (using the methods of sections 2.5 and 3.5). If so, return $\mathbf{z}'$;
   (c) If no depth-one operator is found, go to step 4.
```

When the CSS code has a known symmetry, we can search for depth-one logical operators more efficiently by modifying the embedding operator. The depth-one algorithm can take as input a permutation of the physical qubits in cycle form such that the cycles partition the n physical qubits. Let $\mathbf{c} = (c_1, c_2, \ldots, c_l)$ be a cycle in the permutation and let $\mathbf{b}_\mathbf{c}^n$ be the length n binary vector which is zero apart from the components $i \in \mathbf{c}$ that are one. The rows of the embedding matrix V are the vectors $\mathbf{b}_\mathbf{c}^n$ for the cycles c in the permutation.

The algorithm as outlined above yields logical operators composed of physical phase-rotation gates. To search for logical operators composed of controlled-phase gates, transform the matrix K_L by using the duality result of section 4.2. In this case, due to the commutation relation in equation (31), we need to ensure that for all $\mathbf{v} \in V$ any length n binary vector whose support is a subset of the support of $\mathbf{v}$ is also in V—that is if $\mathbf{v} \in V$ and $\mathbf{u} \preccurlyeq \mathbf{v}$ then $\mathbf{u} \in V$.

Note that M_t^n has $\left[{n \atop t}\right] := \sum_{1 \leqslant j \leqslant t} \binom{n}{j}$ rows. Hence, the dimensions of the key matrix K_L are $\mathcal{O}(n^t \times n^t)$. Where there are no depth-one logical operators at level t, the algorithm checks all possible linear combinations of K_L. Hence, the worst-case time complexity of the algorithm is exponential in the number of rows of K_L and we would generally apply the algorithm only to small codes of around 30 physical qubits. As an illustration, for $t = 2$ and $n = 30, M_t^n$ has 465 rows, but for $n = 100, M_t^n$ has 5050 rows. In appendix B.4 we describe a method for more efficiently exploring the search space.

Example 4.2 (depth-one algorithm). In the linked Jupyter notebook, we illustrate the depth-one search algorithm for small codes. For a given code and a desired level of the Clifford hierarchy t, the output is a logical operator with a depth-one circuit implementation whose logical action is at level t of the diagonal Clifford hierarchy, or FALSE if no such operator exists. This is done with no knowledge of the logical action of the operator or symmetries of the code. For example, we identify the depth-one implementation of the logical $\overline{S}_0\overline{S}_1^3$ of the 2D toric code as discussed in [11–13]. Users can also apply the algorithm to Bring's code which is a 30-qubit LDPC code discussed in [12] and various examples of morphed codes which are discussed in [28]. Users can also choose to use a known symmetry of the code to speed up the search—this can be used for instance to verify the partitioned logical operators of the symmetric hypergraph product codes of [13].

5. Other applications of embedded codes

In this section, we discuss other applications of the embedded code method of section 4.3. We first show that for any CSS code with k logical qubits and any diagonal Clifford hierarchy operator B on k qubits, we can write a closed-form expression for a logical $\overline{B}$ operator on the codespace composed of phase-rotation gates (see section 4.1). As a consequence, the embedded code has a logical B operator composed of single-qubit phase gates. This leads to a method of generating CSS codes that have transversal implementations of any desired diagonal logical Clifford hierarchy operator.

5.1. Canonical implementations of logical controlled-phase operators

Here, we show how to implement a desired logical controlled-phase operator on an arbitrary CSS code via a canonical form composed of the phase-rotation gates of section 4.1. We demonstrate implementations of logical S, T, CZ and CS operators using the 2D toric code as an example. As the canonical implementation is in terms of phase-rotation operators, we can apply the embedded code method of section 4.3 and implement the logical operator in the embedded codespace using single qubit phase gates. We use this fact to generate families of CSS codes that have transversal implementations of a desired logical controlled-phase operator using single-qubit phase gates. The methodology is illustrated in figure 5.

5.2. Canonical form for logical phase operators

In the proposition below, we show that logical phase operators have a particularly simple form in terms of the phase-rotation gates of section 4.1.

Proposition 5.1 (canonical logical *P* operator). *Let $\mathbf{z}_i \in \mathbb{Z}_2^n$ be the Z-component of a logical Z_i operator $\overline{Z_i} := \mathrm{XP}_2(0|0|\mathbf{z}_i)$. The operator $\overline{P_i} := \mathrm{RP}_N(2, \mathbf{z}_i)$ is a logical P_i operator.*

Proof. The action of a P_i operator on a computational basis element $|\mathbf{v}\rangle$ where $\mathbf{v} \in \mathbb{Z}_2^k$ can be written $P_i|\mathbf{v}\rangle = \omega^{2\mathbf{v}[i]}|\mathbf{v}\rangle$. Let $\mathcal{C} : \mathcal{H}_2^k \to \mathcal{H}_2^n$ be the encoding operator $\mathcal{C}|\mathbf{v}\rangle = |\mathbf{v}\rangle_\mathrm{L}$ for $\mathbf{v} \in \mathbb{Z}_2^k$. From equation (8), $\overline{P_i}$ is a logical P_i operator if $\overline{P_i}\mathcal{C} = \mathcal{C}P_i$. Hence:

$$\mathcal{C}P_i|\mathbf{v}\rangle = \omega^{2\mathbf{v}[i]}|\mathbf{v}\rangle_\mathrm{L} = \sum_{\mathbf{u}\in\mathbb{Z}_2^r} \omega^{2\mathbf{v}[i]}|\mathbf{e}_{\mathbf{uv}}\rangle \tag{41}$$

$$= \overline{P_i}|\mathbf{v}\rangle_\mathrm{L} = \sum_{\mathbf{u}\in\mathbb{Z}_2^r} \overline{P_i}|\mathbf{e}_{\mathbf{uv}}\rangle. \tag{42}$$

Hence, we require $\overline{P_i}|\mathbf{e}_{\mathbf{uv}}\rangle = \omega^{2\mathbf{v}[i]}|\mathbf{e}_{\mathbf{uv}}\rangle$. Set the precision $N = 2$, and we have $\overline{Z_i}|\mathbf{e}_{\mathbf{uv}}\rangle = (-1)^{\mathbf{v}[i]}|\mathbf{e}_{\mathbf{uv}}\rangle$. Applying equation (17), we have $\overline{Z_i}|\mathbf{e}_{\mathbf{uv}}\rangle = \mathrm{XP}_2(0|0|\mathbf{z}_i)|\mathbf{e}_{\mathbf{uv}}\rangle = (-1)^{\mathbf{e}_{\mathbf{uv}}\cdot\mathbf{z}_i}|\mathbf{e}_{\mathbf{uv}}\rangle$. Therefore, $\mathbf{e}_{\mathbf{uv}}\cdot\mathbf{z}_i \mod 2 = \mathbf{v}[i]$. Now consider the action of $\overline{P_i} := \mathrm{RP}_N(2, \mathbf{z}_i)$ on $|\mathbf{e}_{\mathbf{uv}}\rangle$ using proposition A.4:

$$\mathrm{RP}_N(2, \mathbf{z}_i)|\mathbf{e}_{\mathbf{uv}}\rangle = \omega^{2(\mathbf{e}_{\mathbf{uv}}\cdot\mathbf{z}_i \mod 2)}|\mathbf{e}_{\mathbf{uv}}\rangle = \omega^{2\mathbf{v}[i]}|\mathbf{e}_{\mathbf{uv}}\rangle, \tag{43}$$

as required for $\overline{P_i}$ to act as a logical P operator. □

Using the duality of RP and CP operators of section 4.2, we can write $\overline{P}_i$ as a product of CP gates:

$$\overline{P}_i := \mathrm{RP}_N(2, \mathbf{z}_i) = \prod_{0\neq\mathbf{u}\preccurlyeq\mathbf{z}_i} \mathrm{CP}_N\left(2\cdot(-2)^{\mathrm{wt}(\mathbf{u})-1}, \mathbf{u}\right). \tag{44}$$

As $N = 2^t$, any terms with $\mathrm{wt}(\mathbf{u}) > t$ disappear. Hence the support of the CP gates in the implementation are of maximum size t. The implementation may not be transversal, as a qubit may be acted upon by more than one gate.

$$\begin{tikzcd}
\mathcal{H}_2^k \arrow[r, "\mathcal{C}"] \arrow[d, "B"'] & \mathcal{H}_2^n \arrow[r, "\mathcal{E}_V"] \arrow[d, "\overline{B}"'] & \mathcal{H}_2^{|V|} \arrow[d, "\overline{B}_V"'] \\
\mathcal{H}_2^k \arrow[r, "\mathcal{C}"] & \mathcal{H}_2^n \arrow[r, "\mathcal{E}_V"] & \mathcal{H}_2^{|V|}
\end{tikzcd}$$

Figure 5. Logical operators of CSS codes and embedded codes: a CSS encoding maps k logical qubits into n physical qubits via $\mathcal{C}:\mathcal{H}_2^k\to\mathcal{H}_2^n$, which takes computational basis elements $|\mathbf{v}\rangle$ to codewords $|\mathbf{v}\rangle_L$. Consider a level-t controlled-phase operator B acting on $\mathcal{H}_2^k$. An operator $\overline{B}$ acting on $\mathcal{H}_2^n$ is a logical B operator if $\overline{B}\mathcal{C}=\mathcal{C}B$. We show how to construct a canonical logical B operator $\overline{B}$ from level-t phase-rotation gates. Let V be the matrix whose rows are length n binary vectors representing the support of the controlled-phase operators making up $\overline{B}$. The embedded codespace is formed by applying the embedding $\mathcal{E}_V:\mathcal{H}_2^n\to\mathcal{H}_2^{|V|}$ which takes the computational basis element $|\mathbf{e}\rangle$ to $|\mathbf{e}V^T \mod 2\rangle$. This enables us to construct a logical B operator $\overline{B}_V$ on the embedded codespace from single qubit phase gates.

Example 5.1 (logical phase operators of the 2D toric code). We illustrate the canonical form of logical controlled-phase operators by considering the 2D toric code. Using the XP operator notation of equation (16), let $Z_0 := \mathrm{XP}_2(0|0|\mathbf{z}_0), Z_1 := \mathrm{XP}_2(0|0|\mathbf{z}_1)$ be logical Z operators on logical qubit 0 and 1 respectively with $d := \mathrm{wt}(\mathbf{z}_0) = \mathrm{wt}(\mathbf{z}_1) \geqslant 3$. Applying equation (44) and using the notation of equation (1) for controlled-phase operators, the canonical forms for the logical S and T operators on qubit 0 are as follows:

$$\overline{S}_0 := \mathrm{RP}_4(2,\mathbf{z}_0) = \prod_{0\neq\mathbf{u}\preccurlyeq\mathbf{z}_0} \mathrm{CP}_4\left(2\cdot(-2)^{\mathrm{wt}(\mathbf{u})-1},\mathbf{u}\right) \tag{45}$$

$$= \prod_{\substack{\mathbf{u}\preccurlyeq\mathbf{z}_0\\ \mathrm{wt}(\mathbf{u})=2}} \mathrm{CP}_4(-4,\mathbf{u}) \prod_{\substack{\mathbf{u}\preccurlyeq\mathbf{z}_0\\ \mathrm{wt}(\mathbf{u})=1}} \mathrm{CP}_4(2,\mathbf{u}) \tag{46}$$

$$= \prod_{i<j\preccurlyeq\mathbf{z}_0} CZ_{ij} \prod_{i\preccurlyeq\mathbf{z}_0} S_i \tag{47}$$

$$\overline{T}_0 := \mathrm{RP}_8(2,\mathbf{z}_0) = \prod_{0\neq\mathbf{u}\preccurlyeq\mathbf{z}_0} \mathrm{CP}_8\left(2\cdot(-2)^{\mathrm{wt}(\mathbf{u})-1},\mathbf{u}\right) \tag{48}$$

$$= \prod_{\substack{\mathbf{u}\preccurlyeq\mathbf{z}_0\\ \mathrm{wt}(\mathbf{u})=3}} \mathrm{CP}_8(8,\mathbf{u}) \prod_{\substack{\mathbf{u}\preccurlyeq\mathbf{z}_0\\ \mathrm{wt}(\mathbf{u})=2}} \mathrm{CP}_8(-4,\mathbf{u}) \prod_{\substack{\mathbf{u}\preccurlyeq\mathbf{z}_0\\ \mathrm{wt}(\mathbf{u})=1}} \mathrm{CP}_8(2,\mathbf{u}) \tag{49}$$

$$= \prod_{i<j<k\preccurlyeq\mathbf{z}_0} CCZ_{ijk} \prod_{i<j\preccurlyeq\mathbf{z}_0} CS_{ij}^{-1} \prod_{i\preccurlyeq\mathbf{z}_0} T_i \tag{50}$$

These results hold for any CSS code with $\mathrm{wt}(\mathbf{z}_0) \geqslant 3$, as no other special properties of the toric code have been used.

5.3. Canonical form of logical phase-rotation and controlled-phase operators

We now generalise the method in section 5.2 and show how to implement logical phase-rotation operators for CSS codes using physical phase-rotation gates. Let L_Z be the $k\times n$ binary matrix representing logical Z operators such that $L_Z L_X^T \mod 2 = I_k$ where $k = |L_X|$. This means that $\mathrm{XP}_2(0|0|\mathbf{z}_i)$ anti-commutes with $XP_2(0|\mathbf{x}_j|0)$ if and only if $i=j$. Let $\mathbf{u}$ be a binary vector of length k. In proposition B.5 we show that the following is a logical phase-rotation operator:

$$\overline{\mathrm{RP}_N(2,\mathbf{u})} := \mathrm{RP}_N(2,\mathbf{u}L_Z) \tag{51}$$

By the duality result of section 4.2, we can write logical phase-rotation operators as follows:

$$\overline{\mathrm{CP}_N\left(2^{\mathrm{wt}(\mathbf{v})},\mathbf{v}\right)} := \prod_{0\neq\mathbf{u}\preccurlyeq\mathbf{v}} \overline{\mathrm{RP}_N\left(2\cdot(-1)^{\mathrm{wt}(\mathbf{u})-1},\mathbf{u}\right)} \tag{52}$$

$$= \prod_{0\neq\mathbf{u}\preccurlyeq\mathbf{v}} \mathrm{RP}_N\left(2\cdot(-1)^{\mathrm{wt}(\mathbf{u})-1},\mathbf{u}L_Z\right) \tag{53}$$

This in turn can be converted into products of physical controlled-phase gates by applying the duality result a second time.

Algorithm 7. Canonical logical controlled-phase operators.

Input:
1. The Z-logicals L_Z of a CSS code (see above);
2. A level-t diagonal Clifford hierarchy operator B on k qubits (section 2.1).

Output: a logical $\overline{B}$ operator (section 2.4) on the code composed of physical phase rotation gates (section 4.1) with maximum support size t.

Method:
1. Express $B = \prod_{\mathbf{u}} \mathrm{RP}_N(\mathbf{q}_{\mathbf{u}}, \mathbf{u})$ as a product of phase rotation gates using the duality result of section 4.2 where $N = 2^t$ and $\mathbf{u} \in \mathbb{Z}_2^k$;
2. The operator $\overline{B} = \prod_{\mathbf{u}} \mathrm{RP}_N(\mathbf{q}_{\mathbf{u}}, \mathbf{u}L_Z)$ is a logical B operator;
3. Apply the duality result of section 4.2 twice to express $\overline{B}$ as a product of phase-rotation gates of maximum support size t.

Example 5.2 (logical controlled-phase operators of toric code). We now demonstrate a canonical implementation of a logical CZ operator on the 2D toric code of example 5.1 composed of physical controlled-phase gates. Using equation (53) and the fact that $\mathrm{RP}_4(2, \mathbf{z}_0) = \prod_{i<j\preccurlyeq \mathbf{z}_0} CZ_{ij} \prod_{i\preccurlyeq \mathbf{z}_0} S_i$ from example 5.1:

$$\overline{CZ_{01}} := \overline{CP_4(4,11)} \tag{54}$$

$$= \prod_{0\neq \mathbf{u}\preccurlyeq 11} \mathrm{RP}_4\left(2\cdot(-1)^{\mathrm{wt}(\mathbf{u})-1}, \mathbf{u}L_Z\right) \tag{55}$$

$$= \mathrm{RP}_4(-2, \mathbf{z}_0 \oplus \mathbf{z}_1)\,\mathrm{RP}_4(2, \mathbf{z}_0)\,\mathrm{RP}_4(2, \mathbf{z}_1) \tag{56}$$

$$= \left(\prod_{i<j\preccurlyeq \mathbf{z}_0\oplus \mathbf{z}_1} CZ_{ij} \prod_{i\preccurlyeq \mathbf{z}_0\oplus \mathbf{z}_1} S_i\right)^{-1} \left(\prod_{i<j\preccurlyeq \mathbf{z}_0} CZ_{ij} \prod_{i\preccurlyeq \mathbf{z}_0} S_i\right) \left(\prod_{i<j\preccurlyeq \mathbf{z}_1} CZ_{ij} \prod_{i\preccurlyeq \mathbf{z}_1} S_i\right). \tag{57}$$

We can choose logical Z operators for the 2D toric code such that $\mathrm{supp}(\mathbf{z}_0) \cap \mathrm{supp}(\mathbf{z}_1) = \emptyset$. In this case, all S operators in equation (57) cancel, as do any CZ operators which lie entirely on the support of either $\mathbf{z}_0$ or $\mathbf{z}_1$, and so we have:

$$\overline{CZ_{01}} := \prod_{i\preccurlyeq \mathbf{z}_0, j\preccurlyeq \mathbf{z}_1} CZ_{ij}. \tag{58}$$

This is an instance of claim 2 in [33] for logical multi-controlled-Z operators. Our method applies to arbitrary diagonal Clifford hierarchy logical operators and we can also show:

$$\overline{CS_{01}} := \overline{CP_8(4,11)} \tag{59}$$

$$= \prod_{0\neq \mathbf{u}\preccurlyeq 11} \mathrm{RP}_8\left(2\cdot(-1)^{\mathrm{wt}(\mathbf{u})-1}, \mathbf{u}L_Z\right) \tag{60}$$

$$= \prod_{i\preccurlyeq \mathbf{z}_0, j\preccurlyeq \mathbf{z}_1} CS_{ij} \prod_{i\preccurlyeq \mathbf{z}_0, j<k\preccurlyeq \mathbf{z}_1} CCZ_{ijk} \prod_{i<j\preccurlyeq \mathbf{z}_0, k\preccurlyeq \mathbf{z}_1} CCZ_{ijk} \tag{61}$$

Note that the number of physical gates used in the implementation is $\mathcal{O}(d^t)$. As we are not guaranteed that $\mathrm{supp}(\mathbf{z}_0) \cap \mathrm{supp}(\mathbf{z}_1) = \emptyset$ for arbitrary CSS codes, the above identities are not completely general. In the linked Jupyter notebook, users can calculate identities of this kind for any desired CSS code for any diagonal Clifford hierarchy logical operator.

5.4. Constructing a CSS code with a desired diagonal logical Clifford hierarchy operator

In this section, we apply the canonical logical operator form of section 5.3 to generate a CSS code with a transversal implementation of a desired logical controlled-phase operator using single-qubit phase gates.

Table 2. Parameters of CSS codes generated by the embedded code method when searching for implementations of logical operators based on the toric code of distance d. For a logical operator acting on k qubits, we use a k-dimensional toric code.

	Logical S			Logical CZ			Logical T			Logical CS			Logical CCZ		
d	n	d_X	d_Z	n	d_X	d_Z	n	d_X	d_Z	n	d_X	d_Z	n	d_X	d_Z
2	1	1	1	4	2	2	1	1	1	12	6	2	8	4	2
3	6	3	2	15	4	3	1	1	1	33	14	2	63	16	3
4	6	3	2	16	4	4	14	7	2	64	22	2	64	16	4
5	15	5	3	35	6	5	15	7	3	155	40	3	215	36	5
6	15	5	3	36	6	6	35	15	2	228	52	4	216	36	6
7	28	7	4	63	8	7	36	15	3	385	76	4	511	64	7
8	28	7	4	64	8	8	92	29	3	512	92	5	512	64	8
9	45	9	5	99	10	9	93	29	3	819	126	5	999	100	9
10	45	9	5	100	10	10	165	45	4	1020	146	6	1000	100	10

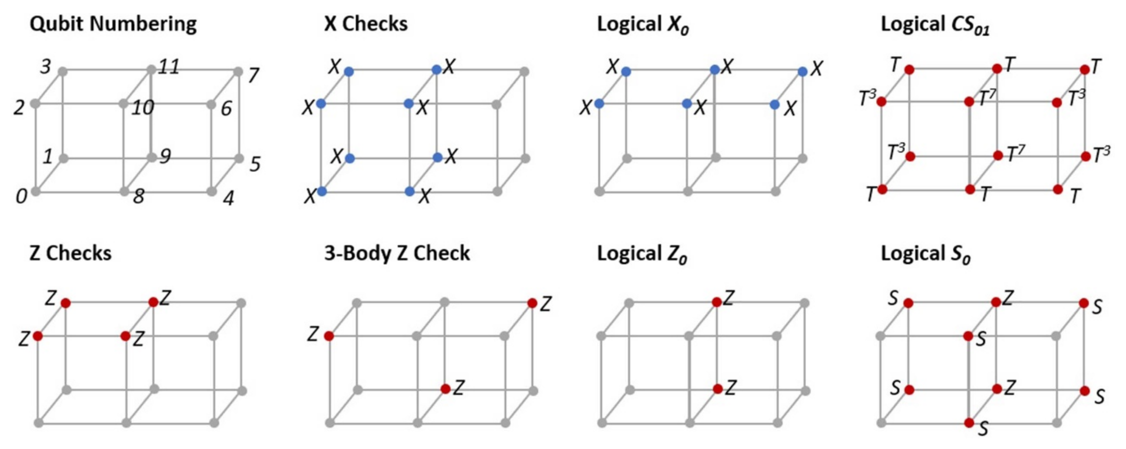


Figure 6. CSS code with transversal logical controlled-S operator: the [[12, 2, 2]] code of example 5.3 is formed from two [[8, 3, 2]] hypercube codes of example 3.1 joined at a common face, with additional three-body Z-checks. Similarly, the [[14, 1, 2]] code with a transversal logical T operator is formed from three [[8, 3, 2]] codes joined pairwise at faces and sharing a common edge.

Algorithm 8. Constructing CSS codes with a desired diagonal logical Clifford hierarchy operator.

Input: a controlled-phase operator B on k qubits (section 2.1) and a target distance d.
Output: a CSS code with a logical $\overline{B}$ operator (section 2.4) composed of single-qubit phase gates.

Method:
1. Let $\mathbf{C}$ be a k-dimensional toric code of distance d. We construct the stabiliser generators of $\mathbf{C}$ using the total complex of the tensor product of k classical repetition codes on d bits (see section II.D of [34]). The resulting CSS code has k non-overlapping logical Z operators of weight d;
2. Find the canonical implementation of $\overline{B} = \prod_{\mathbf{v}\in V} \mathrm{RP}_N(q_{\mathbf{v}}, \mathbf{v})$ composed of phase-rotation gates of maximum support size t using algorithm 7;
3. Remove any elements of V where $q_{\mathbf{v}} = 0$ and apply the embedding $\mathcal{E}_V$ to find the X-checks and X-logicals of the embedded code $\mathbf{C}_V$ as in section 4.3;
4. The resulting code has a logical B operator $\overline{B}_V$ composed of level-t phase gates acting on the embedded codespace.

Example 5.3 (constructing CSS codes with transversal logical controlled phase operators). In table 2, we list the parameters of CSS codes with transversal implementations of various target logical controlled phase operators using the method in section 5.4. The CSS codes are generated from toric codes as follows. For a target operator acting on k logical qubits, we use a k-dimensional toric code. We generate a series of codes by increasing the distance d of the toric code. Looking at the CZ column we have a family of $[[4m^2, 2, 2m]]$ codes with a transversal CZ operator, the first member of which is the [[4, 2, 2]] code of example 2.2. Looking at the CCZ column, we have a family of $[[8m^3, 3, 2m]]$ codes which have a transversal CCZ operator, the first member of which is the hypercube code of example 3.1. The six-qubit code in the S column is the six-qubit code discussed in example 4.1. The 15-qubit code in the T column is the 15-qubit Reed–Muller code. The first entry in the CS column is the [[12, 2, 2]] code with the following X-checks and X-logicals which is illustrated in figure 6:

$$S_X := \begin{pmatrix} 111100001111 \\ 000011111111 \end{pmatrix}; \qquad L_X := \begin{pmatrix} 010101010101 \\ 001100110011 \end{pmatrix}. \tag{62}$$

An interactive version is available in the linked Jupyter notebook.

6. Conclusion and open questions

We have presented efficient new methods to identify and test diagonal logical operators on CSS codes using both single- and multi-qubit diagonal Clifford hierarchy gates as building blocks. In addition, we provided a technique for generating CSS codes with implementations of any desired diagonal Clifford hierarchy logical operator using single-qubit phase gates. The methods generalise to non-CSS stabiliser codes as demonstrated in appendix C. The algorithms are available in a GitHub repository and are intended to be of benefit to researchers in understanding the logical operator structure of stabiliser codes.

Our methods rely on representing diagonal Clifford hierarchy operators as diagonal XP operators. Our algorithms use the vector representation of XP operators and linear algebra modulo N, and so have reduced computational complexity compared to existing work in this area.

The ability to represent diagonal Clifford hierarchy operators as XP operators may have a number of other possible applications. Custom design of CSS codes for devices that have known coherent noise models is one possibility. If the noise can be represented as a series of multi-qubit diagonal operators, we could design a CSS code where these operators are in the logical identity group and so mitigate coherent noise. The simulation of quantum circuits could be another application. A circuit composed of multi-qubit diagonal operators, such as those used for measuring the stabiliser generators of a CSS code, could be amenable to simulation using XP representations of the gates used. As any diagonal Clifford hierarchy operator can be represented as a diagonal XP operator, there could also be implications for computational complexity theory.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

This research was supported by the Australian Research Council via the Centre of Excellence in Engineered Quantum Systems Project Number CE170100009. M W was supported by the Sydney Quantum Academy, Sydney, NSW, Australia. M W and A O Q were students at the 2022 IBM Quantum Error Correction Summer School and the idea for this paper originated there; we wish to express our thanks to IBM for the opportunity to attend the Summer School. M W thanks Robert Calderbank for hosting his visit to Duke University in August 2022—the basic structure of this paper was developed during the visit.

Appendix A. Controlled-phase and phase-rotation operators

In this appendix, we give the detailed proofs of results relating to controlled-phase and phase-rotation operators. The action and duality property of these operators derive from sum/product duality properties for binary vectors and binary variables, and we start by proving these results. We then prove results relating to phase-rotation operators. We first show that phase-rotation operators can be written as a sum of projectors. This allows us to calculate the logical action of phase-rotation operators. We then prove the duality result between controlled-phase and phase-rotation operators. Finally we prove the key commutation relations for phase-rotation and controlled-phase operators.

A.1. Product/sum duality results for binary vectors and variables

Proposition A.1 (sum/product duality of binary vectors). *Let L be a binary matrix with rows $\mathbf{x}_i$ for $0 \leqslant i < r$ and $\mathbf{v}$ a binary vector of length r. Define:*

$$\mathbf{s}_{\mathbf{v}}(L) := \bigoplus_{i \preccurlyeq \mathbf{v}} \mathbf{x}_i; \tag{63}$$

$$\mathbf{p}_{\mathbf{v}}(L) := \prod_{i \preccurlyeq \mathbf{v}} \mathbf{x}_i. \tag{64}$$

Then over the integers:

$$\mathbf{s}_{\mathbf{v}}(L) = \sum_{0 \neq \mathbf{u} \preccurlyeq \mathbf{v}} (-2)^{\mathrm{wt}(\mathbf{u})-1} \mathbf{p}_{\mathbf{u}}(L); \tag{65}$$

$$2^{\mathrm{wt}(\mathbf{v})-1} \mathbf{p}_{\mathbf{v}}(L) = \sum_{0 \neq \mathbf{u} \preccurlyeq \mathbf{v}} (-1)^{\mathrm{wt}(\mathbf{u})-1} \mathbf{s}_{\mathbf{u}}(L). \tag{66}$$

Proof. Restatement of proposition E.10 of [14]. □

Proposition A.2 (sum/product duality of binary variables). *Let* $\mathbf{e}$ *be a vector of r binary variables and let* $\mathbf{v}$ *a binary vector of length r. Define:*

$$s_{\mathbf{v}}(\mathbf{e}) := \bigoplus_{i \preccurlyeq \mathbf{v}} \mathbf{e}[i]; \tag{67}$$

$$p_{\mathbf{v}}(\mathbf{e}) := \prod_{i \preccurlyeq \mathbf{v}} \mathbf{e}[i]. \tag{68}$$

Then over the integers:

$$s_{\mathbf{v}}(\mathbf{e}) = \sum_{0 \neq \mathbf{u} \preccurlyeq \mathbf{v}} (-2)^{\mathrm{wt}(\mathbf{u})-1} p_{\mathbf{u}}(\mathbf{e}); \tag{69}$$

$$2^{\mathrm{wt}(\mathbf{v})-1} p_{\mathbf{v}}(\mathbf{e}) = \sum_{0 \neq \mathbf{u} \preccurlyeq \mathbf{v}} (-1)^{\mathrm{wt}(\mathbf{u})-1} s_{\mathbf{u}}(\mathbf{e}). \tag{70}$$

Proof. Application of proposition A.1 with L the single-column matrix $\mathbf{e}^{\mathrm{T}}$. □

A.2. Phase-rotation operators

Proposition A.3 (projector form of RP operators). *Phase-rotation operators can be written in terms of projectors* $A_{\pm 1} := (I \pm A)/2$*:*

$$\mathrm{RP}_N(q, \mathbf{v}) := \exp\left(\frac{q\pi i}{N} A_{-1}\right) = A_{+1} + \omega^q A_{-1}. \tag{71}$$

Proof. Because A_{-1} is a projector, $A_{-1}^m = A_{-1}$ for integers $m > 0$. Also $A_{-1}^0 = I = A_{+1} + A_{-1}$. Hence:

$$\exp((q\pi i/N) A_{-1}) = I + A_{-1} \sum_{m>0} (q\pi i/N)^m / m! \tag{72}$$

$$= A_{+1} + A_{-1} \sum_{m \geqslant 0} (q\pi i/N)^m / m! \tag{73}$$

$$= A_{+1} + \mathrm{e}^{q\pi i/N} A_{-1}. \tag{74}$$

□

Proposition A.4 (action of RP operators). *The action of a phase-rotation operator on a computational basis element* $|\mathbf{e}\rangle$ *where* $\mathbf{e} \in \mathbb{Z}_2^n$ *and* $\omega := \exp(\pi i/N)$ *is:*

$$\mathrm{RP}_N(q, \mathbf{v}) |\mathbf{e}\rangle = \begin{cases} \omega^q |\mathbf{e}\rangle & \text{if } \mathbf{e} \cdot \mathbf{v} \mod 2 = 1 \\ |\mathbf{e}\rangle & \text{otherwise.} \end{cases} \tag{75}$$

Proof. Straightforward application of projector form of phase-rotation operators in proposition A.3. □

A.3. Duality of controlled-phase and phase-rotation operators

The proposition below allows us to express controlled-phase operators (section 2.1) as products of phase-rotation operators (section 4.1) and vice-versa.

Proposition A.5 (duality of controlled-phase and phase-rotation operators). *For* $N = 2^t$ *and* $\mathbf{u}, \mathbf{v}$ *binary vectors of length n the following identities hold:*

$$\mathrm{RP}_N(2, \mathbf{v}) = \prod_{0 \neq \mathbf{u} \preccurlyeq \mathbf{v}} \mathrm{CP}_N\left(2 \cdot (-2)^{\mathrm{wt}(\mathbf{u})-1}, \mathbf{u}\right) \tag{76}$$

$$\mathrm{CP}_N\left(2^{\mathrm{wt}(\mathbf{v})}, \mathbf{v}\right) = \prod_{0 \neq \mathbf{u} \preccurlyeq \mathbf{v}} \mathrm{RP}_N\left(2 \cdot (-1)^{\mathrm{wt}(\mathbf{u})-1}, \mathbf{u}\right). \tag{77}$$

Proof. Using equation (26) and the notation of proposition A.2, we can write $\mathrm{RP}_N(2,\mathbf{v})|\mathbf{e}\rangle = \omega^{2s_\mathbf{v}(\mathbf{e})}|\mathbf{e}\rangle$. From proposition A.2, we have $2s_\mathbf{v}(\mathbf{e}) = \sum_{0\neq\mathbf{u}\preccurlyeq\mathbf{v}} 2\cdot(-2)^{\mathrm{wt}(\mathbf{u})-1}p_\mathbf{u}(\mathbf{e})$.

Similarly, from equation (2), we can write $\mathrm{CP}_N(2^{\mathrm{wt}(\mathbf{v})},\mathbf{v})|\mathbf{e}\rangle = \omega^{2^{\mathrm{wt}(\mathbf{v})}p_\mathbf{v}(\mathbf{e})}|\mathbf{e}\rangle$ and due to proposition A.2, we have $2^{\mathrm{wt}(\mathbf{v})}p_\mathbf{v}(\mathbf{e}) = \sum_{0\neq\mathbf{u}\preccurlyeq\mathbf{v}} 2\cdot(-1)^{\mathrm{wt}(\mathbf{u})-1}s_\mathbf{u}(\mathbf{e})$.

Hence the phases applied on the RHS and LHS are the same and the result follows. □

A.4. Commutator relations for controlled-phase and phase-rotation operators

In this section, we prove the commutation relations for Pauli X operators with controlled-phase operators (section 2.1) and phase-rotation operators (section 4.1).

Proposition A.6 (commutator relation for phase-rotation operators). *Let X_i denote a Pauli X operator on qubit i. The following identity applies for phase-rotation operators:*

$$\mathrm{RP}_N(q,\mathbf{v})\,X_i = \begin{cases} \omega^q X_i \mathrm{RP}_N(-q,\mathbf{v}) & \textit{if } \mathbf{v}[i]=1 \\ X_i\,\mathrm{RP}_N(q,\mathbf{v}) & \textit{otherwise.}\end{cases} \tag{78}$$

Proof. If $\mathbf{v}[i]=0$, the support of the operators do not overlap and hence the operators commute and the second case follows.

For the case where $\mathbf{v}[i]=1$, let $\mathbf{b}_i^n$ be the length n vector which is zero apart from component i which is one. Then, for a computational basis vector $|\mathbf{e}\rangle$, we have:

$$\mathrm{RP}_N(q,\mathbf{v})\,X_i|\mathbf{e}\rangle = \mathrm{RP}_N(q,\mathbf{v})\,|\mathbf{e}\oplus\mathbf{b}_i^n\rangle \tag{79}$$

$$= \begin{cases} |\mathbf{e}\oplus\mathbf{b}_i^n\rangle & \text{if } (\mathbf{e}\oplus\mathbf{b}_i^n)\cdot\mathbf{v}=0 \mod 2 \\ \omega^q|\mathbf{e}\oplus\mathbf{b}_i^n\rangle & \text{otherwise.}\end{cases} \tag{80}$$

$$\omega^q X_i \mathrm{RP}_N(-q,\mathbf{v})\,|\mathbf{e}\rangle = \begin{cases} \omega^q|\mathbf{e}\oplus\mathbf{b}_i^n\rangle & \text{if } \mathbf{e}\cdot\mathbf{v}=0 \mod 2 \\ \omega^q\omega^{-q}|\mathbf{e}\oplus\mathbf{b}_i^n\rangle & \text{otherwise.}\end{cases} \tag{81}$$

Since, by assumption $\mathbf{v}[i]=1$, $\mathbf{e}\cdot\mathbf{v}=0 \mod 2 \iff (\mathbf{e}\oplus\mathbf{b}_i^n)\cdot\mathbf{v}=1 \mod 2$. Hence, the action on computational basis vectors is identical and the result follows. □

Proposition A.7 (commutation relation for controlled-phase operators).

$$\mathrm{CP}_N(q,\mathbf{v})\,X_i = \begin{cases} X_i\mathrm{CP}_N(q,\mathbf{v}) & \textit{if } \mathbf{v}[i]=0 \\ X_i\mathrm{CP}_N(-q,\mathbf{v})\,\mathrm{CP}_N(q,\mathbf{v}\oplus\mathbf{b}_i^n) & \textit{otherwise}\end{cases} \tag{82}$$

where $\mathbf{b}_i^n$ is the length n binary vector which is zero apart from component i which is one.

Proof. If $\mathbf{v}[i]=0$ then $\mathrm{CP}_N(q,\mathbf{v})$ has no support in common with X_i so the operators commute. Now assume $\mathbf{v}[i]=1$ then the operator on the LHS acts on the computational basis element $|\mathbf{e}\rangle$ as follows:

$$\mathrm{CP}_N(q,\mathbf{v})\,X_i|\mathbf{e}\rangle = \mathrm{CP}_N(q,\mathbf{v})\,|\mathbf{e}\oplus\mathbf{b}_i^n\rangle \tag{83}$$

$$= \begin{cases} \omega^q|\mathbf{e}\oplus\mathbf{b}_i^n\rangle & \text{if } \mathbf{v}\preccurlyeq(\mathbf{e}\oplus\mathbf{b}_i^n) \\ |\mathbf{e}\oplus\mathbf{b}_i^n\rangle & \text{otherwise.}\end{cases} \tag{84}$$

Since by assumption $\mathbf{v}[i]=1$, a phase of ω^q is applied $\iff \mathbf{v}\preccurlyeq(\mathbf{e}\oplus\mathbf{b}_i^n) \iff \mathbf{e}[i]=0$ AND $(\mathbf{v}\oplus\mathbf{b}_i^n)\preccurlyeq\mathbf{e}$. Now consider the RHS and assume $\mathbf{e}[i]=0$ AND $(\mathbf{v}\oplus\mathbf{b}_i^n)\preccurlyeq\mathbf{e}$. In this case, we do not have $\mathbf{v}\preccurlyeq\mathbf{e}$ because $\mathbf{v}[i]=1$ but $\mathbf{e}[i]=0$. Hence:

$$X_i\mathrm{CP}_N(-q,\mathbf{v})\,\mathrm{CP}_N(q,\mathbf{v}\oplus\mathbf{b}_i^n)\,|\mathbf{e}\rangle = \omega^q X_i\mathrm{CP}_N(-q,\mathbf{v})\,|\mathbf{e}\rangle \tag{85}$$

$$= \omega^q X_i|\mathbf{e}\rangle = \omega^q|\mathbf{e}\oplus\mathbf{b}_i^n\rangle. \tag{86}$$

We now show that all other cases result in a trivial phase. Assume $\mathbf{e}[i]=1$AND $(\mathbf{v}\oplus\mathbf{b}_i^n)\preccurlyeq\mathbf{e}$. In this case, $\mathbf{v}\preccurlyeq\mathbf{e}$ and so:

$$X_i\mathrm{CP}_N(-q,\mathbf{v})\,\mathrm{CP}_N(q,\mathbf{v}\oplus\mathbf{b}_i^n)\,|\mathbf{e}\rangle = \omega^q X_i\mathrm{CP}_N(-q,\mathbf{v})\,|\mathbf{e}\rangle \tag{87}$$

$$= \omega^q\omega^{-q}X_i|\mathbf{e}\rangle = |\mathbf{e}\oplus\mathbf{b}_i^n\rangle. \tag{88}$$

Now assume that $(\mathbf{v}\oplus\mathbf{b}_i^n)\preccurlyeq\mathbf{e}$ is not true. In this case, we can never have $\mathbf{v}\preccurlyeq\mathbf{e}$ and so neither of the controlled-phase operators apply a phase, regardless of the value of $\mathbf{e}[i]$. Hence the LHS and RHS have the same action on computational basis elements and the result follows. □

Example A.1 (commutation relation for controlled-phase operators). Using proposition A.7, we can conjugate controlled-phase operators by strings of X operators and vice versa. We first compute $\mathrm{CS}_{01}X_1\mathrm{CS}_{01}^{-1}$ where CS_{01} is a controlled-S operator on qubits 0 and 1. Using the notation of equation (1):

$$\mathrm{CS}_{01}X_1\mathrm{CS}_{01}^{-1} = \mathrm{CP}_8(4,11)X_1\mathrm{CP}_8(-4,11) \tag{89}$$
$$= X_1\mathrm{CP}_8(-4,11)\mathrm{CP}_8(4,10)\mathrm{CP}_8(-4,11) \tag{90}$$
$$= X_1\mathrm{CP}_8(-8,11)\mathrm{CP}_8(4,10) \tag{91}$$
$$= X_1\mathrm{CZ}_{01}S_0 \tag{92}$$

We now compute $(X_0X_1X_2)CCZ_{012}(X_0X_1X_2)^{-1}$. Using equation (31) with $\mathbf{x}=\mathbf{v}=111$, and letting $\mathbf{w}:=\mathbf{u}\oplus\mathbf{v}$:

$$(X_0X_1X_2)\,CCZ_{012}\,(X_0X_1X_2)^{-1} = \mathrm{XP}_2(0|111|0)\,\mathrm{CP}_8(8,111)\,\mathrm{XP}_2(0|111|0) \tag{93}$$
$$= \prod_{\mathbf{u}\preccurlyeq 111}\mathrm{CP}_8\left(8\cdot(-1)^{3+\mathrm{wt}(\mathbf{u})},\mathbf{v}\oplus\mathbf{u}\right) \tag{94}$$
$$= \prod_{0\leqslant \mathrm{wt}(\mathbf{w})\leqslant 3}\mathrm{CP}_8(8,\mathbf{w}) \tag{95}$$
$$= \mathrm{CP}_8(8,0)\prod_{0<\mathrm{wt}(\mathbf{w})\leqslant 3}\mathrm{CP}_8(8,\mathbf{w}) \tag{96}$$
$$= -Z_0Z_1Z_2CZ_{01}CZ_{02}CZ_{12}CCZ_{012} \tag{97}$$

Interactive versions of these examples are available in the linked Jupyter notebook.

Appendix B. Additional details for logical operator algorithms

This appendix provides further details on the various logical operator algorithms. We first prove results that reduce the complexity of the logical action and logical operator test algorithms of sections 3.2, 3.3 and 3.5. We then show how to calculate valid $\mathbf{z}$ vectors that result in diagonal operators that commute with the X-checks up to a logical identity for use in section 3.4. We then demonstrate a method for more efficiently searching for depth-one logical operators composed of multi-qubit controlled phase gates for use in section 4.4. We then show that the embedding operator of section 4.3 acts as a group homomorphism on the group generated by phase-rotation and Pauli X operators. Finally, we show that the canonical form of section 5.3 results in a logical operator with the required action.

B.1. Reducing complexity of logical action algorithms

In this section we show how to reduce the complexity of algorithms which work with the logical action of diagonal XP operators on the canonical codewords. If $B:=\mathrm{XP}_N(0|0|\mathbf{z})$ is a diagonal logical operator of precision $N:=2^t$, then the action of B on the computational basis vectors $\mathbf{e}_{\mathbf{uv}}:=\mathbf{u}S_X+\mathbf{v}L_X$ making up the canonical codewords of equation (7) can be written as $B|\mathbf{e}_{\mathbf{uv}}\rangle=\omega^{2\mathbf{e}_{\mathbf{uv}}\cdot\mathbf{z}}|\mathbf{e}_{\mathbf{uv}}\rangle$. In the proposition below, we show that the phase component $\mathbf{e}_{\mathbf{uv}}\cdot\mathbf{z}$ is completely determined by terms of form $\mathbf{e}_{\mathbf{u}'\mathbf{v}'}\cdot\mathbf{z}$ where $\mathrm{wt}(\mathbf{u}')+\mathrm{wt}(\mathbf{v}')\leqslant t$. As a result, when working with logical actions, we do not need to consider all 2^{k+r} of the $\mathbf{e}_{\mathbf{uv}}$ vectors, just a limited set which is of size polynomial in k and r. This reduces the computational complexity of the algorithms in sections 3.2 and 3.5.

Proposition B.1. *Let $N:=2^t$ and $\mathbf{z}\in\mathbb{Z}_N^n$. The phase component $\mathbf{e}_{\mathbf{uv}}\cdot\mathbf{z}$ can be expressed as a $\mathbb{Z}_N$ linear combination of $\mathbf{e}_{\mathbf{u}'\mathbf{v}'}\cdot\mathbf{z}$ where $\mathrm{wt}(\mathbf{u}')+\mathrm{wt}(\mathbf{v}')\leqslant t$.*

Proof. Let $G_X:=\begin{pmatrix}S_X\\L_X\end{pmatrix}$ and let $\mathbf{a}:=(\mathbf{u}|\mathbf{v})$. Noting that $\mathbf{e}_{\mathbf{uv}}=\mathbf{a}G_X=\mathbf{s}_{\mathbf{a}}(G_X)$ and applying proposition A.1 we have:

$$\mathbf{a}G_X=\mathbf{s}_{\mathbf{a}}(G_X)=\sum_{0\neq\mathbf{b}\preccurlyeq\mathbf{a}}(-2)^{\mathrm{wt}(\mathbf{b})-1}\mathbf{p}_{\mathbf{b}}(G_X). \tag{98}$$

Terms with $\mathrm{wt}(\mathbf{b})>t$ disappear modulo $N=2^t$. Using the linearity of dot product and expressing the $\mathbf{p}_{\mathbf{b}}(G_X)$ in terms of $\mathbf{s}_{\mathbf{c}}(G_X)$:

$$\mathbf{a}G_X \cdot \mathbf{z} \mod N = \left(\sum_{\substack{0 \neq \mathbf{b} \preccurlyeq \mathbf{a} \\ \mathrm{wt}(\mathbf{a}) \leqslant t}} (-2)^{\mathrm{wt}(\mathbf{b})-1} \mathbf{p}_{\mathbf{b}}(G_X) \right) \cdot \mathbf{z} \bmod N \tag{99}$$

$$= \left(\sum_{\substack{0 \neq \mathbf{b} \preccurlyeq \mathbf{a} \\ \mathrm{wt}(\mathbf{a}) \leqslant t}} (-2)^{\mathrm{wt}(\mathbf{b})-1} \sum_{0 \neq \mathbf{c} \preccurlyeq \mathbf{b}} (-1)^{\mathrm{wt}(\mathbf{c})-1} \mathbf{s}_{\mathbf{c}}(G_X) \right) \cdot \mathbf{z} \bmod N \tag{100}$$

$$= \left(\sum_{\substack{0 \neq \mathbf{b} \preccurlyeq \mathbf{a} \\ \mathrm{wt}(\mathbf{a}) \leqslant t}} (-2)^{\mathrm{wt}(\mathbf{b})-1} \sum_{0 \neq \mathbf{c} \preccurlyeq \mathbf{b}} (-1)^{\mathrm{wt}(\mathbf{c})-1} \mathbf{c}G_X \cdot \mathbf{z} \right) \bmod N. \tag{101}$$

As $\mathrm{wt}(\mathbf{c}) \leqslant \mathrm{wt}(\mathbf{b}) \leqslant \mathrm{wt}(\mathbf{a}) \leqslant t$, the result follows. □

B.2. Test for diagonal logical XP operators

In this section, we prove that the algorithm in section 3.3 correctly identifies diagonal logical operators of XP form. We first show that if the group commutator of an operator B with each of the logical identities A_1, A_2 is a logical identity, then the group commutator of the product A_1A_2 is a logical identity.

Proposition B.2 (commutators of logical identities). *Let $\mathcal{I}_{\mathrm{XP}}$ be the logical identity group as defined in section 2.6 and let $A_1, A_2 \in \mathcal{I}_{\mathrm{XP}}$. Let B be an XP operator such that $[[A_1, B]]$ and $[[A_2, B]] \in \mathcal{I}_{\mathrm{XP}}$. Then $[[A_1A_2, B]] \in \mathcal{I}_{\mathrm{XP}}$.*

Proof. As $\mathcal{I}_{\mathrm{XP}}$ is a group, $A_1, A_2 \in \mathcal{I}_{\mathrm{XP}} \implies [[A_1, A_2]] \in \mathcal{I}_{\mathrm{XP}}$. Hence we can write $A_1A_2 = CA_2A_1$ for some $C \in \mathcal{I}_{\mathrm{XP}}$. Calculating $[[A_1A_2, B]]$, for some $C, C', C'' \in \mathcal{I}_{\mathrm{XP}}$:

$$[[A_1A_2, B]] = A_1A_2BA_2^{-1}A_1^{-1}B^{-1} \tag{102}$$

$$= A_1\left(A_2BA_2^{-1}B^{-1}\right)BA_1^{-1}B^{-1} \tag{103}$$

$$= A_1CBA_1^{-1}B^{-1} \tag{104}$$

$$= C'A_1BA_1^{-1}B^{-1} \tag{105}$$

$$= C'C'' \in \mathcal{I}_{\mathrm{XP}}. \tag{106}$$

□

We now show that for a diagonal XP operator B, it is sufficient to check group commutators with the $r := |S_X|$ operators of form $\mathrm{XP}_N(0|\mathbf{x}_i|0)$ where $\mathbf{x}_i$ are the rows of the X-checks S_X.

Proposition B.3. *Let $\mathbf{C}$ be a CSS code with X-checks S_X and logical identity XP group $\mathcal{I}_{\mathrm{XP}}$ of precision N. Let $B := \mathrm{XP}_N(0|0|\mathbf{z})$ be a diagonal XP operator. B is a logical operator if and only if $[[\mathrm{XP}_N(0|\mathbf{x}_i|0), B]] \in \mathcal{I}_{\mathrm{XP}}$ for all rows $\mathbf{x}_i$ of S_X.*

Proof. B is a logical operator if and only if $[[A, B]] \in \mathcal{I}_{\mathrm{XP}}$ for all $A \in \mathcal{I}_{\mathrm{XP}}$. If B is a logical operator then $[[\mathrm{XP}_N(0|\mathbf{x}_i|0), B]] \in \mathcal{I}_{\mathrm{XP}}$ because $\mathrm{XP}_N(0|\mathbf{x}_i|0) \in \mathcal{I}_{\mathrm{XP}}$.

Conversely, assume $[[\mathrm{XP}_N(0|\mathbf{x}_i|0), B]] \in \mathcal{I}_{\mathrm{XP}}$ for all rows $\mathbf{x}_i$ of S_X. Let K_{M} be the matrix whose rows are a generating set of the Z-components of the logical identities as defined in section 3.1. Any logical identity A can be written as a product of terms of form $\mathrm{XP}_N(0|\mathbf{x}_i|0)$ and $\mathrm{XP}_N(0|0|\mathbf{z}_j)$ where $\mathbf{z}_j$ is a row of K_{M}. By assumption, $[[\mathrm{XP}_N(0|\mathbf{x}_i|0), B]] \in \mathcal{I}_{\mathrm{XP}}$ and $[[\mathrm{XP}_N(0|0|\mathbf{z}_j), B]] = I$. Due to proposition B.2, the commutator of the product is a logical identity and the result follows. □

B.3. Algorithm to determine commutators of a given X-check

In the method of section 3.4, for a given X-check $\mathbf{x} \in S_X$ we seek all Z-components $\mathbf{z} \in \mathbb{Z}_N^n$ such that the group commutator $[[\mathrm{XP}_N(0|0|\mathbf{z}), \mathrm{XP}_N(0|\mathbf{x}|0)]]$ is a logical identity. This reduces to solving for $\mathbf{z}$ such that both $\mathbf{x} \cdot \mathbf{z} = 0 \mod N$ and $2\mathbf{x}\mathbf{z} \in \langle K_{\mathrm{M}} \rangle_{\mathbb{Z}_N}$ where the rows of K_{M} are the Z-components of the diagonal logical identities as in section 3.1. In this section, we show how to solve for these constraints using linear algebra modulo N. The method is as follows:

Without loss of generality, reorder qubits so that the first m components of $\mathbf{x}$ are one and the remaining $n - m$ components are zero. In the matrices of form $(\mathbf{a}|\mathbf{b})$ below, the first component has m columns corresponding to the non-zero components of $\mathbf{x}$, the next $n - m$ columns correspond to the zero components of $\mathbf{x}$. For $\mathbf{v} \in \mathbb{Z}_N$, let $\mathbf{v} \cdot 1 := \left(\sum_{0 \leqslant i < n} \mathbf{v}[i]\right) \mod N$.

1. The vector $2\mathbf{xz}$ is of the form $(2\mathbf{u}|0)$ where $\mathbf{u}$ is of length m and the row span of $C_0 := (2I_m|0)$ over $\mathbb{Z}_N$ represents all vectors of this form.
2. Group commutators which are also logical identities are in $\langle C_0\rangle_{\mathbb{Z}_N} \bigcap \langle K_\mathrm{M}\rangle_{\mathbb{Z}_N}$ and a Howell basis C_1 is calculated via the intersection of spans method in appendix 4.1 of [14].
3. The rows of C_1 are of form $(\mathbf{u}|0) \in \langle K_\mathrm{M}\rangle_{\mathbb{Z}_N}$ for $\mathbf{u}$ divisible by 2 modulo N. Now let $\mathbf{v} := \mathbf{u}/2$. Because $2(\mathbf{v}+N/2) = 2\mathbf{v} = \mathbf{u} \mod N$, $(\mathbf{v}\cdot 1) \mod N$ is either 0 or $N/2$. Adjust the mth component of $\mathbf{v}$ by subtracting $(\mathbf{v}\cdot 1) \mod N$. Let C_2 be the matrix formed from rows of form $(\mathbf{v}|0)$.
4. Adding pairs of $N/2$ to the first m components does not change $2\mathbf{xz}$ or $\mathbf{x}\cdot\mathbf{z} \mod N$. Let A be I_{m-1} with a column of ones appended. Add the rows $(N/2\cdot A|0)$ to C_2.
5. Columns i where $\mathbf{x}[i] = 0$ can have arbitrary values, as these do not contribute to $2\mathbf{xz}$ or $\mathbf{x}\cdot\mathbf{z}$. Add the rows $(0|I_{n-m})$ to C_2.
6. Return qubits to their original order. The valid Z-components are given by the row span of C_2 over $\mathbb{Z}_N$.

B.4. Algorithm for depth-one operators

In the depth-one algorithm, we find transversal logical operators by starting with a level-t logical XP operator acting on the embedded codespace, then multiplying by all possible elements of the diagonal logical XP group of the embedded code. If the order of the diagonal logical XP group is large, this method can be computationally expensive. In this section, we demonstrate an algorithm for more efficiently exploring the search space and checking if an operator acting on the embedded codespace acts transversally on the codespace. We use the residue function defined in equation 142 of [14]—we say that $\mathbf{z}' = \mathrm{Res}_{\mathbb{Z}_N}(K_\mathrm{L},\mathbf{z})$ if:

$$\mathrm{How}_{\mathbb{Z}_N}\begin{pmatrix}1 & \mathbf{z}\\ 0 & K_\mathrm{L}\end{pmatrix} = \begin{pmatrix}1 & \mathbf{z}'\\ 0 & K_\mathrm{L}'\end{pmatrix} \tag{107}$$

The input to this algorithm is the following:

- A binary matrix V for the embedding operator $\mathcal{E}_V$;
- A matrix K_L representing the Z-components of the generators of the diagonal logical XP group of the embedded code (see section 3.4).
- A row vector $\mathbf{z}$ of K_L which represents the Z-component of a non-trivial logical operator at level t of the diagonal Clifford hierarchy acting on the embedded codespace. This corresponds to a product of phase-rotation gates acting on the original codespace.

The output is a depth-one implementation of a non-trivial logical operator at level t of the diagonal Clifford hierarchy, or `FALSE` if no such operator exists. The algorithm method is as follows:

(1) Remove $\mathbf{z}$ from K_L;
(2) Let `todo` be a list containing only the all ones vector of length $|V|$. The vectors $\mathbf{a}$ in `todo` have columns indexed by rows of V and represent partial partitions of the n qubits. The value of $\mathbf{a}[\mathbf{v}]$ encodes the following information:
  - 0: $\mathrm{supp}(\mathbf{v})$ is not a partition;
  - 1: whether $\mathrm{supp}(\mathbf{v})$ is a partition has not yet been determined;
  - 2: $\mathrm{supp}(\mathbf{v})$ is a partition. For depth-one operators, any $\mathbf{u}$ with overlapping support (i.e. $\mathbf{u}\cdot\mathbf{v} \geqslant 0$ is not a partition).

(3) While `todo` is not empty:
  (a) Pop the vector $\mathbf{a}$ from the end of `todo`;
  (b) Reorder the columns of $\mathbf{z}$ and K_L by moving the columns with $\mathbf{a}[\mathbf{v}] = 0$ to the far left, the columns with $\mathbf{a}[\mathbf{v}] = 1$ to the middle and the columns with $\mathbf{a}[\mathbf{v}] = 2$ to the far right.
  (c) Calculate $\mathbf{z}' := \mathrm{Res}_{\mathbb{Z}_N}(K_\mathrm{L},\mathbf{z})$. If $\mathbf{z}'[\mathbf{v}] > 0$ for any $\mathbf{v}$ where $\mathbf{a}[\mathbf{v}] = 0$ then the partition is not valid. This is because taking the residue will eliminate the leftmost entries of $\mathbf{z}$ if possible by adding rows of K_L;
  (d) If the partition is valid, find the first $\mathbf{v}$ such that $\mathbf{z}'[\mathbf{v}] > 0$ and $\mathbf{a}[\mathbf{v}] = 1$;
  (e) If there is no such $\mathbf{v}$, we have a depth-one implementation. Return the qubits to their original order and return $\mathbf{z}'$;
  (f) Otherwise, let $\mathbf{a}_1$ be the same as $\mathbf{a}$ but where $\mathbf{a}[\mathbf{v}] = 2$ and $\mathbf{a}[\mathbf{u}] = 0$ for all $\mathbf{u}$ such that $\mathbf{u}\cdot\mathbf{v} > 0$. Let $\mathbf{a}_2$ be the same as $\mathbf{a}$ but with $\mathbf{a}[\mathbf{v}] = 2$. The vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ represent the two possible scenarios where either $\mathbf{v}$ is or is not a partition—append them to `todo`;

(g) Return to step 3.

(4) Return FALSE as all possible configurations have been explored.

The above algorithm yields depth-one operators composed of physical phase-rotation gates. If implementations using physical controlled-phase gates are required, convert $\mathbf{z}$ and K_L to controlled-phase representations using the method in section 4.2. If we require a logical operator with exactly the same action as the original operator with Z-component $\mathbf{z}$, substitute the Z-components of the diagonal logical identity generators K_M of section 3.1 for K_L.

B.5. Representation of controlled-phase operators as XP operators via embedding operator

In this section we prove that the phase-rotation operators of section 4.1 acting on a codespace correspond to diagonal XP operators in the embedded codespace defined in section 4.3.2. We do this by demonstrating that the mapping of phase-rotation operators acting on the codespace to XP operators in the embedded codespace of section 4.3.2 is a group homomorphism.

Proposition B.4 (embedding operator induces a group homomorphism). *The embedding operator $\mathcal{E}_V$ defined as follows is a group homomorphism between $\mathcal{XRP}_N^V$ and $\mathcal{XP}_N^{|V|}$:*

$$\mathcal{E}_V\left(\mathrm{XRP}_N^V(p|\mathbf{x}|\mathbf{q})\right) := \mathrm{XP}_N\left(p|\mathbf{x}V^\mathrm{T}|\mathbf{q}\right). \tag{108}$$

Proof. We prove this by considering generators of the group $\mathcal{XRP}_N^V$. Let $\mathbf{b}_i^n$ be the length n binary vector which is zero apart from component i which is one and consider X_i, X_j for $0 \leqslant i,j < n$:

$$\begin{aligned}
\mathcal{E}_V\left(X_iX_j\right) &= \mathcal{E}_V\left(\mathrm{XRP}_V\left(0|\mathbf{b}_i^n + \mathbf{b}_j^n|0\right)\right) && (109)\\
&= XP_N\left(0|\left(\mathbf{b}_i^n + \mathbf{b}_j^n\right)V^\mathrm{T}|0\right) && (110)\\
&= XP_N(0|(\mathbf{b}_i^nV^T|0)XP_N(0|(\mathbf{b}_j^nV^\mathrm{T}|0) && (111)\\
&= \mathcal{E}_V\left(X_i\right)\mathcal{E}_V\left(X_j\right). && (112)
\end{aligned}$$

By a similar argument, $\mathcal{E}_V\big(\mathrm{RP}_N(2,\mathbf{u})\mathrm{RP}_N(2,\mathbf{v})\big) = \mathcal{E}_V\big(\mathrm{RP}_N(2,\mathbf{u})\big)\mathcal{E}_V\big(\mathrm{RP}_N(2,\mathbf{v})\big)$ for $\mathbf{u},\mathbf{v} \in V$. Where X operators precede diagonal operators we have:

$$\begin{aligned}
\mathcal{E}_V\left(X_i\mathrm{RP}_N(2,\mathbf{v})\right) &= \mathcal{E}_V\left(\mathrm{XRP}_V\left(0|\mathbf{b}_i^n|\mathbf{b}_\mathbf{v}^{|V|}\right)\right) && (113)\\
&= \mathrm{XP}_N\left(0|\mathbf{b}_i^nV^\mathrm{T}|\mathbf{b}_\mathbf{v}^{|V|}\right) && (114)\\
&= \mathcal{E}_V\left(X_i\right)\mathcal{E}_V\left(\mathrm{RP}_N(2,\mathbf{v})\right) && (115)
\end{aligned}$$

where diagonal operators precede X operators, we first consider the case where $\mathbf{v}[i] = 0$. In this case, the operators commute so we can swap the order of operators so that the X operators precede the diagonal operator. Now consider the case $\mathbf{v}[i] = 1$ where the operators do not commute:

$$\begin{aligned}
\mathcal{E}_V\left(\mathrm{RP}_N(2,\mathbf{v})X_i\right) &= \mathcal{E}_V\left(\omega^2X_i\mathrm{RP}_N(-2,\mathbf{v})\right) && (116)\\
&= \mathrm{XP}_N\left(2|\mathbf{b}_i^nV^\mathrm{T}| - \mathbf{b}_\mathbf{v}^{|V|}\right) && (117)
\end{aligned}$$

Due to the commutation relation of equation (15) and because $(\mathbf{b}_i^nV^\mathrm{T})\mathbf{b}_\mathbf{v}^{|V|} = \mathbf{b}_\mathbf{v}^{|V|}$ when $\mathbf{v}[i] = 1$:

$$\begin{aligned}
\mathcal{E}_V\left(\mathrm{RP}_N(2,\mathbf{v})\right)\mathcal{E}_V\left(X_i\right) &= \mathrm{XP}_N\left(0|0|\mathbf{b}_\mathbf{v}^{|V|}\right)\mathrm{XP}_N\left(0|\mathbf{b}_i^nV^\mathrm{T}|0\right) && (118)\\
&= \mathrm{XP}_N\left(2|\mathbf{b}_i^nV^\mathrm{T}| - \mathbf{b}_\mathbf{v}^{|V|}\right) && (119)\\
&= \mathcal{E}_V\left(\mathrm{RP}_N(2,\mathbf{v})X_i\right). && (120)
\end{aligned}$$

Because group operations are preserved for generators of the group, the embedding is a group homomorphism. □

B.6. Canonical form of logical phase-rotation operators

In this section, we show that the canonical form of logical phase-rotation operators discussed in section 5.1 acts as a logical operator as claimed.

Proposition B.5 (logical phase rotation operator). *Let L_Z be a binary matrix representing the Z-components of logical Z operators such that $L_Z^{\mathrm{T}}L_X = I_k$ and let* $\mathbf{w}$ *be a binary vector of length k.*

The operator $\mathrm{RP}_N(2,\mathbf{w}L_Z)$ *acts as a logical* $\overline{\mathrm{RP}_N(2,\mathbf{w})}$ *operator.*

Proof. This can be seen by considering the action of the operator on the computational basis element $|\mathbf{e_{uv}}\rangle$ where $\mathbf{e_{uv}} := \mathbf{u}S_X + \mathbf{v}L_X$. From the argument in proposition 5.1, $\mathbf{e_{uv}}\cdot\mathbf{z}_j \mod 2 = 1 \iff \mathbf{v}[j] = 1$. Hence:

$$\mathbf{e_{uv}}\cdot\left(\bigoplus_{j\preccurlyeq\mathbf{w}}\mathbf{z}_j\right) \mod 2 = 1 \iff \bigoplus_{j\preccurlyeq\mathbf{w}}\mathbf{v}[j] = 1 \tag{121}$$

$$\iff \mathbf{v}\cdot\mathbf{w} \mod 2 = 1. \tag{122}$$

Hence, the phases applied by the operators are the same and the result follows. □

Appendix C. Application of methods to non-CSS stabiliser codes

In this work, we have focused on identifying diagonal logical operators for CSS codes in the form defined in section 2.3. In this section, we show how to find diagonal logical operators for arbitrary non-CSS stabiliser codes. We will prove the following main proposition:

Proposition C.1 (mapping non-CSS stabiliser codes to CSS codes). *Let* $\mathbf{C}$ *be the codespace of a Pauli stabiliser code on n qubits. There exists a CSS code on n qubits with codespace* $\mathbf{C}'$ *such that* $\mathbf{C} = DQ\mathbf{C}'$ *where* $Q := \mathrm{XP}_2(0|\mathbf{q}|0)$, $\boldsymbol{q}$ *is a length n binary vector and D is a diagonal level 2 Clifford operator. Furthermore, a diagonal operator* $\overline{B}$ *is a logical B operator of* $\mathbf{C}'$ *if and only if* $Q\overline{B}Q^{-1}$ *is a logical B operator of* $\mathbf{C}$.

The CSS code $\mathbf{C}'$ in proposition C.1 may have different error correction properties to $\mathbf{C}$ (i.e. weight of stabiliser generators and logical operators), but allows us to determine the diagonal logical operator structure of $\mathbf{C}$. In this section, we first introduce some background material on non-CSS stabiliser codes. CSS codes of the form of section 2.3 have diagonal stabiliser generators with zero phase components and non-diagonal stabiliser generators with zero phase and Z-components. This is not the case for arbitrary stabiliser codes, and we show how to eliminate these components in two steps to yield the operators Q and D in the above proposition. We illustrate proposition C.1 by applying it to the perfect five-qubit code of [35].

C.1. Background on non-CSS codes

Arbitrary Pauli stabiliser codes have stabiliser generators from the Pauli group $\langle iI, X, Z\rangle^{\otimes n} = \mathcal{XP}_2^n$. A method of determining a canonical set of independent stabiliser generators, logical X and logical Z operators is given on page 477 of [36]. Let $\mathbf{S}_X$ and $\mathbf{S}_Z$ be the canonical stabiliser generators and let $\mathbf{L}_X$ be the canonical logical X operators. Elements of $\mathbf{S}_Z$ may have signs of ± 1 and elements of $\mathbf{S}_X$ may have non-trivial phase and Z-components. For proposition C.1, we require that $\mathbf{C}' := (DQ)^{-1}\mathbf{C}$ is stabilised by diagonal generators with trivial phase components and non-diagonal generators with trivial phase and Z-components.

We now set out a canonical form for the codewords of the stabiliser code $\mathbf{C}$. Let r be the number of operators in $\mathbf{S}_X$ and k the number of operators in $\mathbf{L}_X$ and let $\mathbf{v}\in\mathbb{Z}_2^k$. Let $\mathbf{q}$ be a binary vector of length n such that $B|\mathbf{q}\rangle = |\mathbf{q}\rangle$ for all $B\in\mathbf{S}_Z$. Define $\mathbf{L}_X^{\mathbf{v}} := \prod_{i\preccurlyeq\mathbf{v}}\mathbf{L}_X[i]$ where $\mathbf{L}_X[i]$ is the ith operator in $\mathbf{L}_X$. Due to the arguments in sections 4.2 and 6.2 of [14], the following codewords span the codespace $\mathbf{C}$ and define the encoding map $\mathcal{C}$ of $\mathbf{C}$ (section 2.4):

$$|\mathbf{v}\rangle_L = \sum_{\mathbf{u}\in\mathbb{Z}_2^r}\mathbf{S}_X^{\mathbf{u}}\mathbf{L}_X^{\mathbf{v}}|\mathbf{q}\rangle. \tag{123}$$

We now discuss how the codewords and logical operators of a stabiliser code $\mathbf{C}$ transform when the codespace is acted upon by a unitary operator U. The codewords of the transformed code $\mathbf{C}' := U\mathbf{C}$ are given by $U|\mathbf{v}\rangle_{\mathrm{L}}$ so the encoding map of $\mathbf{C}'$ is given by $U\mathcal{C}$. The operator A is a logical identity of $\mathbf{C}$ if and only if UAU^{-1} is a logical identity of $\mathbf{C}'$. This is because:

$$\left(UAU^{-1}\right)U|\mathbf{v}\rangle_{\mathrm{L}} = UA|\mathbf{v}\rangle_{\mathrm{L}} = U|\mathbf{v}\rangle_{\mathrm{L}}. \tag{124}$$

As the stabiliser generators $\mathbf{S}_X$ and $\mathbf{S}_Z$ are elements of the logical identity group, they also update via conjugation. The operator $\overline{B}$ is a logical B operator on $\mathbf{C}$ if and only if $U\overline{B}U^{-1}$ is a logical B operator on $\mathbf{C}'$ because for all logical identities A of $\mathbf{C}$ the requirements of sections 3.3 and 2.4 are met as follows:

$$\left[\left[U\overline{B}U^{-1}, UAU^{-1}\right]\right] = U\left[\left[\overline{B}, A\right]\right]U^{-1}; \text{and} \tag{125}$$

$$\left(U\overline{B}U^{-1}\right)UC = U\overline{B}C = (UC)B. \tag{126}$$

C.2. Eliminating phase components from diagonal stabiliser generators

We now show how to find the vector $\mathbf{q}$ in equation (123) which allow us to eliminate signs from the diagonal stabiliser generators of the non-CSS code $\mathbf{C}$ via conjugation by the operator $Q := \mathrm{XP}_2(0|\mathbf{q}|0)$.

The canonical diagonal stabiliser generators $\mathbf{S}_Z$ are of form $XP_2(2p_i|0|\mathbf{z}_i)$ where $p_i \in \mathbb{Z}_2$ and $\mathbf{z}_i \in \mathbb{Z}_2^n$. Let E_s be the binary matrix with rows of form $(p_i|\mathbf{z}_i)$ and let $K_s := \ker_{\mathbb{Z}_2}(E_s)$. If $p_i = 1$ for any i, the top row of K_s is of form $(1|\mathbf{q})$ and satisfies $p_i + \mathbf{q}\cdot\mathbf{z}_i = 0 \mod 2$ for all i. Now let $Q := \mathrm{XP}_2(0|\mathbf{q}|0)$ then we also have $QXP_2(2p_i|0|\mathbf{z}_i)Q = XP_2(2p_i + 2\mathbf{q}\cdot\mathbf{z}_i|0|\mathbf{z}_i) = XP_2(0|0|\mathbf{z}_i)$. Hence conjugation by Q eliminates the phase components of the diagonal stabiliser generators as required. As Q is non-diagonal, the diagonal logical operators and identities may update on conjugation by Q.

C.3. Eliminating phase and Z-components from non-diagonal stabiliser generators

We now show how to find a diagonal level 2 Clifford operator D from proposition C.1 which allows us to eliminate the phase and Z-components of the non-diagonal stabilisers $\mathbf{S}_X$. Let $|S\rangle$ be the state stabilised by the set of n independent operators $\mathbf{S}_X, \mathbf{S}_Z$ and $\mathbf{L}_X$. We can write $|S\rangle$ as follows:

$$|S\rangle = \sum_{\mathbf{u}\in\mathbb{Z}_2^r, \mathbf{v}\in\mathbb{Z}_2^k} \mathbf{S}_X^{\mathbf{u}}\mathbf{L}_X^{\mathbf{v}}|\mathbf{q}\rangle = \sum_{\mathbf{v}} |\mathbf{v}\rangle_{\mathrm{L}}. \tag{127}$$

Let S_X and L_X be the binary matrices formed from the X-components of $\mathbf{S}_X$ and $\mathbf{L}_X$ respectively. Using the terminology of proposition 5.1 of [14], $|S\rangle$ is an XP state of precision $N = 2$ and so is a weighted hypergraph state of form:

$$|S\rangle = D\sum_{\mathbf{u},\mathbf{v}} |\mathbf{u}S_X + \mathbf{v}L_X + \mathbf{q}\rangle = DQ\sum_{\mathbf{u},\mathbf{v}} |\mathbf{u}S_X + \mathbf{v}L_X\rangle. \tag{128}$$

The operator D is a product of diagonal level 2 Clifford operators and can calculated via the method in algorithm 5.3.1 of [14]. Now let $\mathbf{C}'$ be the CSS code specified by the X-checks S_X and X-logicals L_X. Due to section 2.3, codewords of $\mathbf{C}'$ are of form $|\mathbf{v}\rangle_{\mathrm{L}}' := \sum_{\mathbf{u}} |\mathbf{u}S_X + \mathbf{v}L_X\rangle$ and so the codewords of $\mathbf{C}$ can be written:

$$|\mathbf{v}\rangle_{\mathrm{L}} = DQ\sum_{\mathbf{u}} |\mathbf{u}S_X + \mathbf{v}L_X\rangle = DQ|\mathbf{v}\rangle_{\mathrm{L}}'. \tag{129}$$

Hence, $\mathbf{C} = DQ\mathbf{C}'$ as required. Transforming a CSS code $\mathbf{C}'$ by the diagonal operator D has no effect on the diagonal stabiliser generators, logical identities or logical operators. However, it can increase the weight of non-diagonal stabiliser generators and logical X operators, and so increase the code distance.

Example C.1 (perfect five-qubit code). Let $\mathbf{C}$ be the perfect five-qubit code of [35] with stabiliser generators and logical X operator as follows:

$$\mathbf{S} := \begin{pmatrix} XZZXI \\ IXZZX \\ XIXZZ \\ ZXIXZ \end{pmatrix}; \qquad \overline{X} := ZIIZX. \tag{130}$$

Let $\mathbf{C}'$ be the CSS code with X-checks and X-logicals:

$$S_X := \begin{pmatrix} 10010 \\ 01001 \\ 10100 \\ 01010 \end{pmatrix}; \qquad L_X := \left(00001\right). \tag{131}$$

We find that $D = CZ_{01}CZ_{12}CZ_{23}CZ_{01}CZ_{34}CZ_{40}$ satisfies $\mathbf{C} = D\mathbf{C}'$ using the conjugation rule $CZ_{01}X_0CZ_{01} = X_0Z_1$. Whilst $\mathbf{C}$ has distance 3, $\mathbf{C}'$ has distance 1. In the linked Jupyter notebook, users can use the above method to find D, Q and $\mathbf{C}'$ for various non-CSS stabiliser codes from www.codetables.de.

ORCID iDs

Mark A Webster https://orcid.org/0000-0002-5300-1643
Armanda O Quintavalle https://orcid.org/0000-0001-5101-5673
Stephen D Bartlett https://orcid.org/0000-0003-4387-670X

References

[1] Acharya R *et al* 2023 Suppressing quantum errors by scaling a surface code logical qubit *Nature* **614** 676–81
[2] Ryan-Anderson C *et al* 2021 Realization of real-time fault-tolerant quantum error correction *Phys. Rev.* X **11** 041058
[3] Krinner S *et al* 2022 Realization of real-time fault-tolerant quantum error correction *Nature* **605** 669–74
[4] Calderbank R and Shor P W 1996 Good quantum error-correcting codes exist *Phys. Rev.* A **54** 1098–105
[5] Eastin B and Knill E 2009 Restrictions on transversal encoded quantum gate sets *Phys. Rev. Lett.* **102** 110502
[6] Gottesman D and Chuang I L 1999 Demonstrating the viability of universal quantum computation using teleportation and single-qubit operations *Nature* **402** 390–3
[7] Nebe G, Rains E M and Sloane N J A 2001 The invariants of the clifford groups *Des. Codes Cryptogr.* **24** 99–122
[8] Steane A M 1996 Error correcting codes in quantum theory *Phys. Rev. Lett.* **77** 793–7
[9] Bombin H and Martin-Delgado M A 2006 Topological quantum distillation *Phys. Rev. Lett.* **97** 180501
[10] Bravyi S and Haah J 2012 Magic-state distillation with low overhead *Phys. Rev.* A **86** 052329
[11] Moussa J E 2016 Transversal clifford gates on folded surface codes *Phys. Rev.* A **94** 042316
[12] Breuckmann N P and Burton S 2022 Fold-transversal clifford gates for quantum codes (arXiv:2202.06647 [quant-ph])
[13] Quintavalle A O, Webster P and Vasmer M 2022 Partitioning qubits in hypergraph product codes to implement logical gates (arXiv:2204.10812 [quant-ph])
[14] Webster M A, Brown B J and Bartlett S D 2022 The XP stabiliser formalism: a generalisation of the pauli stabiliser formalism with arbitrary phases *Quantum* **6** 815
[15] Calderbank R, Rains E M, Shor P M and Sloane N J A 1998 Quantum error correction via codes over GF(4) *IEEE Trans. Inf. Theory* **44** 1369–87
[16] Babai L'o, Codenotti P, Grochow J A and Qiao Y 2011 Code equivalence and group isomorphism *Proc. 2011 Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)* pp 1395–408
[17] Rengaswamy N, Calderbank R and Pfister H D 2019 Unifying the clifford hierarchy via symmetric matrices over rings *Phys. Rev.* A **100** 022304
[18] Rengaswamy N, Calderbank R, Kadhe S and Pfister H D 2020 Logical clifford synthesis for stabilizer codes *IEEE Trans. Quantum Eng.* **1** 1–17
[19] Hu J, Liang Q and Calderbank R 2021 Climbing the diagonal clifford hierarchy (arXiv:2110.11923 [quant-ph])
[20] Cui S X, Gottesman D and Krishna A 2017 Diagonal gates in the clifford hierarchy *Phys. Rev.* A **95** 012329
[21] Anderson J T and Jochym-O'Connor T 2016 Classification of transversal gates in qubit stabilizer codes *Quantum Inf. Comput.* **16** 771–802
[22] Campbell E T 2016 *Blog Post - the Smallest Interesting Colour Code* (available at: https://earltcampbell.com/2016/09/26/the-smallest-interesting-colour-code/) (Accessed 28 September 2023)
[23] Kubica A, Yoshida B and Pastawski F 2015 Unfolding the color code *New J. Phys.* **17** 083026
[24] Soares W S and Da Silva E B 2018 Hyperbolic quantum color codes *Quantum Inf. Comput.* **18** 306–18
[25] Wu Y and Lee Y 2020 Self-orthogonal codes constructed from posets and their applications in quantum communication *Mathematics* **8** 1495
[26] Howell J A 1986 Spans in the module $(\mathbb{Z}_m)^s$ *Linear Multilinear Algebr.* **19** 67–77
[27] Nezami S and Haah J 2022 Classification of small triorthogonal codes *Phys. Rev.* A **106** 012437
[28] Vasmer M and Kubica A 2022 Morphing quantum codes *PRX Quantum* **3** 030319
[29] Litinski D 2019 A game of surface codes: large-scale quantum computing with lattice surgery *Quantum* **3** 128
[30] Vandersypen L M K and Chuang I L 2005 NMR techniques for quantum control and computation *Rev. Mod. Phys.* **76** 1037–69
[31] Meunier T, Calado V E and Vandersypen L M K 2011 Efficient controlled-phase gate for single-spin qubits in quantum dots *Phys. Rev.* B **83** 121403
[32] Hu J, Liang Q, Rengaswamy N and Calderbank R 2022 Mitigating coherent noise by balancing weight-2 z-stabilizers *IEEE Trans. Inf. Theory* **68** 1795–808
[33] Chao R and Reichardt B W 2018 Fault-tolerant quantum computation with few qubits *npj Quantum Inf.* **4** 42
[34] Breuckmann N P and Eberhardt J N 2021 Balanced product quantum codes *IEEE Trans. Inf. Theory* **67** 6653–74
[35] Laflamme R, Miquel C, Pablo Paz J and Hubert Zurek W 1996 Perfect quantum error correcting code *Phys. Rev. Lett.* **77** 198–201
[36] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press)