# Dynamical codes for hardware with noisy readouts

Peter-Jan H. S. Derks[1], Alex Townsend-Teague[1], Jens Eisert[1], Markus S. Kesselring[1], Oscar Higgott[2], and Benjamin J. Brown[3,4]

[1]Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany
[2]Department of Physics & Astronomy, University College London, WC1E 6BT London, United Kingdom
[3]IBM Quantum, T. J. Watson Research Center, Yorktown Heights, New York 10598, USA
[4]IBM Denmark, Sundkrogsgade 11, 2100 Copenhagen, Denmark

arXiv:2505.07658v2 [quant-ph] 1 Jul 2026

Dynamical stabilizer codes may offer a practical route to large-scale quantum computation. Such codes are defined by a schedule of error-detecting measurements, which allows for flexibility in their construction. In this work, we ask how best to optimise the measurement schedule of dynamically condensed colour codes in various limits of noise bias. We take a particular focus on the setting where measurements introduce more noise than unitary and idling operations – a noise model relevant to some hardware proposals. For measurement-biased noise models, we improve code performance by strategically repeating measurements within the schedule. For unbiased or $Z$-biased noise models, we find repeating measurements offers little improvement – somewhat contrary to our expectations – and investigate why this is. To perform this analysis, we generalise a metric called the teraquop footprint to the teraquop volume. This is the product of the number of qubits and number of rounds of measurements required such that the probability of a spacelike or timelike logical error occurring is less than $10^{-12}$. In most cases, we find differences in performance are primarily due to the number of rounds of measurements required, rather than the number of qubits – emphasising the importance of using the teraquop volume in the analysis. Additionally, our results provide another example of the importance of making use of correlated errors when decoding, in that using belief matching rather than minimum-weight perfect matching can turn a worst-performing code under a given noise model into a best-performing code.

**Code availability:**
https://github.com/peter-janderks/Floquet-codes-and-measurement-bias

Peter-Jan H. S. Derks: peter-janderks@hotmail.com
Alex Townsend-Teague: alex.townsend-teague@outlook.com
Oscar Higgott: Now at Google Quantum AI, Venice, CA 90291, USA.

## Contents


**1 Introduction** **3**
- 1.1 Previous numerical work . . . 3
- 1.2 Overview of results . . . 4

**2 Preliminaries** **10**
- 2.1 Measurement schedule . . . 11
- 2.2 Stabilizers and logical operators . . . 11
- 2.3 Noise models . . . 11
- 2.4 Detectors . . . 15
- 2.5 Decoding graph . . . 16
- 2.6 Logical errors . . . 19
- 2.7 Decoding . . . 19
- 2.8 $E$ and $M$ components . . . 21
- 2.9 Boundaries . . . 21

**3 Teraquop volume** **22**
- 3.1 Estimating the teraquop volume . . . 23
- 3.2 Blocks . . . 26
- 3.3 Limitations . . . 26

**4 Bias-tailoring I: choosing a schedule** **26**
- 4.1 Differences . . . 26
- 4.2 Direct parity measurement noise results . . . 28
- 4.3 Auxiliary qubit circuit noise results . . . 31

**5 Bias-tailoring II: repeating measurements** **31**
- 5.1 Detectors . . . 32
- 5.2 Logical errors . . . 35
- 5.3 Decoding graph . . . 38
- 5.4 Direct parity measurement noise results . . . 39
- 5.5 Auxiliary qubit circuit noise results . . . 39

**6 Discussion and outlook** **42**

**7 Author contributions** **45**

**8 Acknowledgements** **45**

**A Methodology** **49**

**B ZX-calculus and Pauli webs** **50**

**C Detectors cheatsheet** **51**

# 1 Introduction

To be able to execute currently known useful quantum algorithms, a computer with near-perfect qubits and gates is needed. As individual qubits and gates are far from perfect, the plan is to employ *quantum error correction* (QEC) to build a fault-tolerant quantum computer. A quantum computing scheme is considered fault-tolerant if its output is correct with high probability, even though its individual components are faulty. A popular strategy for implementing fault-tolerant quantum computing is to perform gates through lattice surgery on qubits encoded in the surface code [1–4]. The basic unit of a lattice surgery computer is a single qubit encoded in the surface code for a fixed unit of time. We refer to this unit as a *surface code block*. It is an exciting time for lattice surgery computing, with several experimental prototypes of surface code blocks demonstrated [5–11].

In parallel with the development of these prototypes, the recent discovery of *dynamic stabilizer codes* has led to new ways of implementing surface code blocks. This started with the introduction of the Hastings-Haah honeycomb code [12]. Subsequently, a larger class of codes containing the honeycomb code was discovered, which we refer to as *dynamically condensed colour codes* (DCCCs) [13]. Despite having 'colour code' in their name, DCCCs are topologically equivalent to the surface code, so can be used as surface code blocks [13–15].

Schemes for fault-tolerant quantum computing have an enormous resource cost for implementation with modern technology. For this reason, the main motivation for looking for new QEC codes is to reduce the resource requirements of a fault-tolerant computer. Importantly, the performance of a QEC code depends on the characteristics of the noise that affects a computer. Notably, for many hardware platforms, the noise can be biased towards certain types of operations. For example, a certain Pauli error ($Z$, say) can be more likely to occur than another ($X$ or $Y$, say). In the past, it has been shown that QEC codes can be tailored to deal with biased noise [16–25]. These results have been so promising that some quantum computers are now designed such that the noise is deliberately biased [17]. However, fewer works look at tailoring codes to hardware platforms where measurement errors are dominant – i.e. where measurements are more faulty than other types of quantum operations. This type of biased noise is relevant to multiple hardware proposals based on superconducting qubits [11, 26, 27].

In this work, we investigate the resource requirements of a lattice surgery quantum computer that uses a DCCC. Simulations have shown promising results for the *space* cost of two specific DCCCs under an unbiased noise model [13, 28, 29]. However, the *spacetime* cost provides a more precise measure for resource estimation of a lattice surgery quantum computer. This has not previously been studied for either uniform or biased noise models. To measure the spacetime cost, we generalise the teraquop footprint [28] to the *teraquop volume*, defined as the product of the number of qubits and the number of rounds required to achieve a logical error rate of $10^{-12}$, the so-called *teraquop regime*. A logical error rate of $10^{-12}$ or lower would be required for a lattice surgery quantum computer to be able to solve important problems [4, 30–33].

## 1.1 Previous numerical work

The footprint of what we'll refer to as the $X^1Y^1Z^1$ honeycomb code has been studied in Refs. [28, 29, 34, 35]. The footprints of hyperbolic and semi-hyperbolic $X^1Y^1Z^1$ honeycomb codes have been studied in Refs. [36, 37]. The memory threshold of what we'll refer to

as the $X^1Z^1$ honeycomb code has been studied in Refs. [13, 38]. The memory threshold of the $X^1Z^1$ honeycomb code tailored to biased noise has been studied in Ref. [39]. Measurements of $X^1Y^1Z^1$ and $X^1Z^1$ honeycomb code plaquette stabilizers have been performed on superconducting hardware [7].

## 1.2 Overview of results

We simulate a large class of DCCCs which are defined by their *measurement schedules*. All DCCCs are implemented by repeatedly measuring sets of $XX$, $YY$ or $ZZ$ operators according to certain rules. Additionally, one can choose to repeat a particular set of weight-two measurements any number of times [18]. A glossary of the terms used in the figures within this section is provided in Table 1.

### 1.2.1 Direct parity measurement noise results

Figures 1 and 2 show the performance of DCCCs for direct parity measurement noise models decoded using MWPM and belief matching. We have obtained volumes for 16 different noise models, which differ in measurement bias and $Z$ error bias. From the results presented in Figures 1 and 2, we can draw the following conclusions:

**Key point 1.1.** *When decoding with MWPM, it is better to only measure $XX$ and $ZZ$, while when decoding with belief matching, it is better to measure $XX$, $YY$, and $ZZ$.*

**Key point 1.2.** *Repeating measurements can improve the teraquop volume under measurement-biased noise.*

### 1.2.2 Auxiliary qubit circuit noise results

Three auxiliary qubit circuit noise models are simulated: *superconducting-inspired* (SI) noise, *standard depolarizing* (SD) noise, and *entangling measurement* (EM) noise. Figures 3 and 4 show the performance of DCCCs when decoded with MWPM and belief matching, respectively. We can draw the following conclusions:

**Key point 1.3.** *For all three auxiliary qubit circuit noise models, when decoding with MWPM, it is better to only measure $XX$ and $ZZ$, while when decoding with belief matching, it is better to measure $XX$, $YY$, and $ZZ$.*

**Key point 1.4.** *The noise bias in the SI noise model is not strong enough to make repeating measurements worthwhile.*

**Key point 1.5.** *For EM noise and decoding using MWPM the best DCCC is $X^2Z^2$.*

We consider this a key point because DCCCs are especially well suited for EM noise. Namely, for this noise model, it has been shown that the teraquop footprint of the $X^1Y^1Z^1$ code is smaller than the teraquop footprint of the surface code implemented with pair measurements [40, 41].

### 1.2.3 Remainder of this work

Section 2 contains background information on DCCCs. If you are familiar with DCCCs, detectors, logical errors, and decoding graphs, you can skip Section 2. In Section 3 we introduce and explain the teraquop volume, a key performance metric which we use to evaluate the performance of QEC protocols. The remainder of this work is dedicated to understanding the main results and explaining how these results have been obtained. We

first focus on the results comparing the $X^1Z^1$ and $X^1Y^1Z^1$ codes in Section 4. Following this we discuss how and why repeating measurement influences spacetime cost in Section 5. We finish this work with a summary and outlook in Section 6. The source code used to produce the results in this paper can be found at
https://github.com/peter-janderks/floquet_colour_codes_numerics.

| | |
|---|---|
| $X^aY^bZ^c$ code $a, b, c \in \mathbb{Z}_{>0}$ | DCCC whose measurement schedule consists of $a$ consecutive $XX$ measurements, $b$ consecutive $YY$ measurements, and $c$ consecutive $ZZ$ measurements on certain qubits. |
| $X^aZ^b$ code $a, b, c \in \mathbb{Z}_{>0}$ | DCCC whose measurement schedule consist of $a$ consecutive $XX$ measurements and $b$ consecutive $ZZ$ measurements on certain qubits. |
| Teraquop volume | The amount of qubits times the number of measurement rounds at which the probability of any logical error occurring is $10^{-12}$. |
| MWPM | A decoder that uses the minimum-weight perfect-matching algorithm. |
| Belief matching | A decoder that uses belief propagation and the minimum-weight perfect-matching algorithm. |
| Direct parity measurement noise | A circuit noise model in which single-qubit Pauli errors can occur before measurement rounds with probabilities $p_X$, $p_Y$ and $p_Z$, and measurement errors can occur with probability $p_m$. Assumes native two-qubit parity measurements. |
| Measurement error bias | $\frac{p_m}{p_X+p_Y+p_Z}$, assuming $p_X, p_Y, p_Z$ and $p_m$ are chosen such that the physical error rate is kept constant at $10^{-3}$. |
| $Z$ error bias | $\frac{2p_Z}{p_X+p_Y}$, assuming $p_X, p_Y, p_Z$ and $p_m$ are chosen such that the physical error rate is kept constant at $10^{-3}$. |
| Auxiliary qubit circuit noise | A circuit noise model in which a $q$-qubit Pauli error can occur with probability $p_g$ after each gate $g$ on $q$ qubits, and measurement errors can occur with probability $p_m$. Multi-qubit measurements are decomposed using auxiliary qubits. |
| Standard depolarizing noise | An auxiliary qubit circuit noise model that assumes multi-qubit measurements are implemented by single-qubit measurements on auxiliary qubits, and all error probabilities are equal. |
| Superconducting-inspired noise | An auxiliary qubit circuit noise model that assumes multi-qubit measurements are implemented by single-qubit measurements on auxiliary qubits, and the error probabilities model those experienced by real superconducting hardware. |
| Entangling measurement noise | An auxiliary qubit circuit noise model that assumes multi-qubit measurements are implemented directly, and all error probabilities are equal. |

**Table 1:** Glossary for terms used in Figs. 1, 2, 3, and 4. More information on the noise models is given in Subsection 2.3.

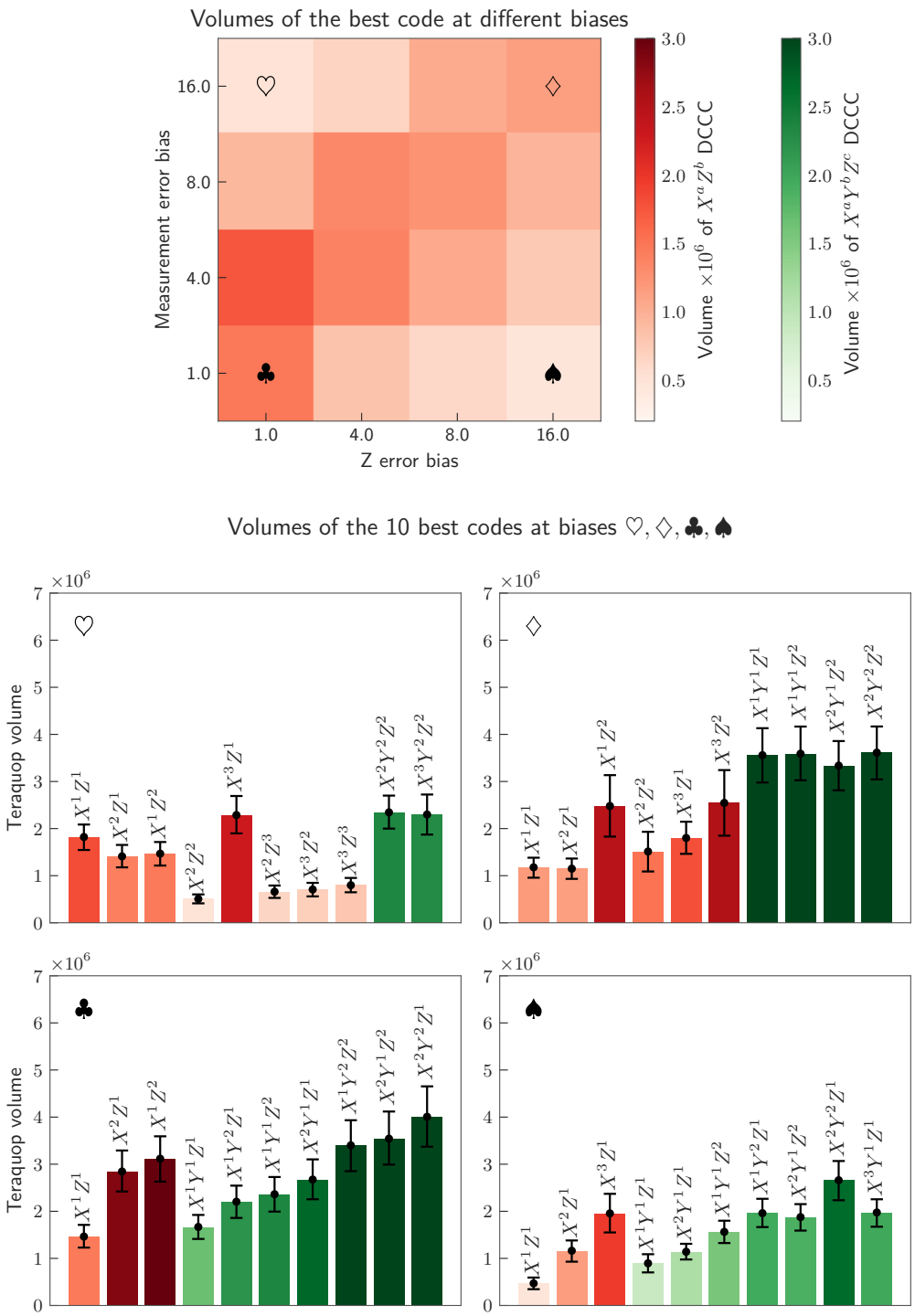



**Figure 1:** Squares in the top patchwork plot are red (green) if the code with the smallest teraquop volume is an $X^aZ^b$ code ($X^aZ^bY^c$ code). A darker shade indicates a higher volume. The four bar plots show the footprints of the 10 best codes at the four limits of the simulated biases. The error bars are explained in Section A. Decoding is performed using MWPM. For all noise models investigated here, the $X^aZ^b$ code outperforms the $X^aY^bZ^c$ code. Repeating measurements improves performance when the noise is biased strongly towards measurement errors only, and has negligible effect otherwise. Codes are ordered first by type ($X^aZ^b$ code, then $X^aY^bZ^c$ code), then by increasing schedule length ($a+b$ or $a+b+c$). Codes with equal schedule length are ordered by balance, with more uniformly distributed repetition parameters appearing first. For noise model details, definitions, and naming conventions see Table 1.

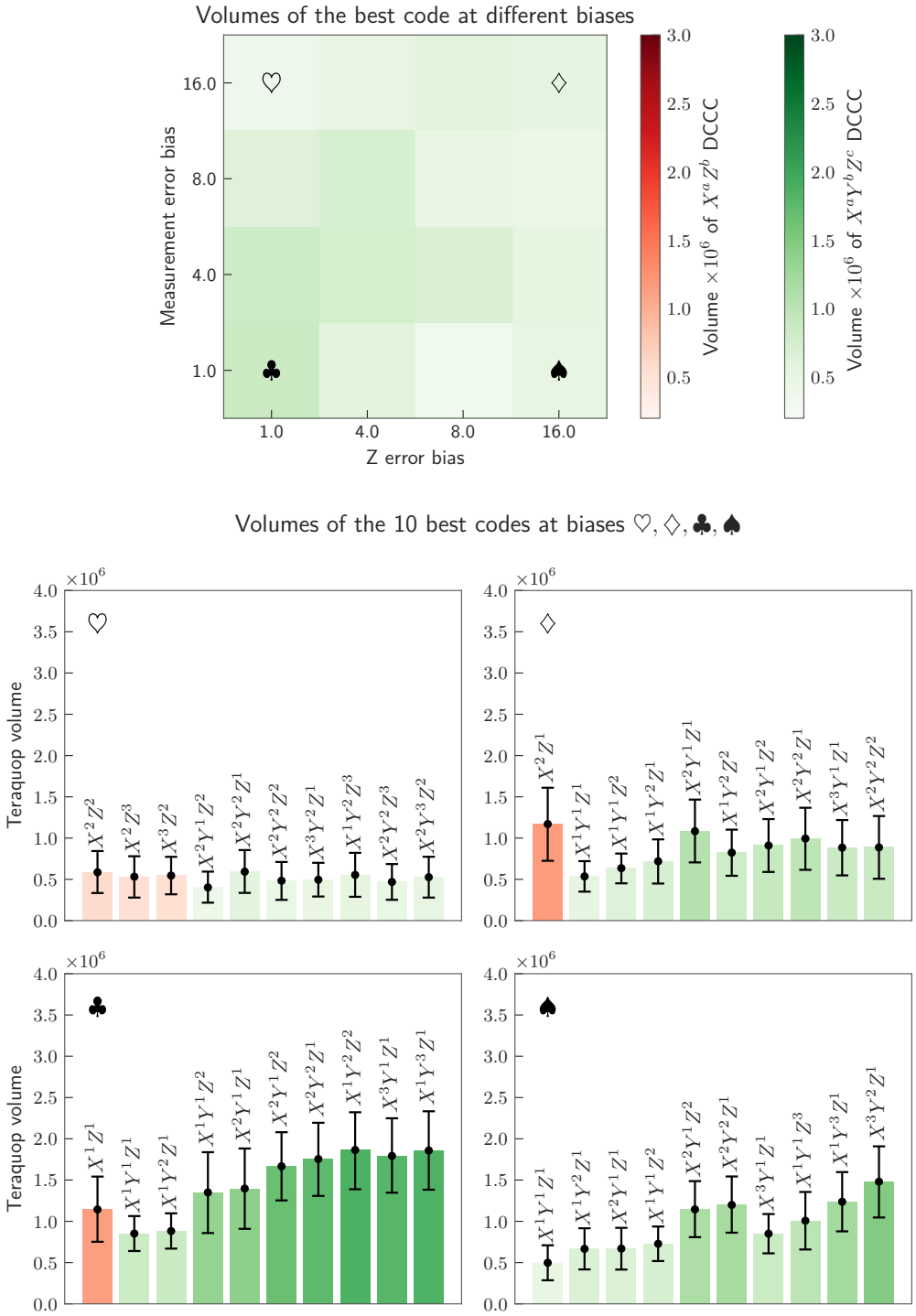



**Figure 2:** Same plot as in Figure 1 but with decoding performed using belief matching. In contrast to Figure 1, for all noise models investigated here, the $X^aY^bZ^c$ code outperforms the $X^aZ^b$ code. However, as before, repeating measurements improves performance when the noise is biased strongly towards measurement errors only, but has negligible effect otherwise.

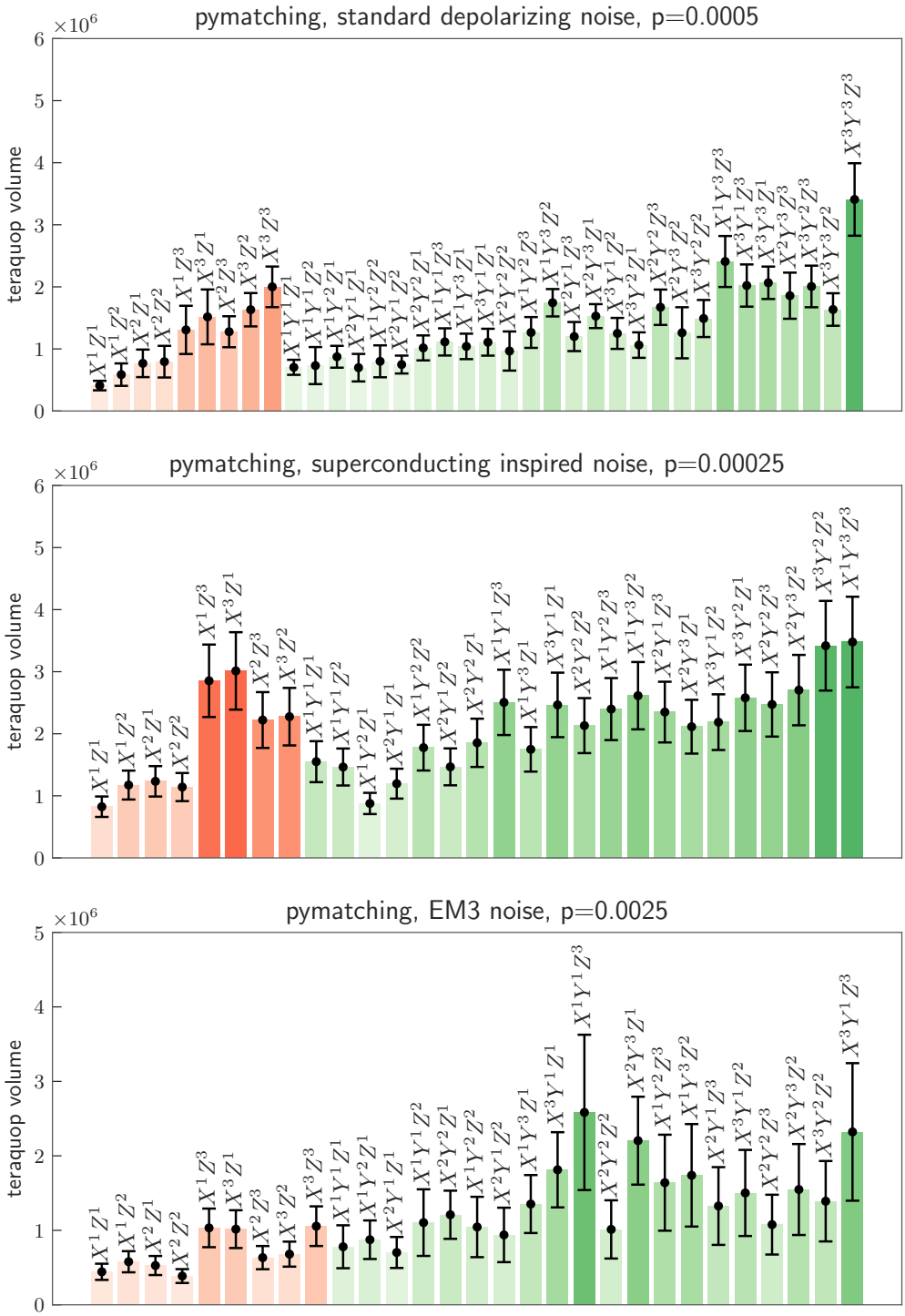



**Figure 3:** Teraquop volumes of $X^aY^bZ^c$ and $X^aZ^b$ codes for three auxiliary qubit circuit noise models; standard depolarizing noise with $p = 0.0005$, superconducting inspired noise with $p = 0.00025$, and entangling measurement noise with $p = 0.0025$. Decoding is performed using MWPM. Codes are ordered first by type ($X^aZ^b$ code, then $X^aY^bZ^c$ code), then by increasing schedule length ($a+b$ or $a+b+c$). Codes with equal schedule length are ordered by balance, with more uniformly distributed repetition parameters appearing first. In each case, the code with the best teraquop volume is from the $X^aZ^b$ family, though the difference is small. Repeating measurements broadly has negligible effect, though does seem to improve the performance of the $X^1Y^1Z^1$ code in the superconducting inspired noise model, which features a moderate measurement bias. The error bars are explained in Section A.

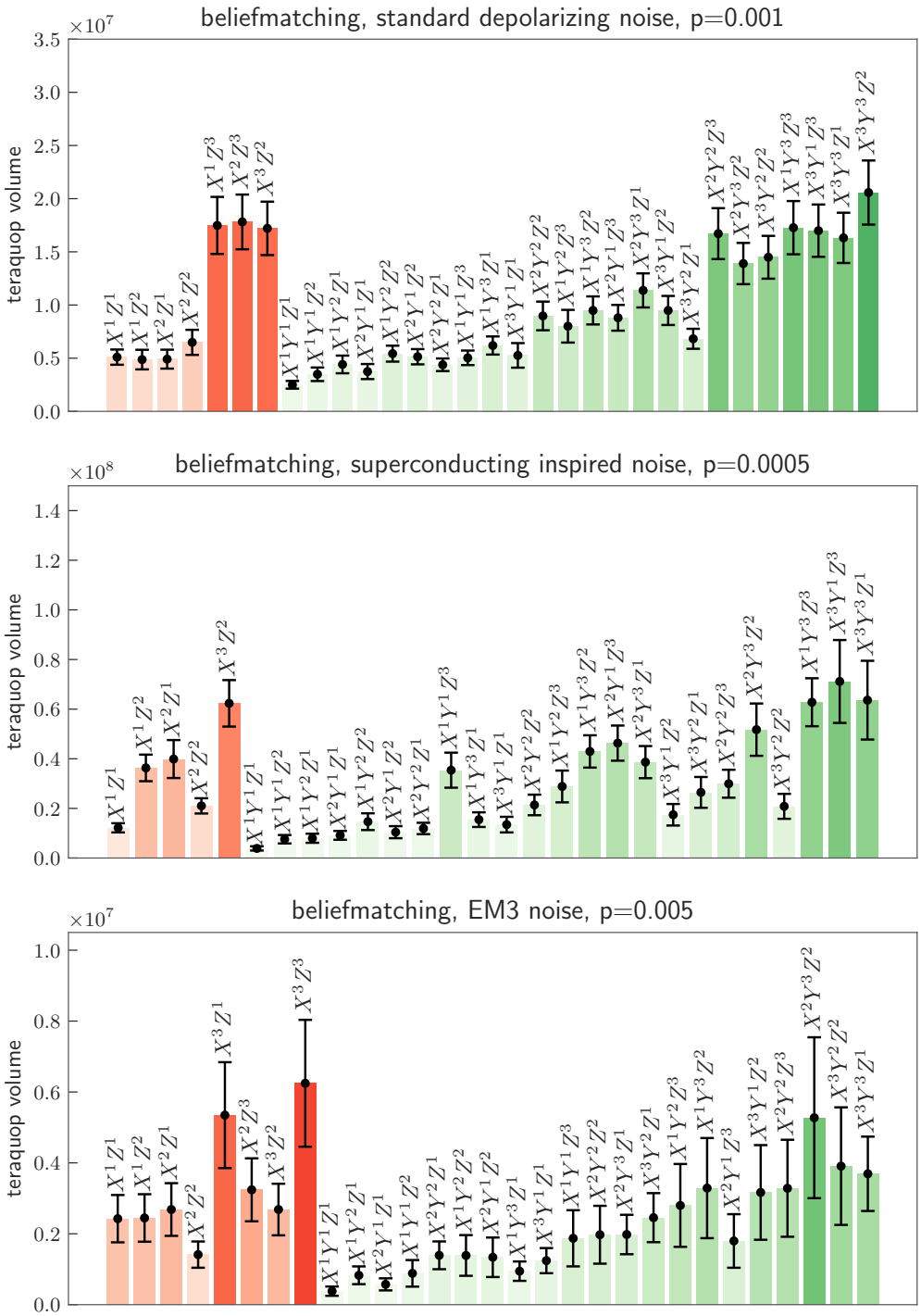



**Figure 4:** Same plot as in Figure 3 but with decoding performed using belief matching and higher physical error rates; standard depolarizing noise with $p = 0.001$, superconducting inspired noise with $p = 0.0005$, and entangling measurement noise with $p = 0.005$. As before, repeating measurements broadly has negligible effect, though does improve the performance of the $X^1Z^1$ code in the entangling measurements noise model.

## 2 Preliminaries

Here we recap *dynamically condensed colour codes* (DCCCs) [13] and how such codes can be used to detect and correct errors. DCCCs are a type of *dynamic stabilizer code*. We refer those interested in a more general introduction to the latter topic to Refs. [42, 43]. We will assume the reader is comfortable with the stabilizer formalism. Otherwise, we recommend Refs. [44, 45] for a basic introduction to this topic.

DCCCs can be defined in terms of the colour code's *boson table*, a $3 \times 3$ grid with columns labelled by colours $c, m$ and $y$ (cyan, magenta, and yellow) and rows labelled by letters $X, Y$ and $Z$ (non-identity Pauli matrices). Cells of the table are thus colour-Pauli pairs $(\kappa, P)$. Often we will want to talk about generic colours $\{\kappa, \kappa', \kappa''\} = \{c, m, y\}$ and generic Pauli matrices $\{P, P', P''\} = \{X, Y, Z\}$, for which we will use a boson table with these generic labels instead:

$$\begin{array}{c|c|c} c, X & m, X & y, X \\ \hline c, Y & m, Y & y, Y \\ \hline c, Z & m, Z & y, Z \end{array} \quad \text{or} \quad \begin{array}{c|c|c} \kappa, P & \kappa', P & \kappa'', P \\ \hline \kappa, P' & \kappa', P' & \kappa'', P' \\ \hline \kappa, P'' & \kappa', P'' & \kappa'', P'' \end{array} \tag{1}$$

A DCCC can be defined on any three-face-colourable, three-valent 2D lattice. Edges of such a lattice inherit a canonical colouring; an edge separating faces of colours $\kappa$ and $\kappa'$ is assigned the third colour $\kappa''$. On every vertex of the lattice we place a qubit. Throughout this work we use the honeycomb lattice. We also assume throughout that this lattice is placed on a torus; we comment on the limitations of this in Subsection 2.9.

For every cell $(\kappa, P)$ of the boson table, we can define certain operators on the lattice. Namely, for every $\kappa$-coloured edge $e_\kappa = (v_0, v_1)$, we can define a $(\kappa, P)$ *edge operator* $e_{\kappa,P} := P_{v_0} P_{v_1}$, and the set of all such edge operators as $E_{\kappa,P}$. Similarly, for every $\kappa$-coloured face $f_\kappa$ with vertices $v_0, v_1, \ldots, v_5$ we can define a $(\kappa, P)$ *face operator* $f_{\kappa,P} := P_{v_0} P_{v_1} \ldots P_{v_5}$, and the set of all such face operators as $F_{\kappa,P}$. Note that each face operator $f_{\kappa,P}$ can be made from edge operators in two ways; as the product of the three $(\kappa', P)$ edge operators around $f$, or the three $(\kappa'', P)$ edge operators. Additionally, it can be written as the product of the other two face operators $f_{\kappa,P'}$ and $f_{\kappa,P''}$ on $f$, up to global phase. Examples of such operators are shown in Figure 5.

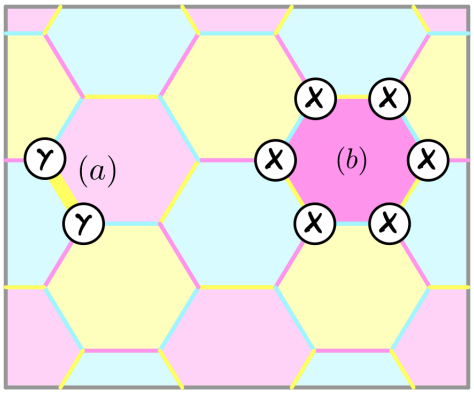



**Figure 5:** $(a)$ A $Y^{\otimes 2}$ yellow edge operator $e_{y,Y}$. $(b)$ An $X^{\otimes 6}$ magenta face operator $f_{m,X}$.

## 2.1 Measurement schedule

A DCCC, then, is a dynamic stabilizer code defined by repeatedly measuring sets $E_{\kappa,P}$ of all $(\kappa, P)$ edge operators. We say it has *measurement schedule* $(\langle E_{\kappa_0,P_0}\rangle, \langle E_{\kappa_1,P_1}\rangle, \ldots)$. For DCCCs as defined in Ref. [13, Section 7], there is a single restriction: if at time $t$ the operators $E_{\kappa_t,P_t}$ are measured, then at time $t+1$ we must choose a new set $E_{\kappa_{t+1},P_{t+1}}$ of edges to measure, such that cell $(\kappa_{t+1}, P_{t+1})$ shares neither a row nor column with $(\kappa_t, P_t)$ in the boson table. Later, in Section 5, we relax this restriction and investigate DCCCs with repeated measurements – i.e. we allow $E_{\kappa_{t+1},P_{t+1}} = E_{\kappa_t,P_t}$. But until then, the two DCCCs we will focus on are the $X^1Y^1Z^1$ *honeycomb code* from Ref. [12] and the $X^1Z^1$ *honeycomb code* from Ref. [13] [1].

These are defined by the measurement schedules

$$\begin{aligned} \mathcal{M}_{X^1Y^1Z^1} &= (\langle E_{c,X}\rangle, \langle E_{m,Y}\rangle, \langle E_{y,Z}\rangle)\,, \\ \mathcal{M}_{X^1Z^1} &= (\langle E_{c,X}\rangle, \langle E_{m,Z}\rangle, \langle E_{y,X}\rangle, \langle E_{c,Z}\rangle, \langle E_{m,X}\rangle, \langle E_{y,Z}\rangle)\,. \end{aligned} \tag{2}$$

Much of this work is concerned with the properties of these two codes and their performances under different noise models and decoders.

## 2.2 Stabilizers and logical operators

In any dynamic stabilizer code on $n$ qubits, once a measurement schedule has been defined, two further sequences can be derived; the *instantaneous stabilizer groups* (ISGs) $\mathcal{S} = (\mathcal{S}_{-1}, \mathcal{S}_0, \mathcal{S}_1, \ldots)$ and the *instantaneous logical Pauli groups* (ILPGs) $\mathcal{L} = (\mathcal{L}_{-1}, \mathcal{L}_0, \mathcal{L}_1, \ldots)$. These respectively track – at each timestep $t$ – which Paulis are stabilizers of our code, and which Paulis are representatives of logical Pauli operators. Explicit presentations for each $\mathcal{S}_t$ and $\mathcal{L}_t$ can be found via the stabilizer formalism [45, Section 6.2]. We let $r_t$ denote the rank of $\mathcal{S}_t$. The group $\mathcal{L}_t$ is defined to be $\mathcal{N}(\mathcal{S}_t)/\mathcal{S}_t$, the normaliser of $\mathcal{S}_t$ in the Pauli group, modulo $\mathcal{S}_t$. Every $\mathcal{L}_t$ is isomorphic to the Pauli group $\mathcal{P}_{k_t}$ on $k_t := n - r_t$ qubits [45, Theorem 3.11]. If there is a timestep $T$ such that for all $t \geq T$ we have $r_t = r_T =: r$, then we say the code is *established*. Correspondingly, for such $t \geq T$, we get $k_t = k_T =: k$. We thus say such a code encodes $k$ logical qubits.

Every DCCC on a torus has an establishment time $T$, after which it encodes $k = 2$ logical qubits. Alternatively, we can always just ignore one of the logical qubits and pretend it only encodes one qubit; this makes our work more straightforward to compare with the more easily implementable planar case, which encodes just one logical qubit, and allows us to tailor the code dimensions to biased noise models later on. In the rest of this work, we will do exactly this. In Figure 7, we show representatives of logical $X$ and $Z$ operators for the $X^1Y^1Z^1$ and $X^1Z^1$ honeycomb codes at timesteps $t$ mod 6. For reasons we discuss shortly (Subsection 2.8), we will actually call these $E$ and $M$ logical operators rather than $X$ and $Z$.

## 2.3 Noise models

Before we can discuss how DCCCs are used to detect and correct errors, we need to meet the errors in question.

[1] The $X^1Y^1Z^1$ honeycomb code is equivalent to the Hastings-Haah code up to conjugating certain qubits by Cliffords.

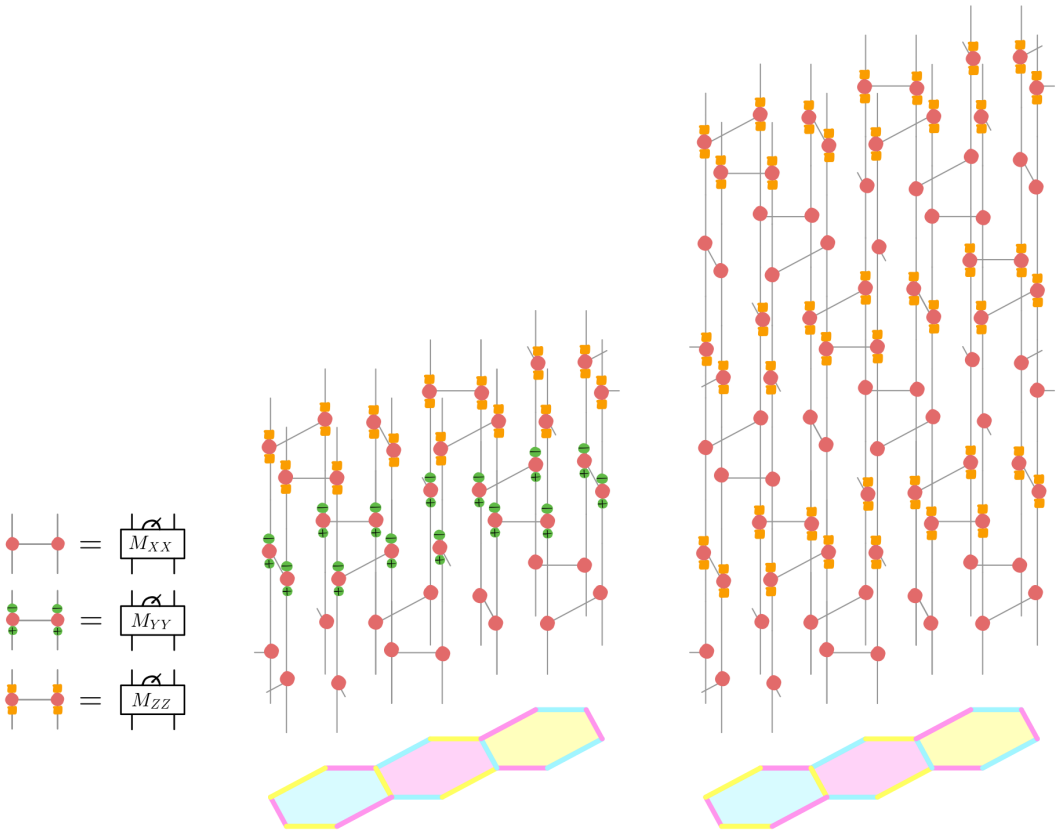



**Figure 6:** For a subset of qubits, we show the sequence of measurements defining the $X^1Y^1Z^1$ (middle) and $X^1Z^1$ (right) honeycomb codes. This figure uses the ZX-calculus [46], a rigorous graphical language for quantum mechanics. For those unfamiliar with the ZX-calculus, a crash course is given in Section B, with proper introductions available in Refs. [47, 48]. Alternatively, you're absolutely free to ignore these diagrams' rigorous meaning and treat them as a visual guide to the algebra in the text – much in the same way as Figure 5 has no rigorous meaning. In which case, all you need to know for now is that the diagrams are read bottom-to-top and the three diagrams on the left of the figure denote measurements of $X \otimes X$, $Y \otimes Y$ and $Z \otimes Z$ respectively.

### 2.3.1 Direct parity measurement noise model

In a quantum circuit that implements a DCCC – one that repeatedly performs the defining sequence of measurements $\mathcal{M} = (\langle E_{\kappa_0,P_0}\rangle, \langle E_{\kappa_1,P_1}\rangle, \ldots)$ – in addition to the $n$ *data qubits*, we may require additional *auxiliary qubits* in order to perform the measurements. In the *direct parity measurement noise model*, we assume that two-qubit parity measurements (such as $XX$, $YY$, $ZZ$) can be performed natively. We assume that at every timestep $t$, before a generating set for $\mathcal{M}_t$ is measured, any of the $n$ data qubits can be afflicted by a single-qubit Pauli error $X$, $Y$, or $Z$ with probability $p_X$, $p_Y$, or $p_Z$ respectively. We call these *data qubit errors*. We also assume that every measurement can be erroneously reported with probability $p_m$ – i.e. the actual measurement outcome is $\pm 1$, but the reported measurement outcome is $\mp 1$. We call this a *measurement error*, or *measurement noise*. A noise model is *biased* unless all of these probabilities are equal. Tailoring DCCCs to be effective against biased noise models will be a key theme of this work. The two types of bias we will focus on are *measurement bias* and *$Z$ bias*. In the direct parity measurement noise model, the measurement bias is defined as $\eta_m = p_m/(p_X + p_Y + p_Z)$, and the $Z$ bias is defined as $\eta_Z = 2p_Z/(p_X + p_Y) = p_Z/p_X$, assuming $p_X = p_Y$.

In any circuit we keep the total physical error rate $p$ fixed as

$$p = \frac{p_m M + (p_X + p_Y + p_Z)Q}{M + 3Q} \tag{3}$$

where $M$ is the number of measurements in the circuit and $Q$ is the number of data qubit noise locations. To generate Figures 1 and 2, we have used $p = 10^{-3}$.

### 2.3.2 Auxiliary qubit circuit noise model

An *auxiliary qubit circuit noise model* decomposes multi-qubit measurements into elementary gates using auxiliary qubits, in contrast to the direct parity measurement noise model which assumes native two-qubit parity measurements. In an auxiliary qubit circuit noise model, we ignore the distinction between data and auxiliary qubits. Instead, we just assume that after every gate $g$ on $q$ qubits, any weight-$q$ Pauli operator can occur on these qubits with some probability $p_g$. We continue to assume that measurement noise can occur. At this point, the actual circuits used to implement the DCCCs become relevant. We consider two different families of circuits, *single-qubit measurement circuits* and *two-qubit measurement circuits*. In the first, the measurements of edge operators $e_{\kappa,P}$ are implemented via auxiliary qubits, two-qubit gates, and single-qubit Pauli measurements. In

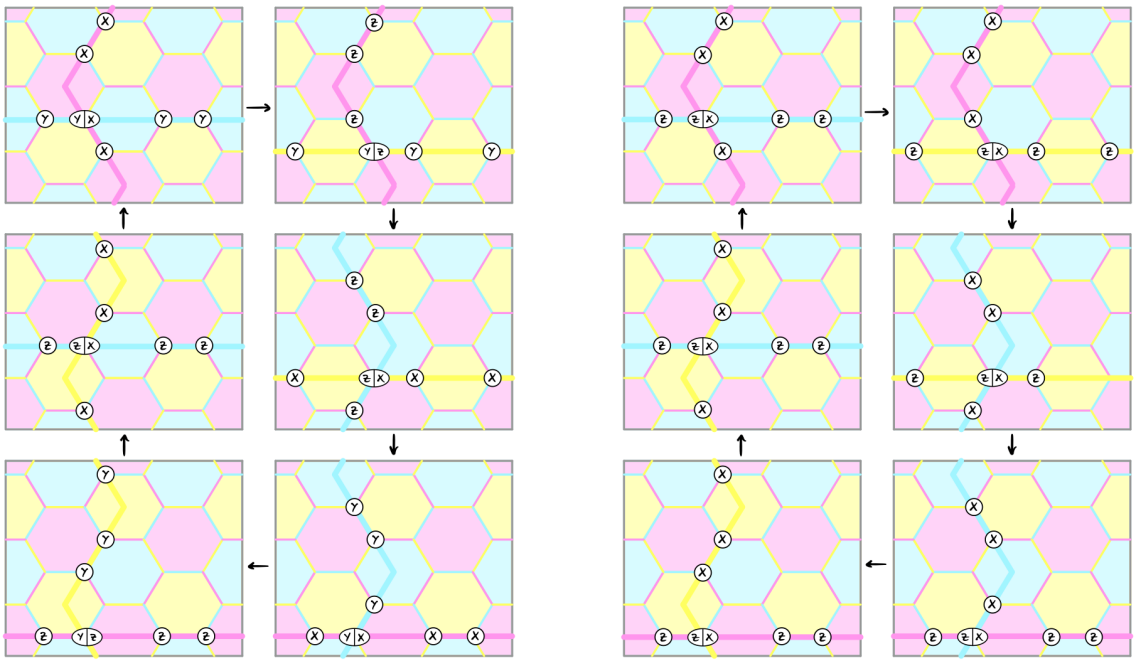


**Figure 7:** Representatives of logical $E$ and $M$ operators for the $X^1Y^1Z^1$ (left) and $X^1Z^1$ (right) honeycomb codes at timesteps $t$ mod 6. Timestep 0 mod 6 is the top left subfigure in each cycle. Boundaries are periodic, forming a torus. Throughout this work we say a DCCC encodes only one logical qubit - we just ignore the other one. We arbitrarily choose that the $E$ logical operator wraps vertically around the torus in this figure, with the $M$ operator wrapping around horizontally. Importantly, the representative of the $E$ operator in the $X^1Z^1$ code always consists of Pauli $Z$ operators, and the $M$ operator always consists of Pauli $X$ operators. In the $X^1Y^1Z^1$ code, in contrast, the representatives of both the $E$ and $M$ operator consist one third of the time of Pauli $X$ operators, one third of the time of $Y$ operators and one third of the time of $Z$ operators. This will be important when discussing the effects of biased noise later on.

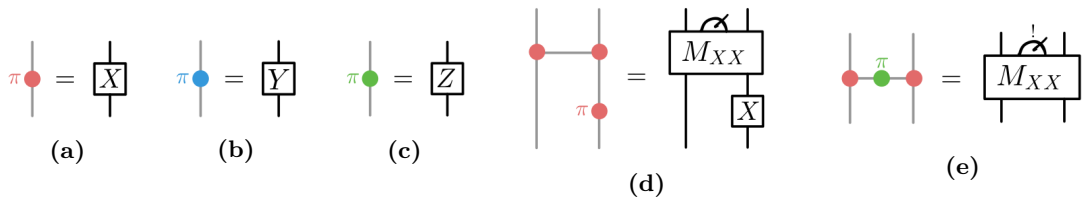



**Figure 8:** Subfigures (a), (b) and (c) show the non-identity Pauli gates as ZX-diagrams. Subfigures (d) and (e) show examples of errors in a phenomenological noise model. Specifically, (d) shows an $X$ error occurring before an $e_{\kappa,X}$ measurement, while (e) shows a measurement error on an $e_{\kappa,X}$ measurement. Indeed, throughout this work, a measurement error on any $e_{\kappa,P}$ measurement will correspond in the ZX-calculus to a two-legged green $\pi$-spider on a 'horizontal' wire.

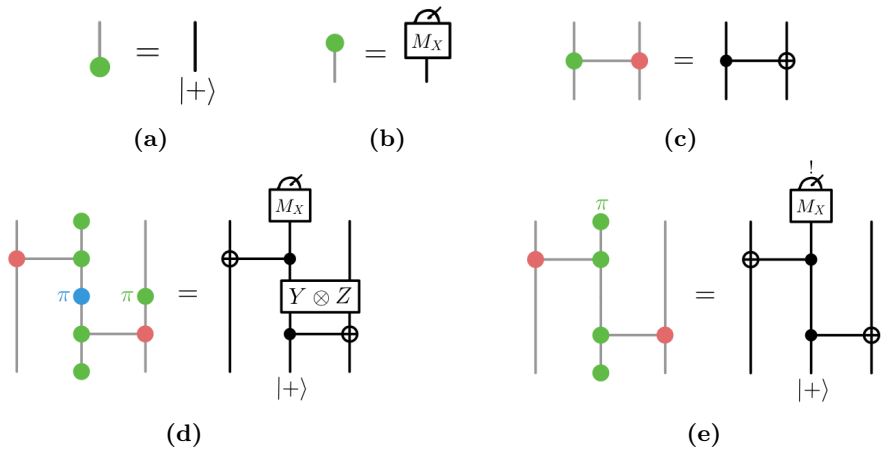



**Figure 9:** Subfigures **(a)**, **(b)** and **(c)** show the $|+\rangle$ state, $X$ measurement and $CX$ gate respectively as ZX-diagrams. Subfigures **(d)** and **(e)** show examples of errors in an auxiliary qubit circuit noise model, where two-qubit gates and a single-qubit measurement are used to measure $e_{\kappa,X}$. Specifically, **(d)** shows a $Y \otimes Z$ error occurring immediately after a $CX$ gate, while **(e)** shows a measurement error on an $X$ measurement.

| Operation | Noise applied after operation | SD | SI | EM3 |
|---|---|---|---|---|
| CX, CY, CZ | $\{I, X, Y, Z\}^{\otimes 2} - \{I \otimes I\}$ | $p$ | $p$ | - |
| Init. $\|+\rangle$ | $Z$ | $p$ | $2p$ | $p$ |
| Init. $\|i\rangle$ | $Y$ | $p$ | $2p$ | $p$ |
| Init. $\|0\rangle$ | $X$ | $p$ | $2p$ | $p$ |
| Meas. $X, Y, Z$ | bit-flip | $p$ | $5p$ | $p$ |
| Idle | $X, Y, Z$ | $p$ | $p/10$ | $p$ |
| Resonator idle* | $X, Y, Z$ | 0 | $2p$ | 0 |
| $M_{PP}$** | $\{I, X, Y, Z\}^{\otimes 2} \times$ {bit-flip, no-flip} | - | - | $p$ |

**Table 2:** Exact details of the noise models used for numerical simulations. The leftmost column contains all types of operations that are applied in the circuits. The second column lists which errors can occur after a gate. If multiple errors are given, one is chosen uniformly at random. *Resonator idle refers to a circuit location during which a qubit is not measured or reset in a time step during which other qubits are being measured or reset. **With the $M_{PP}$ operation the two-qubit Pauli errors are applied before the measurement.

the second, the measurements of edge operators are implemented 'natively' – no auxiliary qubits are used.

The single-qubit measurement circuits are simulated with two different auxiliary qubit circuit noise models: *standard depolarizing* (SD) and *superconducting-inspired* (SI). The SI model is taken from Ref. [28]. There is a slight difference between our SI model and *SI1000* in Ref. [28], namely that in the SI1000 noise model it is assumed that the only two-qubit gate used is $CZ$, while the circuits here contain $CX$, $CY$, and $CZ$ gates. The two-qubit measurement circuits are simulated with a different auxiliary qubit circuit noise model referred to as *entangling measurements* (EM) noise. The EM noise model is exactly the EM3 model taken from Ref. [29]. The different gates, errors, and probabilities of each model are shown in Table 2.

## 2.4 Detectors

We detect and correct errors using *detectors* [49, 50]. A detector is a set of measurement outcomes $\{m_1, \ldots, m_b\}$ whose product is deterministic in the absence of noise. In a stabilizer code, detectors are particularly simple; they consist of the outcomes of measuring the same stabilizer twice at consecutive timesteps. In dynamic stabilizer codes, we will not necessarily measure the same operators at consecutive timesteps, so detectors can look more complicated. However, they are still easy to find using the stabilizer formalism; one is created anytime we measure a Pauli operator $p$ such that $p$ or $-p$ is a stabilizer for the code at a particular timestep. If we ever find that the product $m_1 \ldots m_b$ differs from the expected value, then we can conclude that some errors must have affected our system. We say that such a detector has been *violated* by these errors. Often we'll write a detector $\{m_1, \ldots, m_b\}$ as a formal product $m_1 \ldots m_b$ with powers taken mod 2.[2]

Let's now look at DCCC detectors. We let $m_t(e_{\kappa_t,P_t})$ denote the outcome of measuring the edge operator $e_{\kappa_t,P_t}$ at time $t$, and let $e \in f$ mean edge $e$ borders face $f$. DCCCs have detectors[3] of the form $d = \prod_{t\in\tau} \prod_{e_{\kappa_t}\in f_\kappa} m_t(e_{\kappa_t,P_t})$ for some set of timesteps $\tau$ and some $\kappa$-coloured face $f_\kappa$. For example, on the left of Figure 10 we show a detector in the $XYZ$ honeycomb code, and on the right a detector in the CSS honeycomb code, both associated to some yellow face $f_y$. As a formal product, with timesteps taken modulo 6, the former is written as

$$\prod_{e_m\in f_y} m_4(e_{m,Y}) \prod_{e_c\in f_y} m_3(e_{c,X}) \prod_{e_m\in f_y} m_1(e_{m,Y}) \prod_{e_c\in f_y} m_0(e_{c,X}), \tag{4}$$

while the latter is

$$\prod_{e_m\in f_y} m_4(e_{m,X}) \prod_{e_c\in f_y} m_0(e_{c,X}). \tag{5}$$

[2]By formal product, we mean we forget that symbols like $m_j$ are actually stand-ins for values $-1$ and 1, and treat the symbols just as objects to be moved around algebraically. By taking powers mod 2, we mean – for example – the formal product $m^3$ is the same as $m^1 = m$. It seems only ∼16.7% of the authors of this work think formal products are the best way to write detectors. If you prefer linear algebra, you can also think of a detector as a binary vector – see Ref. [50]. Or you can continue to just think of them as sets of measurement outcomes.

[3]This statement only holds if we make a distinction between detectors and *observables*. An observable here doesn't have the usual meaning of a self-adjoint operator. Instead it refers to a special detector that's used to test whether the decoder correctly decoded the errors in a circuit [49, 50]. There are indeed observables that are not of this form.

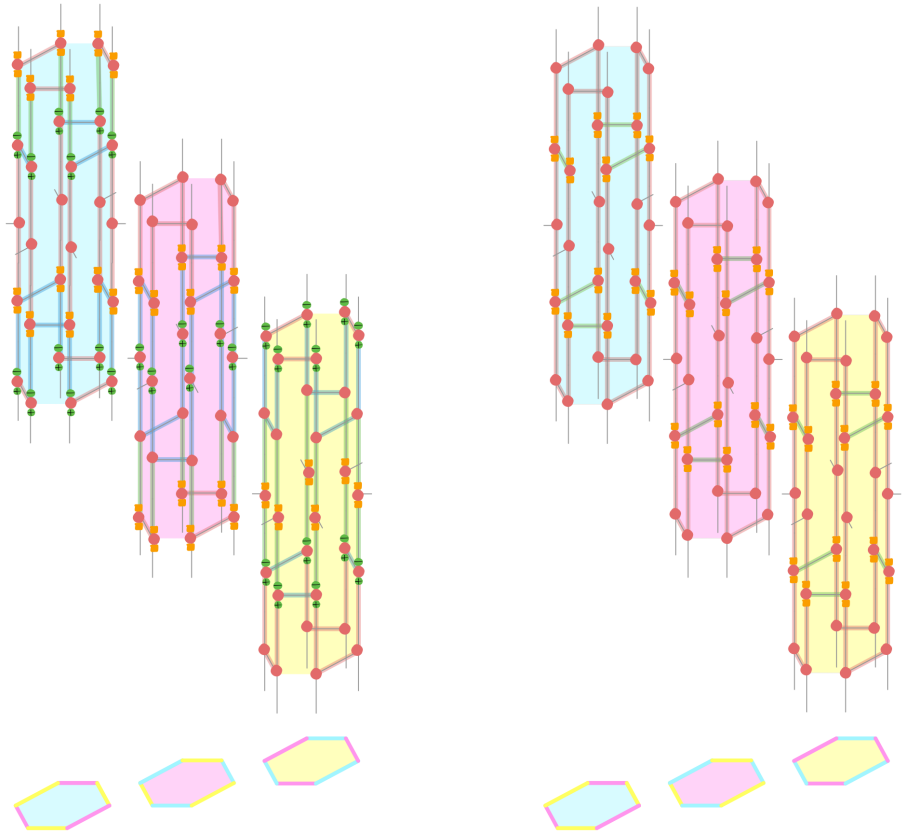


**Figure 10:** A subset of detectors of the $X^1Y^1Z^1$ code (left) and the $X^1Z^1$ code (right). Certain wires in this ZX-diagram are highlighted red, green or blue; they constitute a *Pauli web*, which we go into in more detail in Section B. In essence, one can think of a Pauli web as telling us which types of errors violate this detector at which timesteps. Pauli errors and measurement errors are represented in the ZX-calculus as 2-legged $\pi$-spiders (Figures 8 and 9). A detector is violated by a set of errors if there's an odd number of them of a different colour to the highlighted wire they sit on (see Figure 11, for example). Pauli webs also tell us which measurements are actually involved in a detector and which are not; it's all mesurements corresponding to non-vertical edges highlighted red or blue.

In Figure 10 the detectors are shown graphically via *Pauli webs*, as explained in more detail in the caption.

## 2.5 Decoding graph

The *decoding graph* of a circuit implementing a quantum error-correcting code, under a given noise model, is a weighted hypergraph formed from the detectors of the code and the errors of the noise model. The vertices of this hypergraph are exactly the detectors of the code. For every possible independent error $e$ permitted by the noise model, letting $d_1, \ldots, d_b$ be the detectors it violates and $p_e$ be the probability of it occurring, a hyperedge also labelled $e$ is added to the graph, spanning vertices $\{d_1, \ldots, d_b\}$ and having weight $p_e$. As written, there may be multiple hyperedges spanning the same set of $\{d_1, \ldots, d_b\}$ of vertices; if this is the case, they can be combined into a single hyperedge spanning this set of vertices, whose weight $p$ is the probability of an odd number of the corresponding errors occurring. We will alternatively write a hyperedge $e = \{d_1, \ldots, d_b\}$ as a formal product

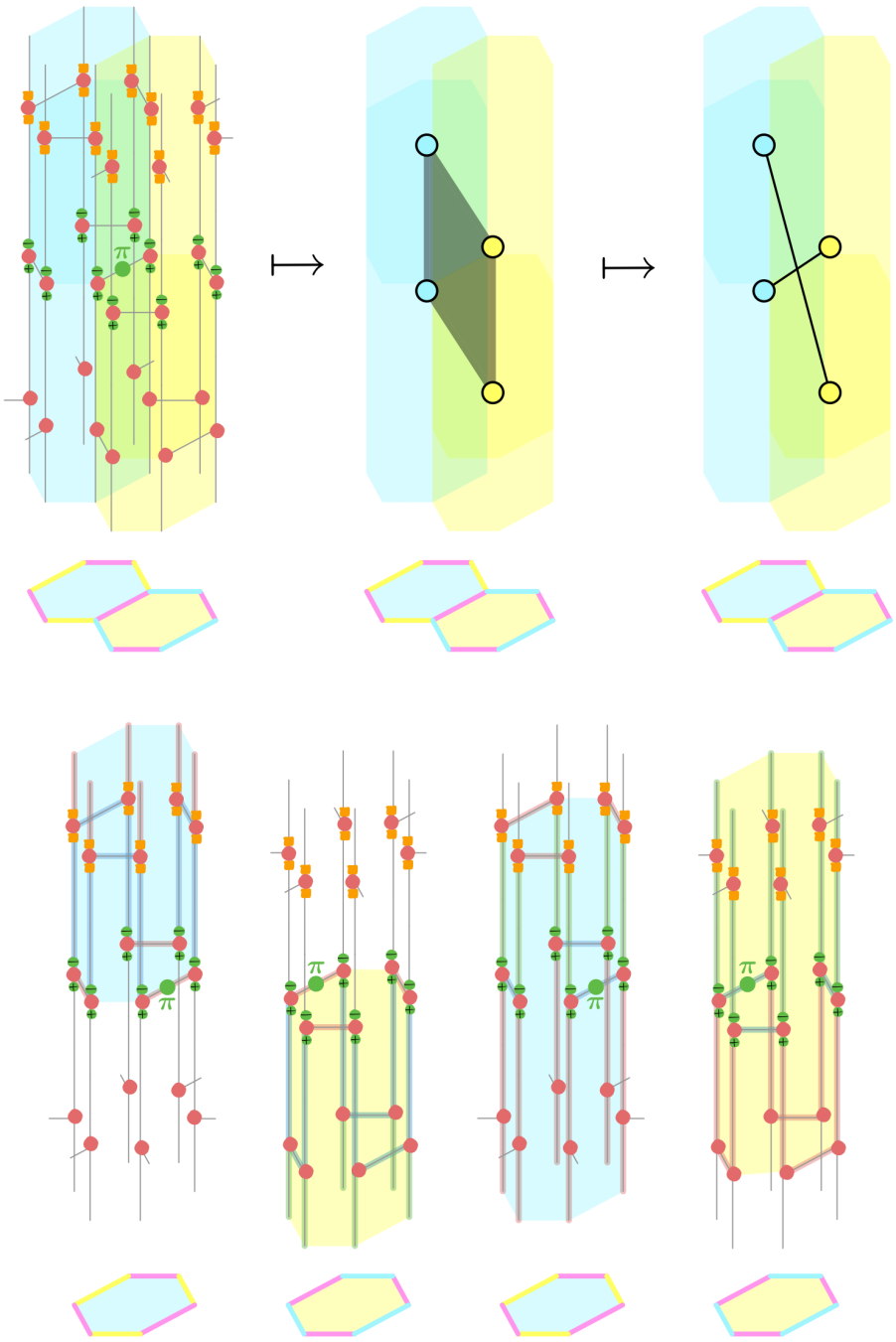



**Figure 11:** An $e_{m,Y}$ measurement error in the $X^1Y^1Z^1$ honeycomb code, which violates four detectors. In the top left subfigure, we show the error as a ZX-diagram. Since it violates four detectors, it corresponds to a hyperedge in the decoding graph (top middle). This hyperedge then decomposes into two ordinary edges (top right). The longer of these two edges in the subfigure is a remnant edge, because there is no actual error in the noise model that corresponds to it, unlike the shorter of the two edges. The bottom subfigures show in more detail the four detectors that this error violates. One can check that these really are violated by noting that, in each case, the edge on which the green $\pi$-spider representing the error sits is highlighted a colour other than green, as explained in Figure 10 and in more detail in Section B.

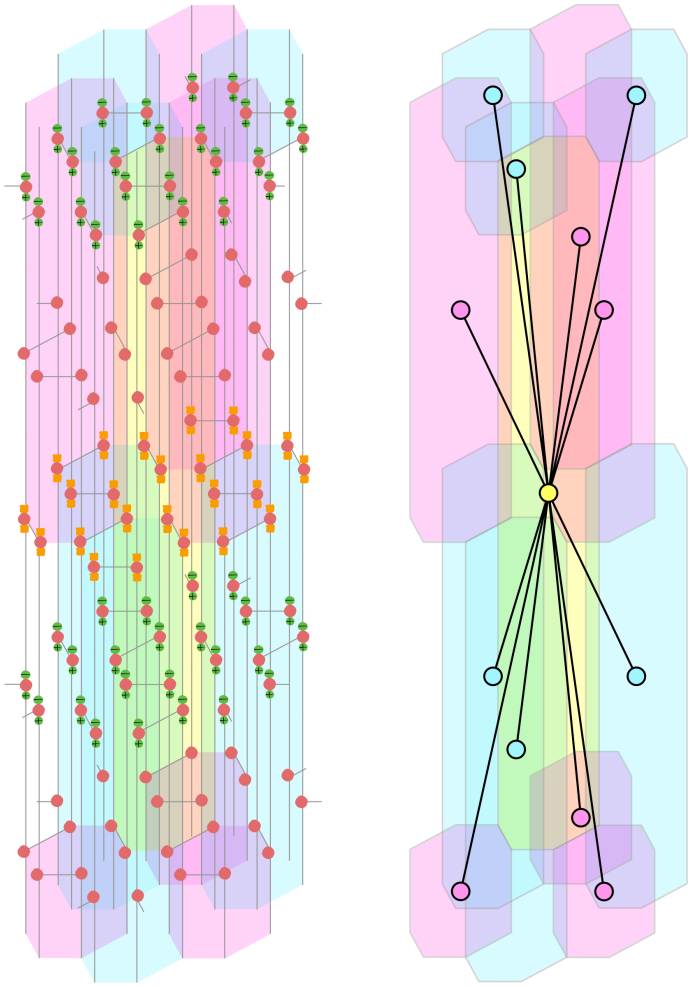


**Figure 12:** On the left we show five timesteps from the $X^1Y^1Z^1$ code. Areas coloured cyan, magenta and yellow denote detectors, though to make the diagram simpler we have not drawn the Pauli webs that define them (see Section C for a detector cheat sheet). On the right we show the neighbourhood of the yellow detector in the decoding graph, once all hyperedges have been decomposed into ordinary edges. This detector has degree 12, as does every other detector (apart from a few special cases right at the very start and end of the circuit). Crucially, this is also the case for the $X^1Z^1$ code – their decoding graphs after decomposition are isomorphic, differing only in the edge weights, i.e. in the probabilities of the corresponding errors occuring.

$e = d_1 \dots d_b$ with powers mod 2. As an example, in Figure 11 we show a measurement error in the $X^1Y^1Z^1$ honeycomb code. This violates four detectors, hence corresponds to a hyperedge spanning four vertices in the decoding graph[4].

Many of the best known decoding algorithms, however, can only be run on *ordinary* graphs – ones in which every hyperedge spans exactly 2 vertices. Decoders that require this form of graph are called *matching decoders*, and include *minimum-weight perfect matching* (MWPM) [1, 52, 53] and *union-find* [54, 55]. We can transform the decoding graph into an ordinary graph, at the cost of the edge weights becoming only approximations of the

[4]What we're calling a decoding graph is often called a *Tanner graph*, though Tanner graphs are usually defined as bipartite graphs, with one type of vertex representing detectors, another type representing errors, and an edge between two vertices exactly if the corresponding error violates the corresponding detector. The two definitions are equivalent. The term *detector error model* is also equivalently used [51].

corresponding errors occurring.

We start by 'decomposing' errors that violate more than 2 detectors into sets of errors that violate at most 2 detectors. For example, if the decoding graph contains a hyperedge $d_1d_2d_3d_4$ with weight $p$, but also a hyperedge $d_1d_2$ with weight $p_{12}$ and a hyperedge $d_3d_4$ with weight $p_{34}$, then we can delete the hyperedge $d_1d_2d_3d_4$ and update the weights $p_{12}$ and $p_{34}$ in some way. In fact, even if the graph only contains hyperedges $d_1d_2d_3d_4$ with weight $p$ and $d_1d_2$ with weight $p_{12}$, we can delete $d_1d_2d_3d_4$, update $p_{12}$ and add a new *remnant edge* $d_3d_4$ with some weight $p_{34}$. In this work, we use the Clifford circuit simulator Stim [49] extensively, and Stim's implementation of decomposing errors is described in Ref. [56] and explained in Ref. [51]. The way in which the edge weights are updated is explained in Appendix C of Ref. [57]. After decomposing all such hyperedges, we add a new virtual *boundary vertex* $\partial$, and we replace every hyperedge spanning a single vertex $d$ with a new ordinary edge with endpoints $d$ and $\partial$ [58]. At this point, every hyperedge will span exactly two vertices.

## 2.6 Logical errors

If a set of errors do not violate any detectors, but act non-trivially on the circuit implementing our code (i.e. the circuit with errors is not equal as a linear map to the circuit without errors, even up to global scalar) then this set constitutes a *logical error*. In a DCCC, we can further label logical errors as *spacelike* or *timelike*. Intuitively, spacelike logical errors are those consisting of sets of errors wrapping around the torus on which the code is defined (in either direction). Intuitively, timelike logical errors are those consisting of sets of errors that span from the start of the circuit to the end. For example, if after some timestep $t$ a set of qubit errors occur whose tensor product is a logical Pauli operator for the code at this timestep $t$ – like those depicted in Figure 7 – then this constitutes a spacelike logical error. In Figure 13, we show one spacelike and one timelike logical error in the $X^1Y^1Z^1$ honeycomb code.

In any particular circuit and noise model, the minimum number of errors required to create a logical error is defined to be the *fault distance* of the circuit. One can be more specific and speak of the *spacelike fault distance* or *timelike fault distance* – the minimum number of errors required to create a spacelike or timelike logical error, respectively.

## 2.7 Decoding

The decoding graph contains everything we need to be able to decode a quantum error-correcting code. In general, decoding works as follows: as we run the circuit implementing the quantum error-correcting code, we mark the vertices $d$ that correspond to violated detectors, then pass this marked graph to a decoder. It is the job of the decoder to find a set of hyperedges (i.e., errors) satisfying two properties: they could have caused these detectors to be violated, and the combination of these errors and the actual errors that occurred do not constitute a logical error. More precisely, let $e_1, \ldots, e_c$ be the actual errors that occurred, which necessarily have formal product $e_1 \ldots e_c$ equal to the syndrome $d_1 \ldots d_b$. The decoder's job is to find a set $e'_1, \ldots, e'_{c'}$ of hyperedges such that the formal product $e'_1 \ldots e'_{c'}$ is again $d_1 \ldots d_b$, and the set of errors $\{e_1, \ldots, e_c, e'_1, \ldots, e'_{c'}\}$ is not a logical error.

In this work, we consider two decoders: *minimum-weight perfect matching (MWPM)* and *belief matching* [57]. These are both matching decoders, so require the decoding graph

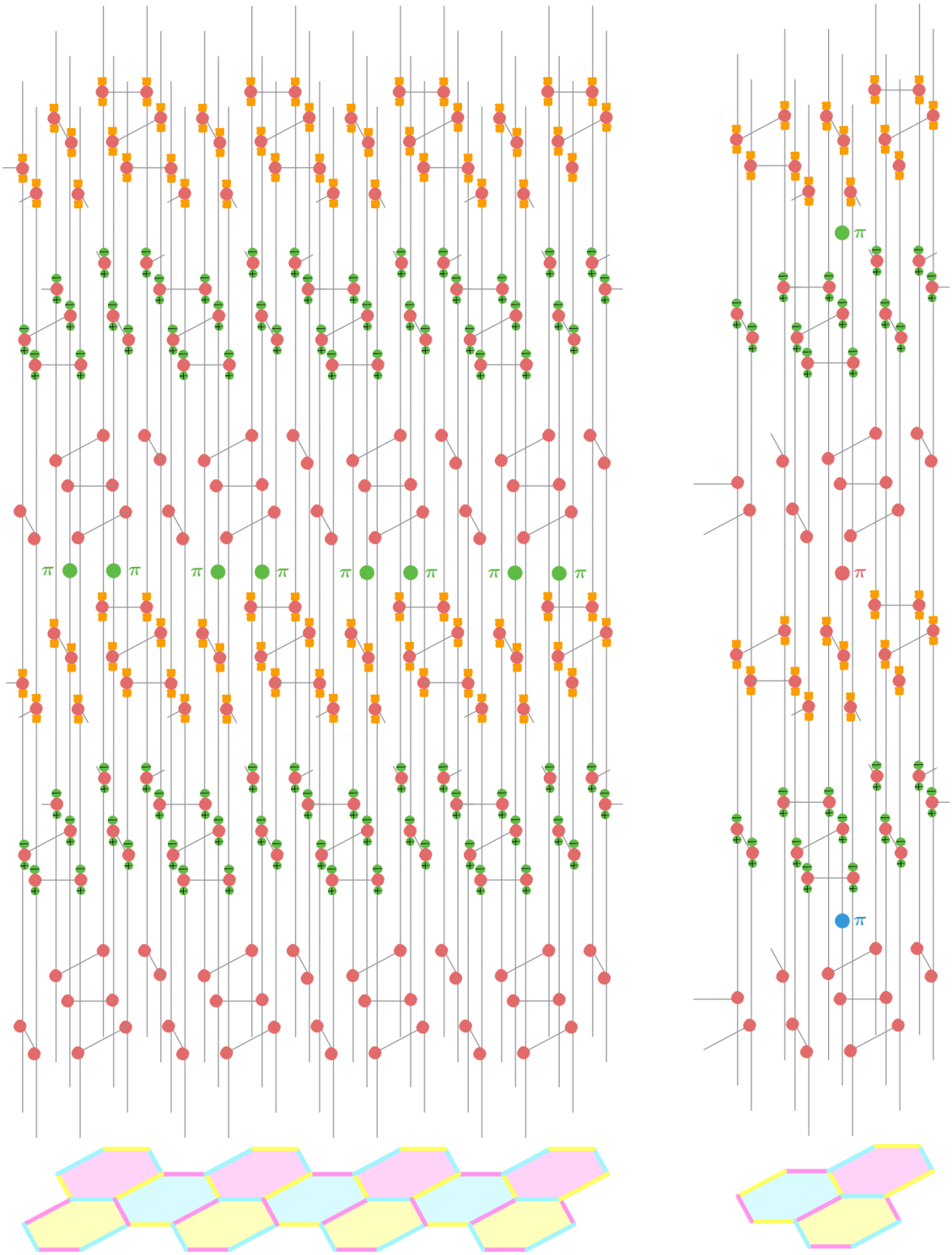



**Figure 13:** On a subset of qubits, we show a spacelike (left) and timelike (right) logical error in the XYZ honeycomb code consisting only of data qubit errors. Click here to view the spacelike and here to view the timelike logical errors in Crumble [59], a prototype tool for Clifford circuit exploration.

to be an ordinary graph (or decomposable into an ordinary graph). In MWPM, we simply apply the transformation described in Subsection 2.5 to the decoding graph (i.e. we convert the hypergraph into a graph). Then, on any given run of the circuit, we apply the MWPM decoder to the resulting marked ordinary graph [58]. In belief matching, we add a pre-processing step which first adjusts the weights of the hyperedges in the marked decoding graph. This pre-processing step is the classical decoding algorithm *belief propagation* (BP) [60–62], which is run on the full decoding hypergraph to estimate the marginal probability of each error mechanism (whether it is an edge or hyperedge). These marginal probabilities are then used to update the edge weights when converting to a graph for MWPM [57]. The details of exactly what this does are unimportant for us, but the key intuition is that it recovers some of the information about the probabilities of errors occurring that's otherwise lost when transforming hyperedges into ordinary edges (i.e. it exploits knowledge of the hyperedge errors when decoding a given syndrome). This allows for more accurate decoding for error models containing hyperedges, at the cost of slowing down the overall decoding process. For those interested, a longer discussion of this trade-off can be found in Ref. [57]. After running BP, we apply the transformation described in Subsection 2.5 to the marked decoding graph, and apply MWPM to the resulting marked ordinary graph. A key line of investigation in this work is understanding for which DCCCs and noise models using belief matching brings most benefit.

## 2.8 $E$ and $M$ components

In any DCCC, after applying the transformation described in Subsection 2.5 to the decoding graph, the new ordinary graph $G$ has the property that $G - \{\partial\}$ consists of two disconnected components, where $\partial$ is the boundary vertex we added. We will call one of these components the $E$ (electric) component and the other the $M$ (magnetic) component. Then many of our other objects of interest inherit a label. Detectors corresponding to vertices in the $E$ or $M$ component are labelled $E$ or $M$ detectors respectively. Errors whose corresponding hyperedge in the original decoding graph spans vertices entirely in the $E$ or $M$ component of $G - \{\partial\}$ are labelled $E$ or $M$ errors, respectively. Errors whose corresponding hyperedge spans vertices in both the $E$ and $M$ components are labelled $F$ (fermion) errors[5]. Spacelike logical errors that – intuitively speaking – wrap around the $E$ or $M$ component are labelled spacelike logical $E$ or $M$ errors respectively. Timelike logical errors likewise inherit an $E$ or $M$ label. Finally, the fault distances can be further divided into a spacelike $E$ and $M$ fault distance – the minimum number of errors needed to create a spacelike logical $E$ or $M$ error, respectively – and a timelike $E$ and $M$ fault distance – the minimum number of errors needed to create a timelike logical $E$ or $M$ error, respectively.

## 2.9 Boundaries

As stated throughout this section, in this work we only consider DCCCs on a torus. Primarily, this is to make comparisons between different measurement schedules fair, since there are no boundaries at which they might behave differently. Helpfully, it also makes our analysis a great deal simpler. On the other hand, if one is ultimately interested in

[5]This $E$, $M$ and $F$ terminology is borrowed from the condensed matter approach, as in Ref. [13]. We could equally have used the more standard $X$, $Z$ and $Y$ respectively, but we felt this could ultimately cause more confusion – e.g. in the $X^1Y^1Z^1$ honeycomb code, some physical Pauli $X$ errors correspond to $E$ errors, some to $M$ errors, and some to $F$ errors. We felt it might be confusing to say that a Pauli $X$ error was actually a $Z$ error or a $Y$ error – indeed, we confused ourselves many times in this project with this sort of terminology.

planar codes (e.g. because of hardware constraints) one might very reasonably question to what extent a code's performance in the toric setting is a good proxy for its performance in the planar setting.

First, we note that a general theory of boundaries for DCCCs is given in Ref. [13, Section 7.3], so every DCCC can indeed be made planar. Furthermore, for the $X^1Y^1Z^1$ and $X^1Z^1$ codes that we focus on in this work, specific boundaries are defined in Ref. [29] and Ref. [13] respectively. In the former work, it's shown numerically that the planar $X^1Y^1Z^1$ code performs essentially as well as the toric version. In the latter work, no numerical results were given, but the detectors at the boundary were fully worked out and depicted. Since these follow exactly the same pattern as the detectors in the bulk, we again anticipate that performance of this planar $X^1Z^1$ code would essentially mirror that of its toric counterpart, but have not verified this.

## 3 Teraquop volume

Before we start looking at particular DCCCs in detail, we first introduce the key metric we use to evaluate performance throughout this work. It is the natural generalisation of the *teraquop footprint* [28] from a 'space only' picture to a 'full spacetime' picture. We call it the *teraquop volume*.

**Definition 3.1.** *Consider a family of circuits implementing a quantum error correcting protocol, whose $j$-th member is defined on a number of qubits $n_j$ and runs for an amount of time $T_j$. Given a noise model and a decoder, the* ***teraquop volume*** *is the smallest volume $n_jT_j$ such that the probability of any spacelike and timelike logical error in the $j$-th circuit is $10^{-12}$.*

Here we deliberately do not specify what exactly a quantum error correcting protocol is and what it means for a set of circuits implementing one to form a family. For a given circuit, noise model and decoder, the *overall logical error rate* is the probability of any logical error – whether spacelike or timelike – occurring after the decoder has been applied. In practice, an approximation of this value will be found numerically. If there is no $(n_j, T_j)$ such that the overall logical error rate is less than $10^{-12}$, then we say the teraquop volume is infinite. For a given code family, noise model, and physical error rate, the teraquop volume typically corresponds to a unique point $(n_j, T_j)$ rather than a smooth curve of equivalent space-time trade-offs. That is, one cannot generally reduce the number of qubits and compensate by increasing the number of rounds (or vice versa) while maintaining the target logical error rate.

The teraquop volume is the appropriate metric for evaluating quantum error correction performance in lattice surgery computations because it accounts for space-time tradeoffs. In surface code architectures, circuit depth can be reduced at the cost of increased qubit overhead, or vice versa [30]. A volume-based metric captures this interchangeability, making it possible to compare codes with different spatial and temporal resource requirements on equal footing. If the teraquop footprint were used instead, codes with high qubit overhead but low circuit depth would be unfairly penalized, and vice versa.

In the next subsection we explain how we determine the teraquop volume of a DCCC. Figure 14 summarizes the concept of teraquop volume and the methodology used for its determination.

### 3.1 Estimating the teraquop volume

Let $n_M$ and $n_E$ denote (respectively) the number of rows and columns of hexagons in a DCCC. For example, in Figures 5 and 7 we showed DCCCs with $n_M = 3$ (admittedly quite wonky) rows of hexagons, and $n_E = 4$ columns. The value of $n_M$ determines the spacelike $M$ distance $d_{s,M}$ – namely $d_{s,M} = \frac{4}{3}n_M$. Similarly $n_E$ determines the spacelike $E$ distance as $d_{s,E} = n_E$.

The total number of data qubits (i.e., vertices) in a DCCC with shape $(n_M, n_E)$ is $2n_M n_E$. The total number of auxiliary qubits varies depending on the circuits used to perform the measurements defining the code. For example, on a heavy-hex lattice where an auxiliary qubit is placed on each edge and measurements are performed via controlled gates and single-qubit measurements, the number of auxiliary qubits is $3n_M n_E$, for a total of $5n_M n_E$ qubits. If measurements are performed via two-qubit measurements directly, then no auxiliary qubits are needed, so the total number of qubits is just $2n_M n_E$. For a given family of circuits, we denote the total number of qubits (the *footprint*) by $f(n_M n_E)$.

Given such a family, suppose we perform $h$ rounds of measurements. In contrast to $n_M$ and $n_E$, the value of $h$ simultaneously determines both the timelike $M$ distance and the timelike $E$ distance, denoted $d_{t,M}$ and $d_{t,E}$ respectively. Furthermore, these timelike distances depend on the specific choice of DCCC, as we will see shortly in Section 4. We say the total *volume* of this circuit is $f(n_M n_E)h$. Determining the teraquop volume requires finding the minimal value of $f(n_M n_E)h$ such that the probability of any logical error occurring is $10^{-12}$. Although this suggests a large search space, the minimal values of $n_E$, $n_M$, and $h$ can be evaluated independently to obtain a good approximation. This is because the probability of each type of logical error occurring (spacelike $E$ or $M$, or timelike $E$ or $M$) depends exponentially on one of $n_E$, $n_M$, and $h$, and linearly on the other two. For example, the probability of a spacelike logical $E$ error occurring depends exponentially on $n_E$ and linearly on $n_M$ and $h$. When the minimum value of all three independently-determined dimensions are used together, each type of logical error remains well-suppressed because the exponential dependence dominates the linear scaling. The overall logical error rate approximates $10^{-12}$, though it may be somewhat below or above this threshold by a small constant factor.

Plots verifying this linear dependence are shown in Figure 16. It could be the case that for different codes the probability of each type of logical error occurring depends exponentially on one parameter and non-linearly on the other two, but we have not observed this in any of the DCCCs we have studied. If this were to occur, we expect that the approximation method we use would still give a reasonable estimate of the teraquop volume. To verify if our approximation method is accurate, one can simulate the overall logical error rate for the estimated minimal values of $n_E$, $n_M$, and $h$ and check if it is indeed below $10^{-12}$.

To actually find these minimal values of $n_E$, $n_M$, and $h$, we perform memory and stability experiments [64]. A memory experiment means initialising a logical qubit in an eigenstate of a certain logical operator, performing the logical identity gate for some amount of time, then measuring this logical operator. In the absence of noise, this measurement outcome will be deterministic, but the occurrence of certain logical errors can cause the decoder to fail and return the wrong measurement outcome. The frequency with which these failures occur is the logical error rate for this experiment. A stability experiment instead benchmarks how reliably the product of a large region of stabilizers can be measured [64].

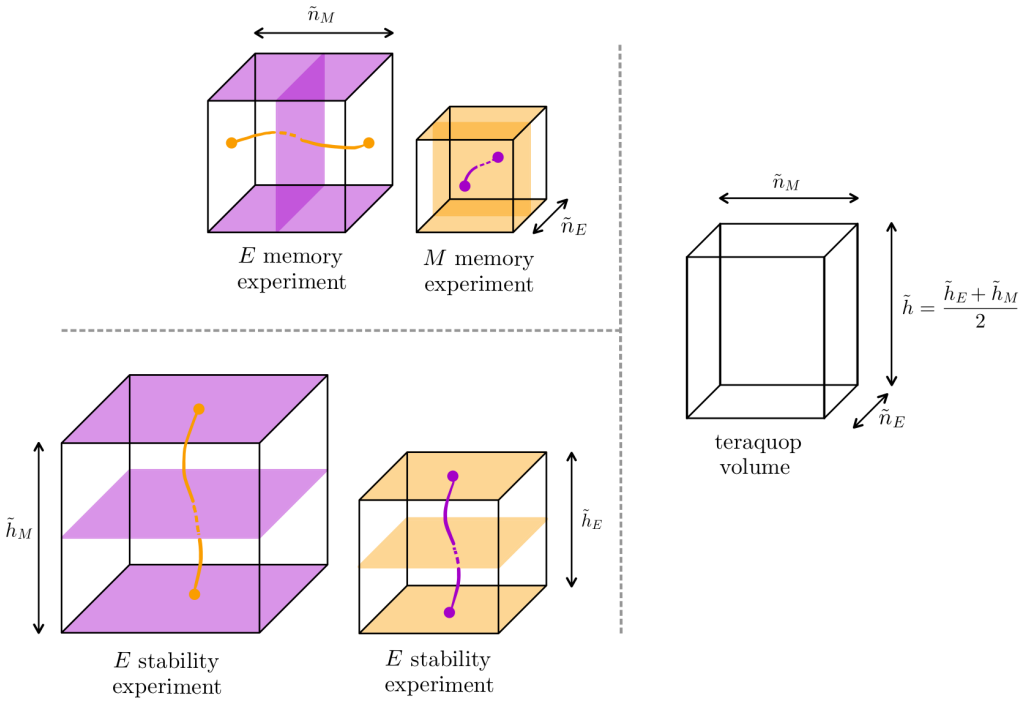



**Figure 14:** Schematic explaining how the teraquop volume of a DCCC is estimated. We perform $E$-memory experiments to estimate the number of columns $\tilde{n}_M$ of hexagons in any DCCC that reaches the teraquop regime. Then $M$-memory experiments estimate the number of rows $\tilde{n}_E$ required, and $E$- and $M$-stability experiments estimate the number of measurement rounds $\tilde{h}$ required.

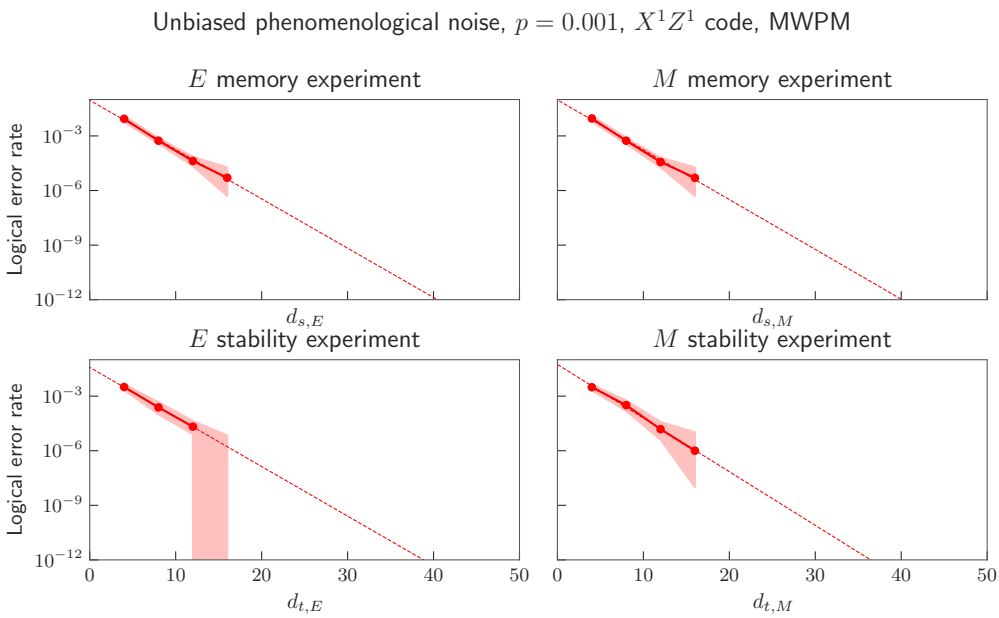



**Figure 15:** Plots showing line-fits to determine the minimal value of $n_E$, $n_M$ $h_E$, and $h_M$ such that the probability of any logical error occurring is $10^{-12}$. The value of the $x$-axis wherever a dotted line would first cross the line $y = 10^{-12}$ is the value used for the corresponding teraquop dimension. The highlighted regions are computed with Sinter and show hypothesis probabilities within 1000× of the max likelihood hypothesis [63].

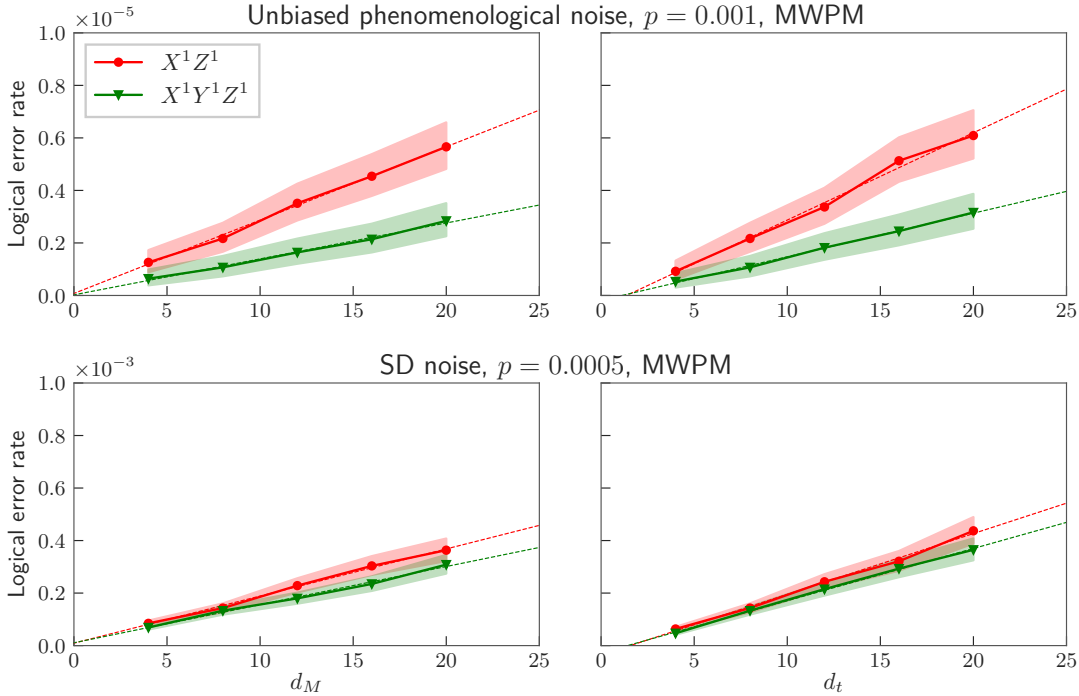



**Figure 16:** Plots showing the logical error rate of $E$ memory experiments with varying $n_M$ and $h$. Decoding is performed with MWPM. The plots confirm that the probability of a spacelike logical $E$ error occurring depends linearly on $n_M$ and $h$. The highlighted regions are computed with Sinter and show hypothesis probabilities within 1000× of the max likelihood hypothesis [63].

This tests a part of a lattice surgery operation (where the product of stabilizers measured corresponds to a logical Pauli product), or a part of an operation to move a logical operator (where the region of stabilizers is multiplied into the logical operator to move it). Whereas a memory experiment benchmarks how well a logical observable is preserved through time, a stability experiment tests how well it is preserved through space. Both experiments test important building blocks of a lattice surgery quantum computer.

For example, in a DCCC, an $E$ memory experiment is one in which our logical qubit is initialised in the $+1$ eigenstate of the $E$ logical operator. Its failure rate tells us how often spacelike $M$ logical errors occur, and depends exponentially on $n_M$ but only linearly on $n_E$ and $h$. So for variable $d$, we can perform $E$ memory experiments with $n_M$, $n_E$ and $h$ chosen such that both spacelike distances are $d$ (i.e., $d_{s,E}(n_E) = d$ and $d_{s,M}(n_M) = d$) and both timelike distances are at least $d$ (i.e., $d_{t,E}(h) \geq d$ and $d_{t,M}(h) \geq d$). For each $d$ we obtain a logical error rate. The minimum $n_M(d)$ at which this logical error rate goes under $10^{-12}$ is the value of $n_M$ we will use in our teraquop volume estimation. We fit a line to the natural log of the logical error rate versus $d$ and extrapolate this line to find the required $n_M(d)$. Figure 15 contains examples of plots showing the described line fits.

We repeat the entire procedure with the roles of $E$ and $M$ swapped to find the value of $n_E$ to use in our teraquop volume estimation. To find the value of $h$ we use, we do essentially the same thing but with stability experiments. An $E$ stability experiment tests for protection against timelike $M$ errors, and vice versa for an $M$ stability experiment. Plotting the logical error rates and extrapolating gives us an estimate for the numbers of rounds of measurements $h_E$ and $h_M$ needed to ensure timelike logical $E$ and $M$ errors occur with probability less than $10^{-12}$. We then just choose the value $h$ that we use in our

teraquop volume estimation to be the mean $(h_E + h_M)/2$. By doing this we assume that places where timelike logical $E$ and $M$ errors can occur are roughly equally prevalent in a lattice surgery computation. Additionally, we assume that logical qubits do not need to be synchronized, meaning that lattice surgery measurements in which timelike logical $E$ errors can occur can have a different duration to ones in which timelike logical $M$ errors can occur.

### 3.2 Blocks

In the rest of this work we describe a DCCC with $n_M$ rows and $n_E$ columns of hexagons implemented for $h$ rounds of measurements as an $(n_M, n_E, h)$-*block* of this DCCC. We say it has *block height* $h$, *block width* $n_E$ and *block depth* $n_M$. Then we'll write $\tilde{n}_M$, $\tilde{n}_E$, $\tilde{h}_M$, and $\tilde{h}_E$ for the minimum values of $n_M$, $n_E$, $h_M$, and $h_E$ such that the corresponding logical error rates are below $10^{-12}$. An $(\tilde{n}_M, \tilde{n}_E, \tilde{h})$-block will be referred to as a *teraquop block*, with *teraquop height* $\tilde{h} = (\tilde{h}_E + \tilde{h}_M)/2$, and so on.

### 3.3 Limitations

The teraquop volume is a measure of the spacetime cost of the basic unit of a lattice surgery quantum computer. As with all single-number metrics, it has its drawbacks. One drawback is that equal cost is attributed to space and time. For example, we might have two (spacetime) blocks of code with equal volume but different dimensions – one tall and narrow, one short and wide. In a setting where qubits are limited but time is plentiful, the tall and narrow block would be the better choice. In contrast, if qubits are plentiful but time is limited, the short and wide block should be chosen. This important difference between the two blocks disappears when just looking at the teraquop volumes.

A second drawback is that the teraquop volume measure does not account for optimizations achieved by altering the measurement schedule and patch size during a computation. Consider a circuit component in which logical qubits idle for a long time. In this scenario, there are no timelike errors to worry about. Therefore, the logical qubits should be encoded in a circuit with low teraquop *footprint*, and not necessarily low teraquop *volume*.

## 4 Bias-tailoring I: choosing a schedule

In this section we discuss differences between the $X^1Z^1$ and $X^1Y^1Z^1$ codes and their impact on the teraquop volumes. We begin by considering the simpler case of direct parity measurement noise – the simplicity here allows us to look closely at specific error mechanisms in order to better understand the two codes – before we move to the more realistic but messier case of auxiliary qubit circuit noise.

### 4.1 Differences

We first outline the major differences between the two codes, and the expected effects on the teraquop volumes. Some of these effects are in conflict with one another – we will see numerically in the rest of this section which features ultimately dominate performance.

**Difference 4.1.** *Detectors in the $X^1Y^1Z^1$ code consist of 12 measurements, while detectors in the $X^1Z^1$ code consist of 6 measurements (Subsection 2.4, Figure 10).*

The intuition is that the fewer measurements involved in a detector, the better. This is because the more measurements that are involved, the more likely the detector is to

be violated, and more violated detectors means more marked vertices in the decoding graph, which makes the job of the decoder harder[6]. For example, MWPM has average-case runtime that's approximately linear in the number of violated detectors, provided the probabilities of qubit and measurement errors occurring are not too high [58]. Indeed, this is the intuition as to why the Bacon-Shor code has no threshold [66]. So this difference between the two codes would suggest the $X^1Z^1$ code might generally perform better, and therefore have a better teraquop volume – especially for measurement-biased noise.

**Difference 4.2.** *The $X^1Y^1Z^1$ code implemented for $h$ rounds has a timelike distance of $h/2$, while the $X^1Z^1$ code implemented for $h$ rounds has a timelike distance of $h/4$ (Table 3, Figure 26).*

In contrast to Difference 4.1 above, this difference favours the $X^1Y^1Z^1$ code; based on this alone, one would expect the $X^1Y^1Z^1$ code to perform better than the $X^1Z^1$ code in stability experiments, i.e. have a smaller teraquop height $\tilde{h}$. In particular, since the weight-$h/4$ timelike logical errors in the $X^1Z^1$ code consist entirely of measurement errors, one would expect this to be particularly relevant for measurement-biased noise.

**Difference 4.3.** *All measurement errors in the $X^1Y^1Z^1$ code correspond to hyperedges in the decoding graph, while no measurement errors in the $X^1Z^1$ code correspond to hyperedges (Table 3).*

Recall that hyperedges are decomposed into ordinary edges in order to use a matching decoder like MWPM, at the cost of losing some information about the likelihood of these errors occurring (Subsection 2.5). But belief matching recovers much of this information (Subsection 2.7). Therefore belief matching performs best on codes where many errors correspond to hyperedges. Accordingly, this difference suggests the $X^1Y^1Z^1$ code may perform worse than the $X^1Z^1$ code under MWPM, but improve a lot under belief matching.

**Difference 4.4.** *In the $X^1Z^1$ code, $E$ logical errors that consist of Pauli errors can only consist of $Z$ or $Y$ errors, whereas $M$ logical errors consisting of Pauli errors can only consist of $X$ or $Y$ errors. In the $X^1Y^1Z^1$ code, both types of logical error can be formed from all types of Pauli errors.*

This would seem to suggest that the $X^1Z^1$ code can be better tailored to $Z$-biased noise in memory experiments, for the following reason. Recall that we are considering our toric DCCCs to implement just one logical qubit, rather than two, for simplicity and for better comparison with the more widely implementable planar case (Subsection 2.9). In the $X^1Z^1$ code under a $Z$-biased noise model, spacelike $E$ logical errors become more likely to occur, and spacelike $M$ errors become less likely to occur. But we can in effect cancel this out by changing the code dimensions. Since the likelihood of spacelike $E$ logical errors occurring depends exponentially on the number of columns $n_E$ of hexagons in the code, and only linearly on the number of rows $n_M$ and the number of measurement rounds $h$, we can reduce the chances of such errors occurring by increasing $n_E$. Similarly, if we decrease $n_M$ we increase the chances of spacelike $E$ errors occurring. Altogether one might think we should be able to define a rectangular code with $n_E > n_M$, such that we save qubits overall while maintaining the same protection against spacelike logical errors. In the $X^1Y^1Z^1$ code this is not possible, because the noise being biased towards Pauli $Z$ errors doesn't increase

[6] A more accurate metric capturing this intuition is that of *detector likelihood* [65]. This is the total probability of a set of errors occurring that violate a given detector. In the comparison between the $X^1Y^1Z^1$ and $X^1Z^1$ codes, measurement errors are the only errors that contribute differently to the detector likelihood – all data qubit errors affect the detector likelihood in the same way across the two codes.

or decrease the probability of either type of spacelike logical error occurring.

## 4.2 Direct parity measurement noise results

Now we look at our simulation results. Figure 17 shows $\tilde{n}_E, \tilde{n}_M, \tilde{h}_E, \tilde{h}_M$ and the teraquop volume of the $X^1Y^1Z^1$ and $X^1Z^1$ codes for unbiased, $Z$-biased, and measurement-biased direct parity measurement noise. We highlight two key points, and examine their relationship to the differences from the last subsection.

**Key point 4.5.** *In all three noise models, the $X^1Y^1Z^1$ code under belief matching has the smallest teraquop volume, while the $X^1Y^1Z^1$ code under MWPM has the largest teraquop volume.*

This is exactly what was predicted by Difference 4.3 – all measurement errors in the $X^1Y^1Z^1$ code correspond to hyperedges in the decoding graph, which makes performance under MWPM worse. But belief matching turns this bug into a feature. Moreover, the effect becomes more dramatic the more the noise model is biased towards measurement noise.

This both agrees and disagrees with the intuition mentioned after Difference 4.1 – that the detectors in the $X^1Y^1Z^1$ consisting of twice as many measurements as in the $X^1Z^1$ code would make the former perform worse, especially under measurement-biased noise. While this is true for MWPM, using belief matching suffices to counteract this effect. Indeed, the high number of measurements per detector and the fact all measurement errors violate four detectors are two sides of the same coin – the latter being exactly what makes belief matching perform so well. Similarly, the better timelike distance in the $X^1Y^1Z^1$ code (Difference 4.2) doesn't seem to play a role under MWPM, but is consistent with performance under belief matching.

| | $X^1Y^1Z^1$ | | | $X^1Z^1$ | | |
|---|---|---|---|---|---|---|
| | Direct parity meas. | Meas. noise | Z noise | Direct parity meas. | Meas. noise | Z noise |
| $d_{s,E}$ | $n_E$ | $n_E$ | $n_E$ | $n_E$ | $n_E$ | $n_E$ |
| $d_{s,M}$ | $\frac{4}{3}n_M$ | $\frac{4}{3}n_M$ | $\frac{4}{3}n_M$ | $\frac{4}{3}n_M$ | $\frac{4}{3}n_M$ | $\infty$ |
| $d_{t,E}$ | $h/2$ | $h/2$ | $h/2$ | $h/4$ | $h/4$ | $h/2$ |
| $d_{t,M}$ | $h/2$ | $h/2$ | $h/2$ | $h/4$ | $h/4$ | $\infty$ |
| $p_h$ | 5/9 | 1 | 1/3 | 2/9 | 0 | 0 |

**Table 3:** Differences in distances between the $X^1Y^1Z^1$ and $X^1Z^1$ codes. The first four rows show graphlike distances of spacelike and timelike logical operators. The bottom row shows the *hyperedge likelihood* – the probability that a randomly chosen error in the noise model will correspond to a hyperedge in the decoding graph. Of note are that the $X^1Z^1$ code has worse timelike distance when measurement errors are possible, but infinite spacelike and timelike $M$ distances when only $Z$ errors are possible. The $X^1Y^1Z^1$ code has a higher hyperedge likelihood in all cases, explaining why it performs the worst under MWPM but the best under belief matching (Key point 4.5).

**Key point 4.6.** *The differences in teraquop volume between the $X^1Y^1Z^1$ and the $X^1Z^1$*

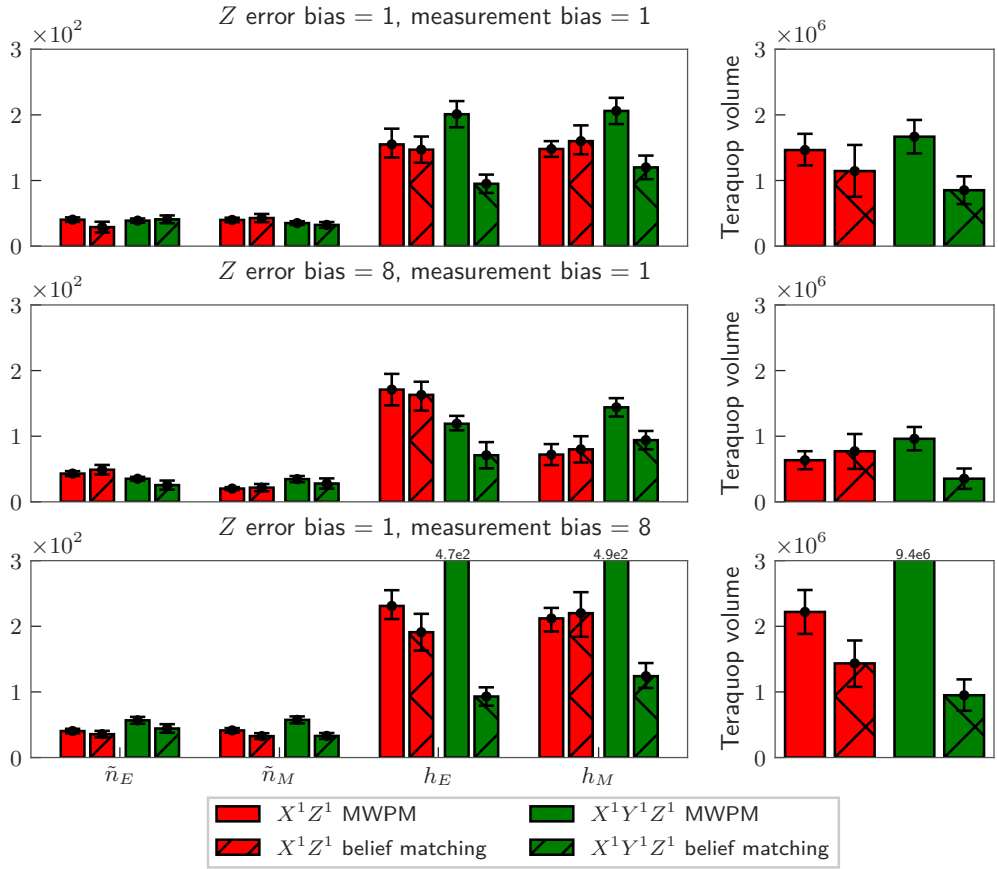



**Figure 17:** Comparison of $\tilde{n}_E, \tilde{n}_M, \tilde{h}_E, \tilde{h}_M$, and teraquop volume of the $X^1Y^1Z^1$ and $X^1Z^1$ code decoded using MWPM and belief matching for unbiased, $Z$-biased, and measurement-biased phenomenological noise. In all cases, using belief matching rather than MWPM turns the $X^1Y^1Z^1$ code from the worst performer to the best performer (Key point 4.5). The difference in teraquop volumes comes primarily from the differences in teraquop height $\tilde{h} = (\tilde{h}_E + \tilde{h}_M)/2$ – these dwarf the differences in teraquop footprints (Key point 4.6). This is thus an advert for analysing codes by looking at teraquop volumes rather than just teraquop footprints.

| | $X^1Y^1Z^1$ | | $X^1Z^1$ | |
|---|---|---|---|---|
| | SI & SD noise | EM noise | SI & SD noise | EM noise |
| $d_{s,E}$ | $n_E$ | $n_E/2$ | $n_E$ | $n_E/2$ |
| $d_{s,M}$ | $\frac{4}{3}n_M$ | $\frac{2}{3}n_M$ | $\frac{4}{3}n_M$ | $\frac{2}{3}n_M$ |
| $d_{t,E}$ | $h/4$ | $h/2$ | $h/4$ | $h/4$ |
| $d_{t,M}$ | $h/4$ | $h/2$ | $h/4$ | $h/4$ |

**Table 4:** Distances of the $X^1Y^1Z^1$ and $X^1Z^1$ codes. Note that the entangling measurements noise model halves the spacelike distances of both codes. Meanwhile the SI and SD noise models halve the timelike distances of the $X^1Y^1Z^1$ code, removing an advantage they had over the $X^1Z^1$ code from Table 3. These distances were obtained using Stim's function `shortest_graphlike_error`. For distance 4 circuits we found the exact same distances when using the function `search_for_undetectable_logical_errors`. This second function takes errors that correspond to hyperedges into account.

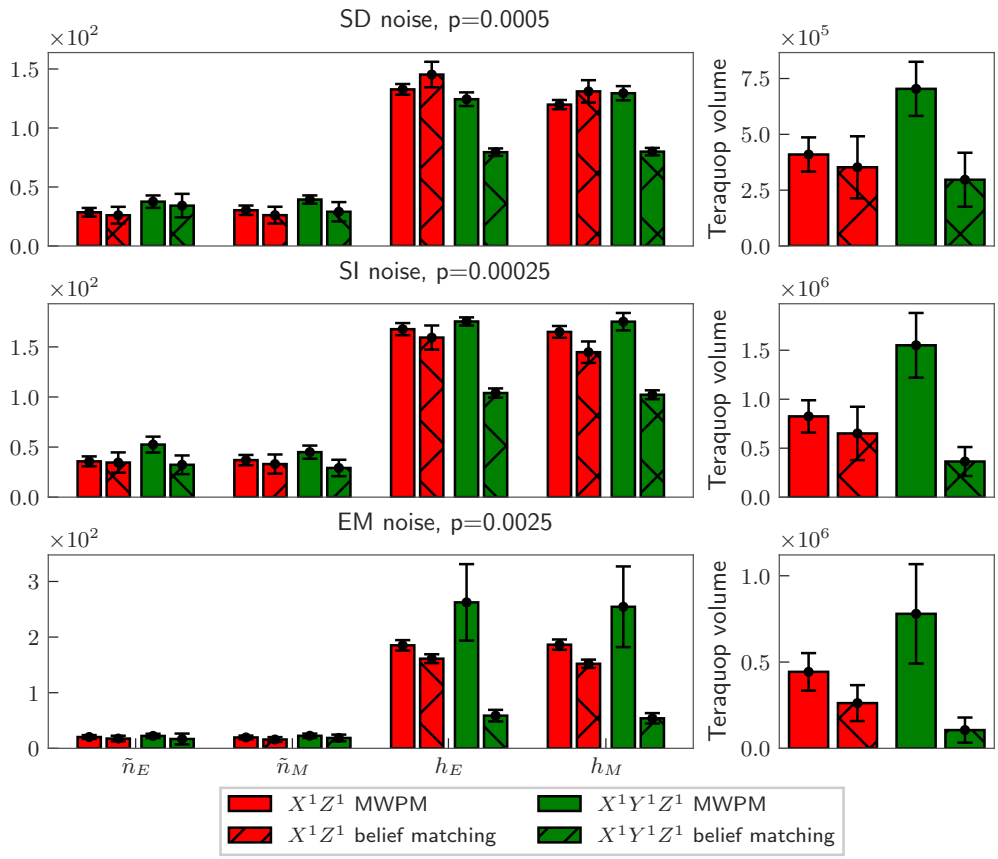



**Figure 18:** Comparison of $\tilde{n}_E, \tilde{n}_M, \tilde{h}_E, \tilde{h}_M$, and teraquop volume of the $X^1Y^1Z^1$ and $X^1Z^1$ code decoded using MWPM and belief matching for three circuit level noise models; standard depolarizing noise with $p = 0.0005$, superconducting inspired noise with $p = 0.00025$, and entangling measurement noise with $p = 0.0025$. As in Figure 17, in all cases, using belief matching rather than MWPM turns the $X^1Y^1Z^1$ code from the worst performer to the best performer (Key point 4.5), and the most dramatic differences are to be found in the teraquop heights rather than footprints (Key point 4.6) – especially for the entangling measurements noise model.

*codes are primarily due to differences in the number of rounds of measurements required to reach the teraquop regime – not in the number of physical qubits (footprint).*

That is, the differences in footprints $\tilde{n}_M\tilde{n}_E$ across codes and noise models are generally far smaller than the difference in the teraquop heights $\tilde{h}$. This contradicts the intuition from Difference 4.4 – we wondered whether the teraquop width $\tilde{n}_E$ and depth $\tilde{n}_M$ of the $X^1Z^1$ code under $Z$-biased noise would be such that the footprint $f(\tilde{n}_E\tilde{n}_M)$ would be smaller than that of the $X^1Y^1Z^1$ code under this noise model. Though $\tilde{n}_M$ is indeed smaller than that of the $X^1Y^1Z^1$ code, $\tilde{n}_E$ is larger. That said, it could be that for a higher $Z$ noise bias than we simulated there could be a significant difference between $\tilde{n}_E\tilde{n}_M$ for the $X^1Z^1$ and $X^1Y^1Z^1$ codes.

A meta-point to make in light of Key point 4.6 above is that using the teraquop volume rather than just the teraquop footprint was important. Had we only used the teraquop footprint, we would not have seen the dramatic differences between the two codes and the two decoders, because these differences almost entirely show up in the teraquop heights rather than the footprints. We claim this serves as evidence in favour of using teraquop volumes in future error correction simulations.

In Table 3, we summarise some important metrics. We show the four types of distances of the two codes under the three different noise models, and additionally show the *hyperedge likelihood* $p_h$ – the likelihood of an error in that noise model corresponding to a hyperedge in the decoding graph.

### 4.3 Auxiliary qubit circuit noise results

Figure 18 presents $\tilde{n}_E, \tilde{n}_M, \tilde{h}_E, \tilde{h}_M$, and the teraquop volume for the $X^1Y^1Z^1$ and $X^1Z^1$ codes under three auxiliary qubit circuit noise models. Logical operator distances are summarized in Table 3. The results confirm the trends identified in Key point 4.5: the $X^1Z^1$ code performs best with MWPM, whereas the $X^1Y^1Z^1$ code is superior with belief matching.

Similarly, Key point 4.6 holds, as differences in teraquop volume primarily stem from variations in the number of measurement rounds. For SI and SD noise decoded with belief matching, $\tilde{h}_E$ and $\tilde{h}_M$ are reduced by a factor of two for the $X^1Y^1Z^1$ code. Surprisingly, the timelike distance remains identical between the two codes, suggesting that distance alone is not always a reliable performance indicator. A similar case was made in Ref. [40].

Despite simulating entangling measurement noise with a physical error rate ten times higher, the teraquop volumes remain of the same order of magnitude across all three auxiliary qubit circuit noise models.

## 5 Bias-tailoring II: repeating measurements

There is another technique we can use to tailor our code to certain noise biases, introduced in Ref. [18], where it was referred to as *schedule-induced gauge-fixing* (SIGF) and applied to *subsystem* codes. In both subsystem codes and DCCCs, the check operators measured do not all mutually commute (these check operators are known as *gauge operators* in a subsystem code or edge operators in DCCCs). As a result, using a conventional measurement schedule, the measurement of each check operator is non-deterministic when considered in isolation, and must be combined with other (different) check operators to form a detector.

The SIGF method consists of *repeating* the measurements of a mutually commuting subset of the check operators. When a subset of mutually commuting gauge operators is measured, each check operator becomes an element of the ISG. When the measurements of these check operators are repeated, the subsequent measurements are all deterministic (in the absence of noise), and so we can construct much smaller detectors formed from consecutive measurements of individual check operators. Intuitively, by remaining in the same ISG while we repeat the commuting measurements, we can form much smaller detectors consisting of far fewer measurements, allowing us to protect against certain types of errors more effectively. For example, if we remain in an ISG with detectors that protect more effectively against $Z$ errors, then this can be effective for handling $Z$-biased noise models (as was done in Ref. [18]).

SIGF has been applied to subsystem codes Ref. [18] and to the Floquet-Bacon-Shor codes Ref. [67]. However, it has not previously been studied how well it can handle a noise model biased towards *measurement* errors. Intuitively we would expect SIGF to be advantageous in this setting, since each detector is formed from far fewer of the noisy measurements. In this work, we apply SIGF to DCCCs. Concretely, we simply repeat the edge operator measurements made at each timestep $t$ some number of times $r_t$. We call the resulting new dynamical code a *repeated DCCC* – or, if we want to be specific, an $\mathbf{r}$-*repeated DCCC*, where $\mathbf{r} = (r_0, r_1, \ldots)$.

In the next subsection we look analytically at how this intuition manifests itself in terms of detectors, logical errors and the decoding graph, and what we expect this to mean for the teraquop volumes. Then we present numerical results and see how well our expectations hold up – again, first for the more tractable case of direct parity measurement noise, and then the more realistic auxiliary qubit circuit noise. Furthermore, we explore how repeated DCCCs perform both for $Z$ error bias and measurement bias.

## 5.1 Detectors

Suppose a (unrepeated) DCCC has a detector

$$\prod_{(t\in\tau)} \prod_{(e_{\kappa_t}\in f_\kappa)} m_t(e_{\kappa_t,P_t}), \tag{6}$$

where $f_\kappa$ is the $\kappa$-coloured face to which the detector is associated, and $\tau$ is a set of timesteps. See, for example, the leftmost diagram of Figure 19. Now let $r_t \geq 1$ denote the number of times each measurement at timestep $t$ in the unrepeated DCCC is repeated before moving to the next measurement type, let $R_t = \sum_{s=0}^{t-1} r_s$ be the total number of measurement rounds up to timestep $t$ in the unrepeated DCCC, and for all $t \in \tau$ let $u_t$ be some integer in $\{0, \ldots, r_t - 1\}$. In the corresponding repeated DCCC, for each $e_{\kappa_t,P_t}$, one can show that using the measurement outcomes $m_{R_t+u_t}(e_{\kappa_t,P_t})$ from any timestep $R_t + u_t$ of each set of $r_t$ repeated measurement rounds gives a detector

$$\prod_{(t\in\tau)} \prod_{(e_{\kappa_t}\in f_\kappa)} m_{R_t+u_t}(e_{\kappa_t,P_t}). \tag{7}$$

We show an example of such a detector in the middle diagram of Figure 19, with $r_t = 2$ for all $t$. However, some choices of $u_t$ are more sensible than others. Firstly, we want the detector to span as few timesteps as possible; this minimises the probability of the detector

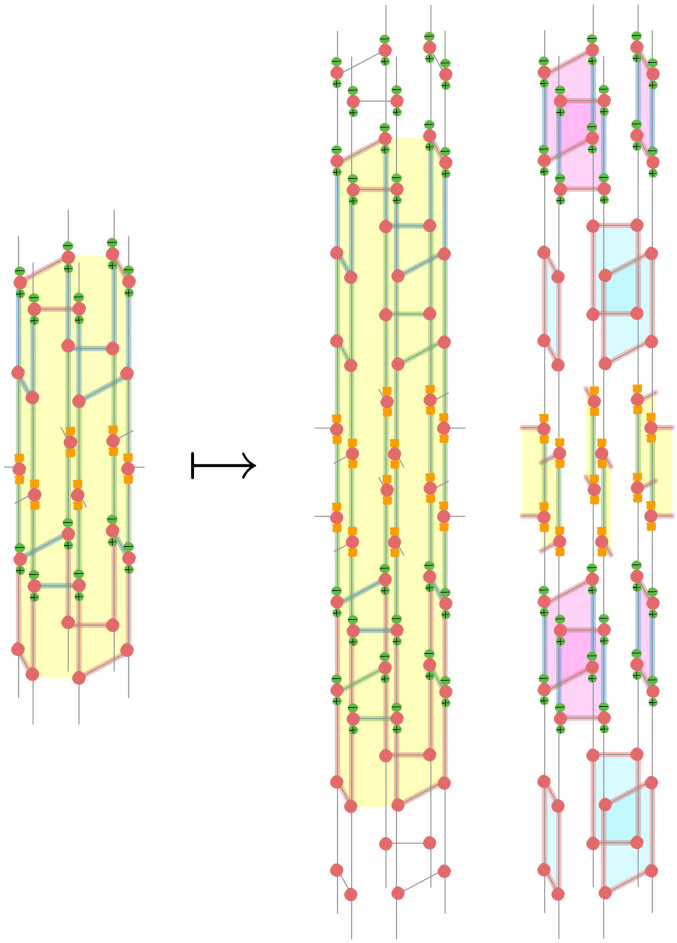


**Figure 19:** Left: a detector in the $X^1Y^1Z^1$ code with volume $6 \cdot 4 = 24$ consisting of 12 measurements (as can be read off from the Pauli web – see Figure 10). Middle: the analogous detector in the repeated $X^2Y^2Z^2$ code, which now has volume $6 \cdot ((4-1) \cdot 2 + 1) = 42$ but still consists of 12 measurements. Right: the edge detectors on these six qubits in the $X^2Y^2Z^2$ code, each of which has volume 2 and consists of two consecutive edge measurements. As discussed more in the caption of Table 5, repeating measurements reduces the mean detector volume and measurements per detector, while increasing the mean number of detectors per measurement – all of which our intuition says are improvements. On the other hand, it increases the maximum detector volume. In any case, our simulations show that these improvements don't generally outweigh the negative effects of repeating measurements, except in the case of high measurement bias.

being violated, which makes the job of the decoder easier, as discussed earlier. Secondly, we still want to be able to use a matching decoder.

Letting $T = \max(\tau)$ be the latest timestep in $\tau$, we can achieve these two conditions by using measurements from the final round $R_t + r_t - 1$ of $r_t$ repeated measurement rounds for all $t \neq T$, but using measurements from the first round $R_T$ of $r_T$ repeated measurements rounds for $t = T$

$$\prod_{(t \in \tau - \{T\})} \prod_{(e_{\kappa_t} \in f_\kappa)} m_{R_t + r_t - 1}(e_{\kappa_t, P_t}) \prod_{(e_{\kappa_T} \in f_\kappa)} m_{R_T}(e_{\kappa_T, P_T}). \tag{8}$$

Indeed, this is the specific choice of measurements used in the example in Figure 19. In this new repeated DCCC, we also have detectors with no analogue to any of those in the

original DCCC. Suppose we have just measured all $(\kappa, P)$ edge operators at some timestep $R_t$ for the first of $r_t$ repeated measurement rounds. Each edge operator $e_{\kappa,P}$ is therefore now a stabilizer up to some sign $m_{R_t}(e_{\kappa,P})$. Assuming $r_t > 1$, at the next timestep $R_t + 1$ we measure all these $(\kappa, P)$ edge operators again. The stabilizer formalism says that each such measurement will give us the deterministic outcome $m_{R_t+1}(e_{\kappa,P}) = m_{R_t}(e_{\kappa,P})$ in the absence of any noise, so the formal product $m_{R_t+1}(e_{\kappa,P})m_{R_t}(e_{\kappa,P})$ is a detector, for all $\kappa$-coloured edges $e_\kappa$. Indeed, we get two-measurement detectors like these at every subsequent timestep in the set of $r_t$ repeated measurement rounds. That is, for every $u \in \{1, \ldots, r_t - 1\}$, we get a detector $m_{R_t+u}(e_{\kappa,P})m_{R_t+u-1}(e_{\kappa,P})$ for all $\kappa$-coloured edges $e_\kappa$. These are shown in the rightmost diagram in Figure 19. Since each such detector is associated to an edge, we call them *edge detectors*, and will refer to the other type of detectors (those associated to a face) as *face detectors*.

### 5.1.1 Volume, measurements per detector, detectors per measurement

Let's now contrast these new detectors in the repeated DCCC with those of the corresponding unrepeated DCCC. First note the number of timesteps spanned by a face detector in the repeated DCCC is at least as large as in the unrepeated DCCC. Indeed, a face detector in an unrepeated DCCC whose first measurements were at timestep $t$ and final measurements were at timestep $T$, and thus spanned $T - t$ timesteps, will now span $R_T - (R_t + r_t - 1)$ timesteps. If any $r_j > 1$ for $j \in \{t+1, \ldots, T-1\}$ then this is larger than $T-t$. We'll define the *volume* of a detector in a DCCC[7] as the product of the number of timesteps it spans and the number of data qubits involved in the measurements that make up this detector. Intuitively, the smaller the volume, the better, because a smaller volume means there are fewer opportunities for this detector to be violated. For example, the detectors in the unrepeated $X^1Y^1Z^1$ and $X^1Z^1$ codes all have volume $6 \cdot 4 = 24$ (Figure 10). In a repeated version with $r_t$ set to some $r$ for all $t$, these detectors would have volume $6 \cdot ((4-1)r + 1)$ (Figure 19).

On the other hand, the number of measurements per face detector remains exactly the same – e.g. 12 for any **r**-repeated $X^aY^bZ^c$ code, and 6 for any **r**-repeated $X^aZ^b$ code. Thanks to the new edge detectors, however, which always have volume two and consist of two measurements, the *average* detector volume and *average* number of measurements per detector decreases. Indeed, in the limit where $r_t \to \infty$ for all $t$, the edge detectors dominate, and these averages both tend to two. For our two favourite DCCCs, we compute these averages exactly in Table 5.

**Difference 5.1.** *Repeating measurements increases the maximum detector volume of a DCCC, while decreasing its minimum detector volume and minimum measurements per detector. Overall, it decreases a DCCC's average detector volume and average number of measurements per detector.*

Intuitively, smaller volumes and fewer measurements per detector are better, so the fact these averages decrease might suggest a general performance improvement. However, there is an important trade-off: while the new edge detectors catch measurement errors quickly (with volume 2), data qubit errors can accumulate for longer before being detected.

[7] Not to be confused with the teraquop volume of the DCCC as a whole! Also, as noted in a previous footnote, a more accurate metric capturing the same intuition is detector likelihood [65].

| | | $X^rY^rZ^r$ | $X^rZ^r$ |
|---|---|---|---|
| Detector volume | Max | $6 \cdot (3r+1) \to \infty$ | $6 \cdot (3r+1) \to \infty$ |
| | Min | 2 | 2 |
| | Average | $2 + \frac{22}{3r-2} \to 2$ | $2 + \frac{22}{3r-2} \to 2$ |
| Meas. per detector | Max | 12 | 6 |
| | Min | 2 | 2 |
| | Average | $2 + \frac{10}{3r-2} \to 2$ | $2 + \frac{4}{3r-2} \to 2$ |
| Detector degree | Max | $12 + 6(r-1) \to \infty$ | $12 + 6(r-1) \to \infty$ |
| | Min | 4 | 4 |
| | Average | $6 + \frac{6}{3r-2} \to 6$ | $6 + \frac{6}{3r-2} \to 6$ |

**Table 5:** Comparison of metrics concerning detectors in the $X^rY^rZ^r$ and $X^rZ^r$ codes, for $r > 1$. The arrows denote the limit of an expression as $r$ tends to infinity. As described in Difference 5.1, repeating measurements increases a DCCC's max. detector volume while decreasing its min. detector volume and min. measurements per detector. Overall, it decreases a DCCC's average detector volume and average number of measurements per detector. Intuitively smaller volumes and fewer measurements per detector are better, so the fact these averages decrease might suggest a general performance improvement. On the other hand, the key metric could be the maximum or minimum over all these values.

### 5.1.2 Asymmetric schedules

By using asymmetric schedules (where $r_t$ is not the same for all $t$), we can try to further tailor our code to $Z$-biased noise models. Here the intuition is that if physical $Z$ errors are more likely to occur, we can spend more time measuring $e_{\kappa,X}$ and $e_{\kappa,Y}$ operators, which will create more edge detectors that are violated by Pauli $Z$ errors. The more of these we have, the more accurately we can detect and correct Pauli $Z$ errors.

For example, consider the $X^aZ^b$ code, where we repeat all $e_{\kappa,X}$ operators $a$ times and all $e_{\kappa,Z}$ operators $b$ times. In a $Z$-biased noise model, we expect that setting $a$ to be larger than $b$ improves performance. Similarly, consider the $X^aY^bZ^c$ code, suppose we repeat all $e_{\kappa,X}$ operators $a$ times, all $e_{\kappa,Y}$ operators $b$ times, and all $e_{\kappa,Z}$ operators $c$ times. Here we expect that setting $a$ and $b$ to be larger than $c$ will improve performance.

**Difference 5.2.** *In a repeated DCCC, we can choose to spend more time measuring operators that will form detectors capable of detecting certain types of Pauli errors. This is particularly relevant for $Z$-biased noise models.*

## 5.2 Logical errors

The $(n_M, m_E, h_{\mathbf{r}})$-block definition from Subsection 3.1 still makes sense for any $\mathbf{r}$-repeated DCCC; we just write $h_{\mathbf{r}}$ instead of $h$ to make it clear that we're considering a repeated DCCC. Repeating measurements can increase our code's resilience against spacelike logical errors consisting only of measurement errors. Consider such a logical error in the unrepeated DCCC, consisting of measurement errors on a set of edge operators $\{e_{\kappa,P}\}$ all

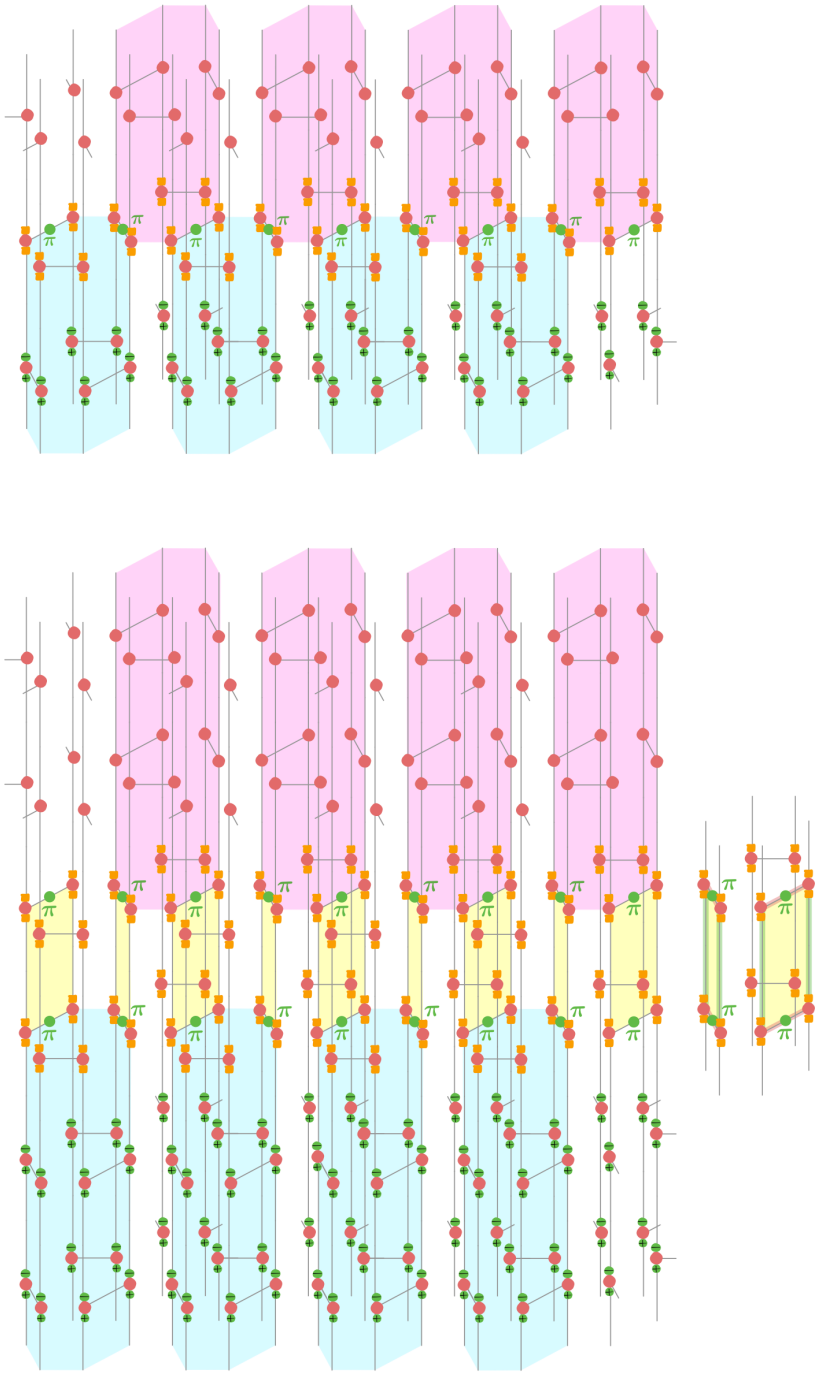


**Figure 20:** Top: a spacelike logical error in the $X^1Y^1Z^1$ code consisting entirely of measurement errors. Bottom: the analogous spacelike logical error in the $X^2Y^2Z^2$ code. This again consists entirely of measurement errors, but there are twice as many of them as before, because of the addition of the yellow edge detectors in the $X^2Y^2Z^2$ code. In both diagrams we only show one type ($E$ or $M$) of detectors. We draw the Pauli webs for two edge detectors to show that the logical error really doesn't violate any detectors (see Figure 11 or Section B), but otherwise omit the Pauli webs – they can be found in Section C. There is a bijection between spacelike logical errors consisting entirely of measurement errors in the $X^1Y^1Z^1$ code and in any $X^aY^bZ^c$ code. Hence repeating measurements in general increases the spacelike distance with respect to pure measurement noise (Table 6). One might then expect this to improve the teraquop width $\tilde{n}_E$ and length $\tilde{n}_M$, especially in more measurement-biased noise models (Difference 5.3). But the simulations don't seem to back this up (Figure 25).

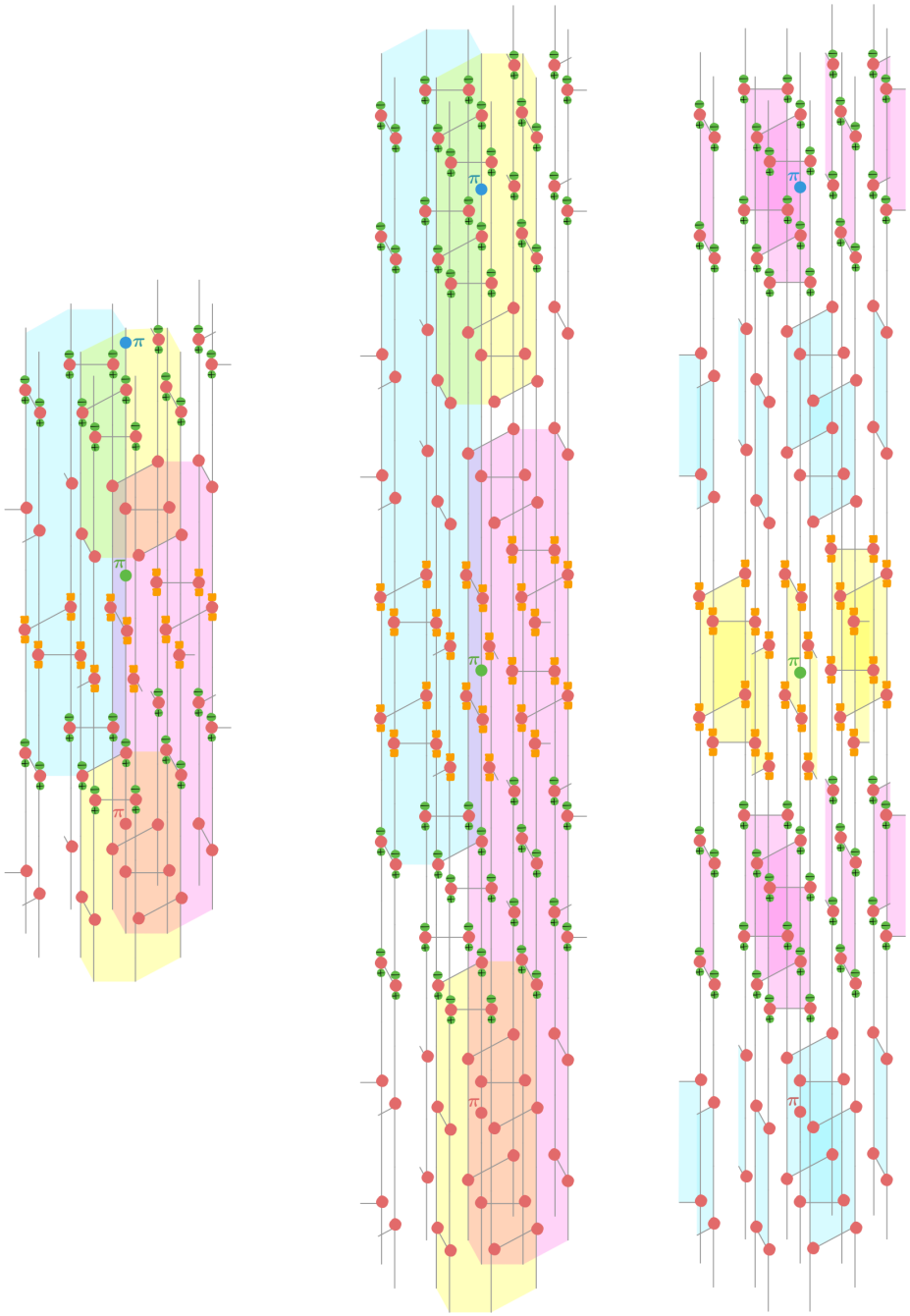


**Figure 21:** Left: a subset of a timelike logical error in the $X^1Y^1Z^1$ code consisting entirely of data qubit errors. Middle: the analogous timelike logical error in the $X^2Y^2Z^2$ code, with half ($E$ or $M$) of the face detectors indicated. Right: the same logical error but with the edge detectors indicated. In all diagrams, the Pauli webs are omitted – they can be found in Section C. The logical error in the $X^2Y^2Z^2$ code in the middle and right subfigures again consists entirely of data qubit errors, but these data qubit errors now occur once every four rounds rather than once every two rounds. In other words, if these two codes were implemented for the same number of rounds, the logical in the $X^2Y^2Z^2$ code would have half the weight of the one in the $X^1Y^1Z^1$ code. Indeed, repeating measurements in general decreases the timelike distance with respect to pure data qubit noise (Table 6). One would then expect this to worsen the teraquop heights $\tilde{h}_E$ and $\tilde{h}_M$, especially in more $Z$-biased noise models (Difference 5.4). While the simulation results do show this to be the case (Figure 25), the fact that this difference is greatest for unbiased noise rather than $Z$-biased noise suggests its cause is something more than the reason discussed here.

occurring at the timestep $t$. For every error in this set, consider an error on the measurement of $e_{\kappa,P}$ at *all* $r_t$ repeated timesteps $R_t + u_t$ for $u_t \in \{0, \ldots, r_t - 1\}$. The union of all such errors forms a spacelike logical error in the repeated DCCC. The key here is that each new edge detector $m_{R_t+u_t}(e_{\kappa,P})m_{R_t+u_t-1}(e_{\kappa,P})$ would be violated by a measurement error on $e_{\kappa,P}$ at timestep $R_t + u_t$ or $R_t + u_t - 1$. So in order for no such detector to be violated, we need the measurements at both timesteps to be erroneous, thereby cancelling each other out. All pure measurement noise spacelike errors in the repeated DCCC are formed in this way. So the spacelike distance with respect to these types of logical errors specifically is $(\min_t r_t) \cdot d_s$. An example is shown in Figure 20.

**Difference 5.3.** *Repeating measurements increases a DCCC's resilience against spacelike logical errors consisting entirely of measurement errors. This is particularly relevant in measurement biased noise models.*

On the other hand, repeating measurements generally decreases a DCCC's resilience against timelike logical errors consisting only of data qubit errors. Again, all such logical errors are analogues of those in the unrepeated DCCC. Consider such an error in the unrepeated DCCC. For every Pauli $P$ data qubit error on qubit $j$ immediately after timestep $t$ in the unrepeated DCCC, the analogous logical error in the repeated DCCC consists of a Pauli $P$ error on qubit $j$ at *any* timestep $R_t + u_t$ for $u_t \in \{0, \ldots, r_t - 1\}$. Letting $\tau$ again be the set of timesteps of the measurement errors in the unrepeated DCCC, where before the logical error consisted of $|\tau|$ errors across $h$ timesteps, it's now $|\tau|$ across $h_{\mathbf{r}} = \sum_{0<t<h} r_t$. As soon as $r_t > 1$ for some $t$, this is a decrease. For example, in an $(n_M, n_E, h)$ block of the $X^1Y^1Z^1$ code, in which there exist timelike logical errors consisting of a data qubit error every 2 rounds. Now suppose we repeat all measurements some fixed number of times $r > 1$ to get an $(n_M, n_E, h_{\mathbf{r}})$ block of the $X^rY^rZ^r$ code. Under pure data qubit noise, the unrepeated code has a timelike distance of $\frac{h}{2}$, whereas the repeated code has a (worse) timelike distance of $\frac{h_{\mathbf{r}}}{2r}$. See Figure 21.

**Difference 5.4.** *Repeating measurements decreases the resilience of a DCCC to timelike logical errors made of data qubit errors.*

The spacelike distance under pure Pauli noise and the timelike distance under pure measurement noise are unchanged from the unrepeated case. So altogether we expect the two differences from this section to mean that a repeated DCCC's teraquop footprint $f(\tilde{n}_M\tilde{n}_E)$ is decreased, but that its teraquop height $\tilde{h}$ is increased. In fact, though it gets hard to discuss analytically for asymmetric schedules, repeating measurements generally reduces not just the timelike distance for logical errors made entirely of data qubit errors, but the timelike distance in general. We show this in Figure 26 – the topmost subfigure shows the direct parity measurement noise case.

## 5.3 Decoding graph

Repeating measurements has an effect on the number of measurement errors that correspond to hyperedges in the decoding graph. For example, in the unrepeated case, all measurement errors in the $X^1Y^1Z^1$ honeycomb code corresponded to hyperedges (Figure 11). But repeating each measurement at time $t$ a number of times $r_t$, only one out of every $r_t$ measurement errors now corresponds to a hyperedge. The remaining measurement errors correspond to ordinary edges – such as the example in Figure 22. (For the $X^1Z^1$ code, in which all measurement errors already corresponded to ordinary edges in the hypergraph in the unrepeated case, nothing changes in this regard when we repeat measurements.)

**Difference 5.5.** *The more we repeat measurements, the fewer measurement errors correspond to hyperedges in the decoding graph.*

So the more we repeat measurements, the less we expect the repeated $X^aY^bZ^c$ code to benefit from using belief matching. In particular, noise models biased towards measurement errors will see less benefit from using belief matching.

The other key difference pertains to vertex degrees in a DCCC's decoding graph. The degree of vertices in a code's decomposed decoding graph in general affects decoding performance; the lower the better [18]. In any unrepeated DCCC, we said every vertex (i.e. detector) has degree 12 (Figure 12). Repeating measurements introduces edge detectors, whose corresponding vertices always have degree 4 (Figure 23), but it also increases the degree of the vertices corresponding to face detectors. For the $X^aY^bZ^c$ and $X^aZ^b$ codes this is overall a positive change, in the sense that the average vertex degree in the decomposed decoding graph tends from 12 down to 6 as the number of repeated measurements increases.

**Difference 5.6.** *Repeating measurements decreases the average vertex degree in the decoding graph of a DCCC.*

This difference would seem to predict a general performance increase for repeated DCCCs versus their unrepeated counterparts.

## 5.4 Direct parity measurement noise results

Figures 1 and 2 show the teraquop volumes of the best $X^aY^bZ^c$ and $X^aZ^b$ codes under direct parity measurement noise models of different biases, decoded with MWPM and belief matching respectively.

**Key point 5.7.** *Repeating measurements improves the teraquop volumes of both codes under both decoders when the noise is measurement-biased, but otherwise it has a negligible effect.*

One can see this most clearly by looking at the hearts (♡) subplot in Figures 1 and 2; under MWPM (Figure 1) all but one of the ten best codes involve repeating measurements in some way, and under belief matching (Figure 2) it's all ten. This chimes with the intuition that repeating measurements gives us more confidence about faulty measurements. At the same time it seems to contradict the intuition that repeating measurements might improve performance in general (Differences 5.1 and 5.6), and particularly in the case of $Z$-biased noise by spending more time measuring certain operators (Difference 5.2). That said, we only tested a $Z$-noise bias of up to $\eta_Z = 16$, whereas in other work orders of magnitude higher values are considered – e.g. bias of up to $\eta_Z = 10^6$ is considered in Ref. [68].

## 5.5 Auxiliary qubit circuit noise results

Figures 3 and 4 show the teraquop volumes of the best codes under the three auxiliary qubit circuit noise models, decoded with MWPM and belief matching respectively. In all three cases, there is not much to be gained from repeating measurements. Perhaps the only exception is that in the belief matching case under the EM3 noise model, the $X^2Z^2$ code has a noticeably better teraquop volume than the $X^1Z^1$ code. But both are outperformed

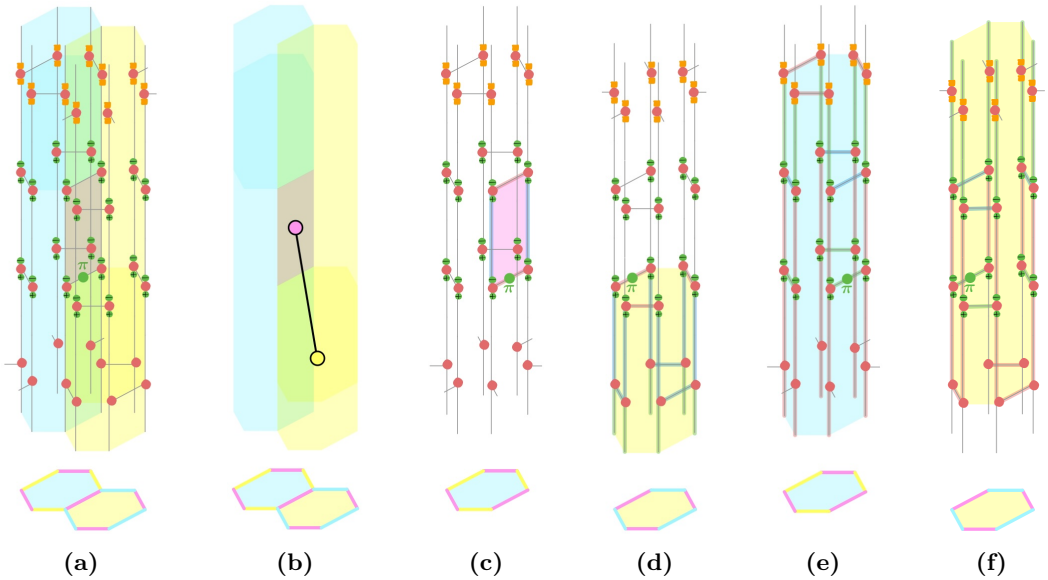
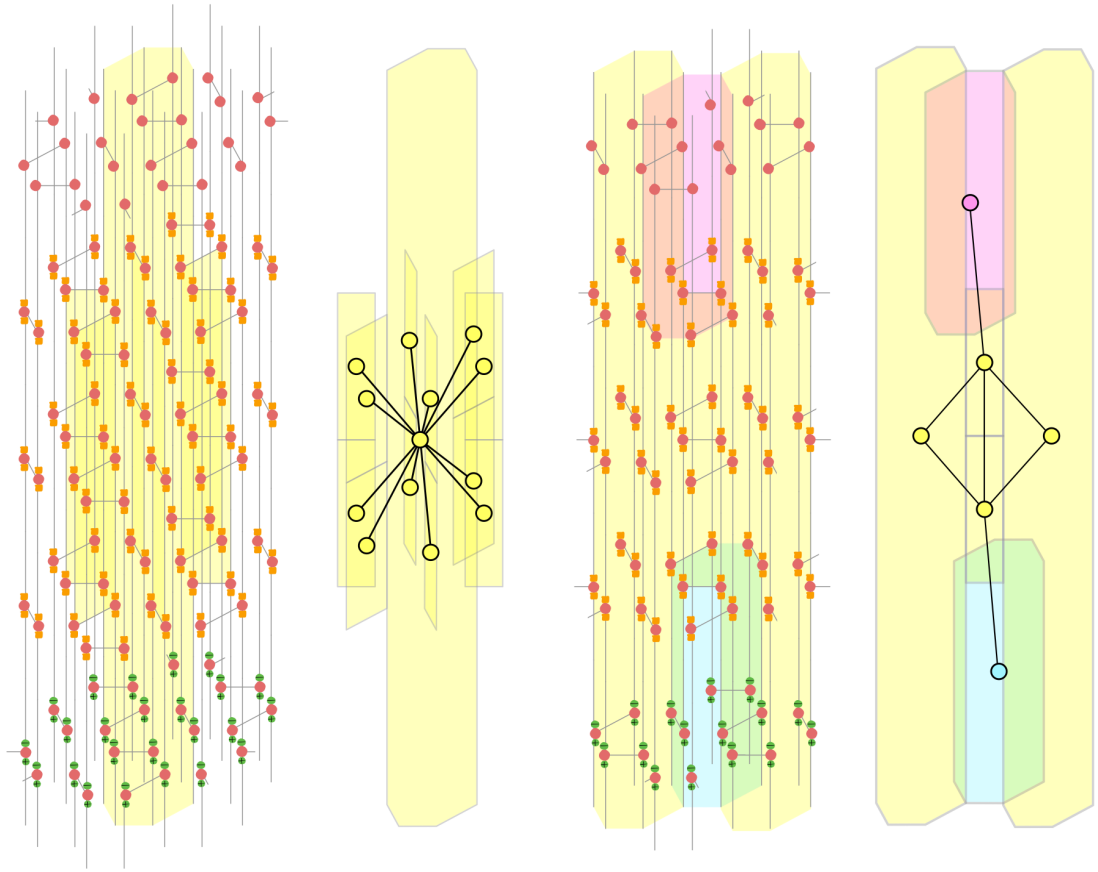



**Figure 22:** **(a)** A measurement error in the $X^2Y^2Z^2$ code. **(b)** The ordinary edge it corresponds to in the decoding graph. **(c, d)** The $E$ detectors the error violates, drawn as Pauli webs. **(e, f)** The $M$ detectors the error sits on but does not violate, drawn as Pauli webs. Compare this figure with Figure 11, where an analogous error in the $X^1Y^1Z^1$ code corresponded to a hyperedge in the decoding graph. This would still have been the case in the $X^2Y^2Z^2$ code had the measurement error occurred in the last of the $r = 2$ repeated rounds of measurements. But for all other rounds such an error now corresponds to an ordinary edge. In general, repeating measurements reduces the number of measurement errors corresponding to hyperedges, which would suggest a plateau in the improvement from using belief matching over MWPM (Difference 5.5), and indeed this seems to be the case numerically (Figure 24).

| | $X^rY^rZ^r$ | | | $X^rZ^r$ | | |
|---|---|---|---|---|---|---|
| | Direct parity meas. | Meas. noise | Z noise | Direct parity meas. | Meas. noise | Z noise |
| $d_{s,E}$ | $n_E$ | $rn_E \to \infty$ | $n_E$ | $n_E$ | $rn_E \to \infty$ | $n_E$ |
| $d_{s,M}$ | $\frac{4}{3}n_M$ | $r\frac{4}{3}n_M \to \infty$ | $\frac{4}{3}n_M$ | $\frac{4}{3}n_M$ | $r\frac{4}{3}n_M \to \infty$ | $\infty$ |
| $d_{t,E}$ | $\frac{h_\mathbf{r}}{2r} \to 0$ | $\frac{h_\mathbf{r}}{2}$ | $\frac{h_\mathbf{r}}{2r} \to 0$ | $\min\{\frac{h_\mathbf{r}}{2r}, \frac{h_\mathbf{r}}{4}\} \to 0$ | $\frac{h_\mathbf{r}}{4}$ | $\frac{h_\mathbf{r}}{2r} \to 0$ |
| $d_{t,M}$ | $\frac{h_\mathbf{r}}{2r} \to 0$ | $\frac{h_\mathbf{r}}{2}$ | $\frac{h_\mathbf{r}}{2r} \to 0$ | $\min\{\frac{h_\mathbf{r}}{2r}, \frac{h_\mathbf{r}}{4}\} \to 0$ | $\frac{h_\mathbf{r}}{4}$ | $\infty$ |
| $p_h$ | $\frac{2}{9} + \frac{1}{3r} \to \frac{2}{9}$ | $\frac{1}{r} \to 0$ | $\frac{1}{3}$ | $\frac{2}{9}$ | 0 | 0 |

**Table 6:** Comparison of the $X^rY^rZ^r$ and $X^rZ^r$ codes, for $r > 1$. The first four rows show the spacelike $E$ and $M$ distances and timelike $E$ and $M$ distances respectively. The final row shows the hyperedge likelihood. The arrows denote the limit of an expression as $r$ tends to infinity. Repeating measurements increases the spacelike distances under pure measurement noise (Difference 5.3), but decreases the timelike distances under pure data qubit noise (Difference 5.4). One might expect this to be reflected in the teraquop widths, lengths and heights, but the simulation results show this isn't the case (Figure 25). Repeating measurements also decreases the frequency with which measurement errors correspond to hyperedges in the decoding graph. This explains why the improvement given by decoding with belief matching rather than MWPM plateaus the more we repeat measurements (Figure 24).



**(a)** **(b)** **(c)** **(d)**

**Figure 23:** **(a)** Five timesteps of the $X^3Y^3Z^3$ code, centered on a yellow face. The shaded yellow areas denote a face detector and 12 edge detectors. **(b)** The edges between the face detector and the 12 edge detectors in the decoding graph. The face detector has 12 further edges (unshown) to other detectors in the ordinary decoding graph, exactly analogous to the 12 edges it has in *Figure* 12 in the $X^1Y^1Z^1$ code. So altogether it now has 24 edges. **(c)** The same five timesteps of the $X^3Y^3Z^3$ code, but now centered on a yellow edge, with shaded areas indicating detectors. **(d)** The neighbourhoods of the two edge detectors in the ordinary decoding graph. Note each has degree 4. All of the above holds more generally, in that in the ordinary decoding graph for each $X^aY^bZ^c$ and $X^aZ^b$ code, every edge detector will have degree 4, and every face detector will have degree $12 + 6(r-1)$, where $r$ is the number of repeated measurement rounds for the edges in consideration (here $r = 3$). So repeating measurements generally increases the maximum vertex degree but reduces the average vertex degree in the decoding graph (Table 5). The former suggests a general decrease in performance while the latter suggests an increase.

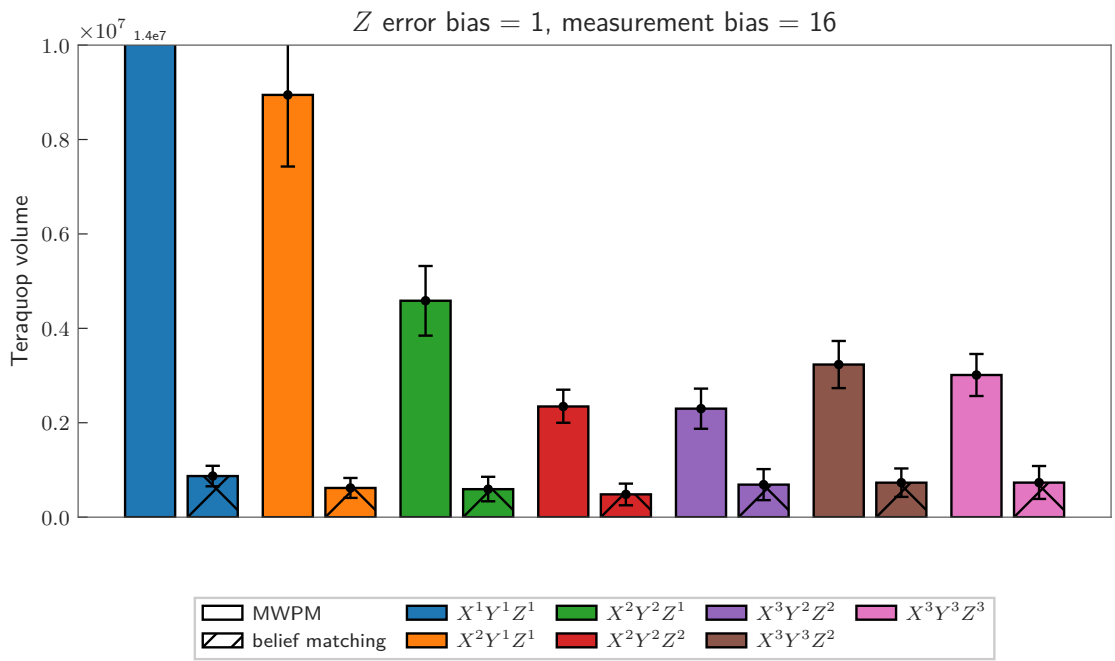



**Figure 24:** Teraquop volumes of the $X^aY^bZ^c$ code for direct parity measurement noise with measurement bias 16, and an increasing number $a + b + c$ of repeated rounds of measurements. As predicted by Difference 5.5, using belief matching rather than MWPM initially makes a huge difference to the volume, but this plateaus the more we repeat measurements.

by the $X^1Y^1Z^1$ code, which has the best teraquop footprint out of all the codes in this regime.

## 6 Discussion and outlook

DCCCs can be used to implement a surface code block, the basic unit of a lattice surgery quantum computer. In this work, we have optimized the overhead of a surface code block by tailoring the measurement schedule of DCCCs to direct parity measurement noise and auxiliary qubit circuit noise models. This was done by repeating measurements and measuring in different bases.

The metric we have introduced and optimized here is the teraquop volume, the spacetime overhead required to reach spacelike and timelike logical error rate $10^{-12}$. We have shown that the optimal measurement schedule depends on the noise model and the type of decoder used. We hope to see experimental demonstrations of DCCCs with tailored measurement schedules. Additionally, we hope that the teraquop volume metric gains wider adoption, as we consider it a crucial measure of overhead in quantum error correction and a tool to make reasonable design choices when setting up specific schemes for quantum error correction.

It would be interesting to see if tailoring measurement schedules can be combined with other techniques to further reduce the teraquop volume. Taking inspiration from Ref. [69], one could envision an adaptive measurement schedule. Furthermore, for strongly biased noise we expect a gain from combining repeated measurements with Clifford conjugated

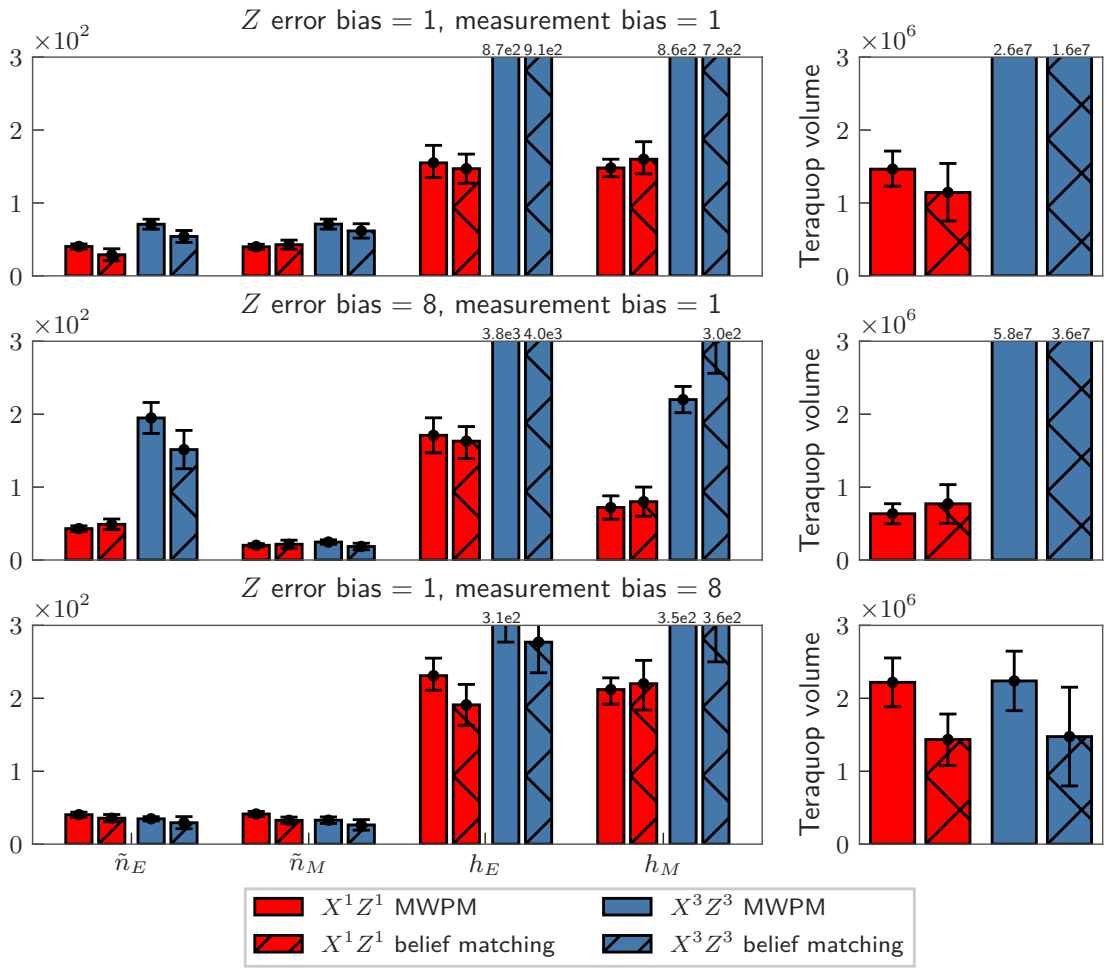



**Figure 25:** Comparison of $\tilde{n}_E, \tilde{n}_M, \tilde{h}_E, \tilde{h}_M$, and teraquop volume of the $X^1Z^1$ and $X^3Z^3$ code decoded using MWPM and belief matching for unbiased, $Z$-biased, and measurement-biased direct parity measurement noise. Based on Difference 5.3, we expected that repeating measurements would reduce the teraquop width and length $\tilde{n}_E$ and $\tilde{n}_M$, especially in the case of measurement-biased noise. Similarly, based on Difference 5.4, we expected that repeating measurements would increase the teraquop heights $\tilde{h}_E$ and $\tilde{h}_M$, especially in the case of $Z$-biased noise. Neither of these predictions are borne out by the simulation results here. In no cases does repeating measurements significantly reduce the teraquop width or length, and while it's true that repeating measurements increases the teraquop heights, the fact that this increase is greatest for unbiased noise suggests the cause is something more than Difference 5.4.

**Figure 26:** Plots showing the timelike distances of $(n_E, n_M, h)$-blocks of $X^aY^bZ^c$ and $X^aZ^b$ codes under different noise models as a fraction of the block height $h$. Since standard depolarizing noise and superconducting-inspired noise differ only in the probabilites of the certain gates occurring, and not in which gates can occur, any notions of distance are identical across both models. So we bundle these models together into a 'single-qubit measurements' noise model (middle row). One can see that the $X^aZ^b$ code timelike distances are unchanged across all three noise models. The timelike distances of $X^aY^bZ^c$ codes are in general reduced under single-qubit measurements noise as compared to direct parity measurement noise, but preserved under entangling measurements noise. This feature doesn't seem to be significant enough to appear in the simulation results – e.g. in Figure 4, under the two single-qubit measurements noise models the $X^1Y^1Z^1$ code still has the best teraquop volume.

DCCCs as in Ref. [39]. Time vortex defects [70] may further reduce teraquop volumes. It is also the hope that ideas such as those presented here will assist in taking steps for designing fault tolerant schemes for quantum computing beyond quantum memories.

## 7 Author contributions

Peter-Jan Derks and Alex Townsend-Teague led this project. Alex wrote most of the code to generate the circuits used in the numerical experiments; Peter-Jan wrote the rest of the code and produced the numerical results. Oscar Higgott wrote the implementation of the belief matching decoder used in this work. All authors contributed to the design of numerical experiments and the analysis of the results. Peter-Jan and Alex drafted the manuscript with input from all of the authors.

## 8 Acknowledgements

We thank Andreas Wallraff for discussions. The Berlin team acknowledges funding from BMBF (RealistiQ, QSolid, MuniQC-Atoms), the Munich Quantum Valley (K-8), the Quantum Flagship (Millenion, PasQuans2), Berlin Quantum, the Einstein Foundation (Einstein Research Unit on quantum devices), the European Research Council (DebuQC), and the DFG (CRC 183). O.H. acknowledges support from the Engineering and Physical Sciences Research Council (Grant No. EP/L015242/1) and a Google PhD fellowship.

## References

[1] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. "Topological quantum memory". J. Math. Phys. **43**, 4452–4505 (2002). arXiv:quant-ph/0110143.

[2] Dominic Horsman, Austin G. Fowler, Simon Devitt, and Rodney Van Meter. "Surface code quantum computing by lattice surgery". New J. Phys. **14**, 123011 (2012). arXiv:1111.4022.

[3] Daniel Litinski. "A game of surface codes: Large-scale quantum computing with lattice surgery". Quantum **3**, 128 (2019). arXiv:1808.02892.

[4] Craig Gidney and Martin Ekerå. "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits". Quantum **5**, 433 (2021). arXiv:1905.09749.

[5] Rajeev Acharya, Dmitry A. Abanin, Laleh Aghababaie-Beni, Igor Aleiner, Trond I. Andersen, Markus Ansmann, Frank Arute, Kunal Arya, Abraham Asfaw, Nikita Astrakhantsev, Juan Atalaya, et al. "Quantum error correction below the surface code threshold". Nature **638**, 920–926 (2025). arXiv:2408.13687.

[6] Laura Caune, Luka Skoric, Nick S. Blunt, Archibald Ruban, Jimmy McDaniel, Joseph A. Valery, Andrew D. Patterson, Alexander V. Gramolin, Joonas Majaniemi, Kenton M. Barnes, et al. "Demonstrating real-time and low-latency quantum error correction with superconducting qubits". Nature Communications (2026). arXiv:2410.05202.

[7] James R. Wootton. "Measurements of Floquet code plaquette stabilizers" (2022). arXiv:2210.13154.

[8] Alec Eickbusch, Matt McEwen, Volodymyr Sivak, Alexandre Bourassa, Juan Atalaya, Jahan Claes, Dvir Kafri, Craig Gidney, Christopher W. Warren, Jonathan Gross, et al. "Demonstration of dynamic surface codes". Nature Physics **21**, 1994–2001 (2025). arXiv:2412.14360.

[9] Hany Ali, Jorge Marques, Ophelia Crawford, Joonas Majaniemi, Marc Serra-Peralta, David Byfield, Boris Varbanov, Barbara M. Terhal, Leonardo DiCarlo, and Earl T. Campbell. "Reducing the error rate of a superconducting logical qubit using analog readout information". Phys. Rev. Appl. **22**, 044031 (2024). arXiv:2403.00706.

[10] Sebastian Krinner, Nathan Lacroix, Ants Remm, Agustin Di Paolo, Elie Genois, Catherine Leroux, Christoph Hellings, Stefania Lazar, Francois Swiadek, Johannes Herrmann, et al. "Realizing repeated quantum error correction in a distance-three surface code". Nature **605**, 669–674 (2022). arXiv:2112.03708.

[11] Robin Harper, Constance Lainé, Evan Hockings, Campbell McLauchlan, Georgia M. Nixon, Benjamin J. Brown, and Stephen D. Bartlett. "Characterising the failure mechanisms of error-corrected quantum logic gates". Nature Communications **17**, 5039 (2026). arXiv:2504.07258.

[12] Matthew B. Hastings and Jeongwan Haah. "Dynamically generated logical qubits". Quantum **5**, 564 (2021). arXiv:2107.02194.

[13] Markus S. Kesselring, Julio C. Magdalena de la Fuente, Felix Thomsen, Jens Eisert, Stephen D. Bartlett, and Benjamin J. Brown. "Anyon condensation and the color code". PRX Quantum **5**, 010342 (2024). arXiv:2212.00042.

[14] Margarita Davydova, Nathanan Tantivasadakarn, and Shankar Balasubramanian. "Floquet codes without parent subsystem codes". PRX Quantum **4**, 020341 (2023). arXiv:2210.02468.

[15] Hector Bombin, Daniel Litinski, Naomi Nickerson, Fernando Pastawski, and Sam Roberts. "Unifying flavors of fault tolerance with the ZX calculus". Quantum **8**, 1379 (2024). arXiv:2303.08829.

[16] J. Pablo Bonilla Ataides, David K. Tuckett, Stephen D. Bartlett, Steven T. Flammia, and Benjamin J. Brown. "The XZZX surface code". Nature Comm. **12**, 2172 (2021). arXiv:2009.07851.

[17] Andrew S. Darmawan, Benjamin J. Brown, Arne L. Grimsmo, David K. Tuckett, and Shruti Puri. "Practical quantum error correction with the XZZX code and Kerr-cat qubits". PRX Quantum **2**, 030345 (2021). arXiv:2104.09539.

[18] Oscar Higgott and Nikolas P. Breuckmann. "Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead". Phys. Rev. X **11**, 031039 (2021). arXiv:2010.09626.

[19] Jonathan F. San Miguel, Dominic J. Williamson, and Benjamin J. Brown. "A cellular automaton decoder for a noise-bias tailored color code". Quantum **7**, 940 (2023). arXiv:2203.16534.

[20] Basudha Srivastava, Anton Frisk Kockum, and Mats Granath. "The $XYZ^2$ hexagonal stabilizer code". Quantum **6**, 698 (2022). arXiv:2112.06036.

[21] Konstantin Tiurev, Peter-Jan H. S. Derks, Joschka Roffe, Jens Eisert, and Jan-Michael Reiner. "Correcting non-independent and non-identically distributed errors with surface codes". Quantum **7**, 1123 (2023). arXiv:2208.02191.

[22] Konstantin Tiurev, Arthur Pesah, Peter-Jan H. S. Derks, Joschka Roffe, Jens Eisert, Markus S. Kesselring, and Jan-Michael Reiner. "Domain wall color code". Phys. Rev. Lett. **133**, 110601 (2024). arXiv:2307.00054.

[23] David K. Tuckett, Stephen D. Bartlett, and Steven T. Flammia. "Ultrahigh error threshold for surface codes with biased noise". Phys. Rev. Lett. **120**, 050505 (2018). arXiv:1708.08474.

[24] David K. Tuckett, Andrew S. Darmawan, Christopher T. Chubb, Sergey Bravyi, Stephen D. Bartlett, and Steven T. Flammia. "Tailoring surface codes for highly biased noise". Phys. Rev. X **9**, 041031 (2019). arXiv:1812.08186.

[25] David K. Tuckett, Stephen D. Bartlett, Steven T. Flammia, and Benjamin J. Brown. "Fault-tolerant thresholds for the surface code in excess of 5% under biased noise". Phys. Rev. Lett. **124**, 130501 (2020). arXiv:1907.02554.

[26] Rigetti Computing. "Quantum processing units". https://qcs.rigetti.com/qpus. Accessed: 2025-05-09.

[27] IQM Quantum Computers. "IQM Radiance – High-Performance Quantum Computing Platform". https://meetiqm.com/products/iqm-radiance/ (2025). Accessed: 2025-05-09.

[28] Craig Gidney, Michael Newman, Austin G. Fowler, and Michael Broughton. "A fault-tolerant honeycomb memory". Quantum **5**, 605 (2021). arXiv:2108.10457.

[29] Craig Gidney, Michael Newman, and Matt McEwen. "Benchmarking the planar honeycomb code". Quantum **6**, 813 (2022). arXiv:2202.11845.

[30] Joonho Lee, Dominic W. Berry, Craig Gidney, William J. Huggins, Jarrod R. McClean, Nathan Wiebe, and Ryan Babbush. "Even more efficient quantum computations of chemistry through tensor hypercontraction". PRX Quantum **2**, 030305 (2021). arXiv:2011.03494.

[31] Thomas Häner, Samuel Jaques, Michael Naehrig, Martin Roetteler, and Mathias Soeken. "Improved quantum circuits for elliptic curve discrete logarithms". In Post-Quantum Cryptography: 11th International Conference, PQCrypto 2020, Paris, France, April 15–17, 2020, Proceedings 11. Pages 425–444. Springer (2020). arXiv:2001.09580.

[32] Craig Gidney. "How to factor 2048 bit RSA integers with less than a million noisy qubits" (2025). arXiv:2505.15917.

[33] Boris Bourdoncle, Peter-Jan Derks, Théo Dessertaine, and Johannes Frank. "Two layers, no swaps: Biplanar SPOQC architecture improves runtime of Fermi-Hubbard simulation" (2026). arXiv:2605.05315.

[34] Théo Dessertaine, Boris Bourdoncle, Aurélie Denys, Grégoire de Gliniasty, Pierre Colonna d'Istria, Gerard Valentí-Rojas, Shane Mansfield, and Paul Hilaire. "Enhanced fault-tolerance in photonic quantum computing: Comparing the honeycomb Floquet code and the surface code in tailored architecture" (2025). arXiv:2410.07065.

[35] Ming Lai Chan, Thomas J Bell, Love A Pettersson, Susan X Chen, Patrick Yard, Anders S Sørensen, and Stefano Paesani. "Tailoring fusion-based photonic quantum computing schemes to quantum emitters". PRX Quantum **6**, 020304 (2025). arXiv:2410.06784.

[36] Oscar Higgott and Nikolas P. Breuckmann. "Constructions and performance of hyperbolic and semi-hyperbolic Floquet codes". PRX Quantum **5**, 040327 (2024). arXiv:2308.03750.

[37] Evan Sutcliffe, Bhargavi Jonnadula, Claire Le Gall, Alexandra E. Moylett, and Coral M. Westoby. "Distributed quantum error correction based on hyperbolic Floquet codes". In 2025 IEEE International Conference on Quantum Computing and Engineering (QCE). Pages 649–657. IEEE (2025). arXiv:2501.14029.

[38] Bence Hetényi and James R. Wootton. "Creating entangled logical qubits in the heavy-hex lattice with topological codes". PRX Quantum **5**, 040334 (2024). arXiv:2404.15989.

[39] Fnu Setiawan and Campbell McLauchlan. "Tailoring dynamical codes for biased noise: The $X^3Z^3$ Floquet code". npj Quantum Information **11**, 149 (2025). arXiv:2411.04974.

[40] Craig Gidney. "A pair measurement surface code on pentagons". Quantum **7**, 1156 (2023). arXiv:2206.12780.

[41] Adam Paetznick, Christina Knapp, Nicolas Delfosse, Bela Bauer, Jeongwan Haah, Matthew B. Hastings, and Marcus P. da Silva. "Performance of planar Floquet codes with Majorana-based qubits". PRX Quantum **4**, 010310 (2023). arXiv:2202.11829.

[42] Esther Xiaozhen Fu and Daniel Gottesman. "Error correction in dynamical codes". Quantum **9**, 1886 (2025). arXiv:2403.04163.

[43] Alex Townsend-Teague, Julio Magdalena de la Fuente, and Markus Kesselring. "Floquetifying the colour code". Electronic Proc. Th. Compu. Sc. **384**, 265–303 (2023). arXiv:2307.11136.

[44] Michael A. Nielsen and I. L. Chuang. "Quantum computation and quantum information". Cambridge University Press. (2010). 10th anniversary edition.

[45] Daniel Gottesman. "Surviving as a quantum computer in a classical world - 2024 draft". `https://www.cs.umd.edu/~dgottesm/QECCbook-2024.pdf` (2024).

[46] Bob Coecke and Ross Duncan. "Interacting quantum observables: categorical algebra and diagrammatics". New J. Phys. **13**, 043016 (2011). arXiv:0906.4725.

[47] John van de Wetering. "ZX-calculus for the working quantum computer scientist" (2020). arXiv:2012.13966.

[48] Aleks Kissinger and John van de Wetering. "Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation". Preprint. (2024). url: `https://zxcalc.github.io/book/`.

[49] Craig Gidney. "Stim: a fast stabilizer circuit simulator". Quantum **5**, 497 (2021). arXiv:2103.02202.

[50] Peter-Jan H. S. Derks, Alex Townsend-Teague, Ansgar G. Burchards, and Jens Eisert. "Designing fault-tolerant circuits using detector error models". Quantum **9**, 1905 (2025). arXiv:2407.13826.

[51] Craig Gidney. "Where do lines like 'error(0.00033) D0 D9 L0 ^ D7 ^ D8' come from in a Stim detector error model?". `https://quantumcomputing.stackexchange.com/q/23768` (2022). Accessed: 2025-01-21.

[52] Jack Edmonds. "Paths, trees, and flowers". Canad. J. Math. **17**, 449–467 (1965).

[53] Aleksander Kubica and Nicolas Delfosse. "Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders". Quantum **7**, 929 (2023). arXiv:1905.07393.

[54] Nicolas Delfosse and Gilles Zémor. "Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel". Phys. Rev. Res. **2**, 033042 (2020). arXiv:1703.01517.

[55] Nicolas Delfosse and Naomi H. Nickerson. "Almost-linear time decoding algorithm for topological codes". Quantum **5**, 595 (2021). arXiv:1709.06218.

[56] Craig Gidney. "Stim: Command line usage documentation". `https://github.com/quantumlib/Stim/blob/main/doc/usage_command_line.md`. Accessed: 2025-01-21.

[57] Oscar Higgott, Thomas C. Bohdanowicz, Aleksander Kubica, Steven T. Flammia, and Earl T. Campbell. "Improved decoding of circuit noise and fragile boundaries of tailored surface codes". Phys. Rev. X **13**, 031007 (2023). arXiv:2203.04948.

[58] Oscar Higgott and Craig Gidney. "Sparse blossom: correcting a million errors per core second with minimum-weight matching". Quantum **9**, 1600 (2025). arXiv:2303.15933.

[59] Craig Gidney. "Crumble". `https://algassert.com/crumble` (2023). In-development tool for exploring and inspecting 2D stabilizer circuits.

[60] David J. C. MacKay. "Good error-correcting codes based on very sparse matrices". IEEE Trans. Inf. Th. **45**, 399–431 (1999).

[61] Matt Leifer and David Poulin. "Quantum graphical models and belief propagation". Ann. Phys. **323**, 1899–1946 (2008). arXiv:0708.1337.

[62] Joschka Roffe, David R. White, Simon Burton, and Earl T. Campbell. "Decoding across the quantum low-density parity-check code landscape". Phys. Rev. Res. **2**, 043423 (2020). arXiv:2005.07016.

[63] AG47. "Uncertainty of estimates computed by stim/sinter". https://quantumcomputing.stackexchange.com/q/37267 (2024). Accessed: 2025-03-13.

[64] Craig Gidney. "Stability experiments: The overlooked dual of memory experiments". Quantum **6**, 786 (2022). arXiv:2204.13834.

[65] Ian Hesner, Bence Hetényi, and James R Wootton. "Using detector likelihood for benchmarking quantum error correction". Phys. Rev. A **111**, 052452 (2025). arXiv:2408.02082.

[66] Craig Gidney and Dave Bacon. "Less Bacon more threshold" (2023). arXiv:2305.12046.

[67] M. Sohaib Alam and Eleanor Rieffel. "Dynamical logical qubits in the Bacon-Shor code". Phys. Rev. A **112**, 022436 (2025). arXiv:2403.03291.

[68] Christopher Chamberland, Kyungjoo Noh, Patricio Arrangoiz-Arriola, Earl T. Campbell, Connor T. Hann, Joseph Iverson, Harald Putterman, Thomas C. Bohdanowicz, Steven T. Flammia, Andrew Keller, Gil Refael, John Preskill, Liang Jiang, Amir H. Safavi-Naeini, Oskar Painter, and Fernando G.S.L. Brandão. "Building a fault-tolerant quantum computer using concatenated cat codes". PRX Quantum **3**, 010329 (2022). arXiv:2012.04108.

[69] Noah Berthusen, Shi J. S. Tan, Eric Huang, and Daniel Gottesman. "Adaptive syndrome extraction". PRX Quantum **6**, 030307 (2025). arXiv:2502.14835.

[70] Gilad Kishony and Erez Berg. "Increasing the distance of topological codes with time vortex defects". Quantum **10**, 2006 (2026). arXiv:2502.12236.

[71] Oscar Higgott. "BeliefMatching". https://github.com/oscarhiggott/BeliefMatching (2023).

[72] Benjamin Rodatz, Boldizsár Poór, and Aleks Kissinger. "Floquetifying stabiliser codes with distance-preserving rewrites" (2024). arXiv:2410.17240.

[73] Matt McEwen, Dave Bacon, and Craig Gidney. "Relaxing hardware requirements for surface code circuits using time-dynamics". Quantum **7**, 1172 (2023). arXiv:2302.02192.

[74] Kwok Ho Wan and Zhenghao Zhong. "Pauli web of the $|Y\rangle$ state surface code injection" (2025). arXiv:2501.15566.

[75] Julio C. Magdalena de la Fuente, Josias Old, Alex Townsend-Teague, Manuel Rispler, Jens Eisert, and Markus Müller. "The XYZ ruby code: Making a case for a three-colored graphical calculus for quantum error correction in spacetime". PRX Quantum **6**, 010360 (2025). arXiv:2407.08566.

## A Methodology

This section provides details on the numerics, though not exhaustively. For full reproducibility, the source code includes instructions for generating circuits, running simulations, and using Jupyter notebooks to produce all plots.

We built circuits that implement the memory and stability experiments using Stim [49]. For convenience, we simulate the DCCCs on a torus, but we check the error rate of 1 logical $X$ operator and 1 logical $Z$ operator. For phenomenological, SD, and SI noise we simulate distances $4, 8, 12$, and 16. For EM noise we simulate distances $2, 4, 6$, and 8. MWPM decoding is performed using the PyMatching package [58] and belief matching via the BeliefMatching package [57, 71]. We collect shots using Sinter. When decoding each

circuit with MWPM, we collect $10^6$ shots or as many shots required until 100 logical errors occur, whichever comes first. With belief matching, we collect $10^4$ shots or as many shots required until 100 logical errors occur, whichever comes first.

The highlighted regions in Figures 15 and 16 are computed with Sinter and show hypothesis probabilities within 1000× of the max likelihood hypothesis [63]. The error in $\tilde{n}_E, \tilde{n}_M, \tilde{h}_E, \tilde{h}_M$ shown in Figures 17, 18 and 25 is determined by fitting two lines with a maximum additional squared error set to 1 to the data using Sinter. The error is then given by the value at which these two lines intersect the horizontal line $y = 10^{-12}$. More info on the error in the line fits is given in Ref. [40, Appendix A]. We denote the maximum/minimum errors as $\tilde{n}_{E,\mathrm{max/min}}$, $\tilde{n}_{M,\mathrm{max/min}}$, $\tilde{h}_{E,\mathrm{max/min}}$, and $\tilde{h}_{M,\mathrm{max/min}}$. The error $\tilde{h}_{\mathrm{max/min}}$ is given by $\frac{\tilde{h}_{E,\mathrm{max/min}}+\tilde{h}_{M,\mathrm{max/min}}}{2}$. We denote the relative error of a variable $x$ as

$$\Delta x = \frac{x_{\mathrm{high}} - x_{\mathrm{best}}}{x_{\mathrm{best}}}. \tag{9}$$

The maximum and minimum error in the teraquop volume is given as

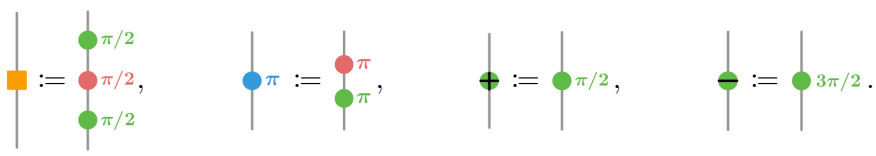
$$\epsilon_{\mathrm{max/min}} = \sqrt{\Delta\tilde{n}^2_{E,\mathrm{max/min}} + \Delta\tilde{n}^2_{M,\mathrm{max/min}} + \Delta\tilde{h}^2_{\mathrm{max/min}}}. \tag{10}$$

## B ZX-calculus and Pauli webs

Throughout this work we've used the *ZX-calculus* and *Pauli webs* in the figures. Here we give a very quick explainer of some of our notation, and give references for the reader interested in learning more. The ZX-calculus is a graphical language for reasoning about quantum mechanics. Two great introductions are Refs. [47, 48]. A ZX-diagram consists of green and red *spiders* with input and output legs, which one can think of as just very good notation for families of linear maps

$$:= |0\rangle^{\otimes n} \langle 0|^{\otimes m} + e^{i\alpha} |1\rangle^{\otimes n} \langle 1|^{\otimes m},$$
$$:= |+\rangle^{\otimes n} \langle +|^{\otimes m} + e^{i\alpha} |-\rangle^{\otimes n} \langle -|^{\otimes m}. \tag{11}$$

These spider legs can be connected together, corresponding to composition of linear maps, and spiders can be placed alongside each other, corresponding to taking tensor products. In this work, we additionally used the shorthands



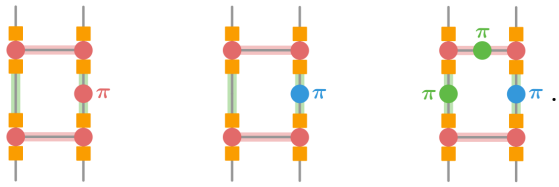

(12)

We also got a bit cheeky and drew some spiders/shorthands right on top of each other, hiding the connecting wire

(13)

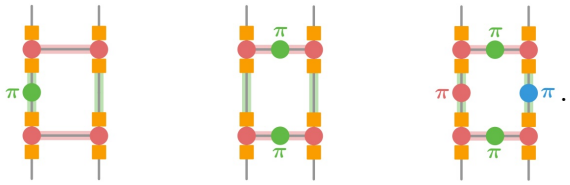
We continued to be cheeky when we said that certain diagrams correspond to measurements of $X \otimes X$, $Y \otimes Y$ and $Z \otimes Z$ in Figure 6. In reality, the diagrams there are equal (up to positive scalar) to the projections onto the $+1$ eigenspaces of $X \otimes X$, $Y \otimes Y$ and $Z \otimes Z$ respectively. In other words, it's as if we're assuming throughout that we always get measurement outcome $+1$, despite many of these measurements being totally random. It turns out that for reasoning about QEC protocols, we can get away with this, as explained (very briefly!) in Ref. [43, Appendix G].

ZX-diagrams can additionally be annotated with *Pauli webs*. One can think of these as describing how Pauli gates commute with individual spiders in a ZX-diagram. A nice introduction to them can be found in Ref. [72]. The important part with respect to making sense of the figures in this work is that 'closed' Pauli webs (ones in which no overall input or output leg of the ZX-diagram is highlighted) correspond to detectors. Whether a set of Pauli errors violates a detector can then be seen graphically – this happens if and only if there are an odd number of edges that are highlighted in a colour other than the colour of the Pauli error that sits on it. For example, the following sets of Pauli errors on consecutive $Z \otimes Z$ measurements violate the corresponding detector




(14)

On the other hand, these sets of Pauli errors don't violate it




(15)

So given a closed Pauli web, one can read off which measurements make up the corresponding detector, and which types of Pauli errors this detector can detect at different time steps. Namely, since measurement errors correspond to green two-legged $\pi$-spiders on 'horizontal' wires (Figure 8), the corresponding measurement is only included in a detector if this wire is highlighted red or blue. For example, in Figure 10 we can now read off that the detector in the $X^1Y^1Z^1$ honeycomb code consists of twelve detectors, while the detector in the $X^1Z^1$ honeycomb code consists of only six. For more details, see Ref. [72], or the handful of other works that make use of Pauli webs [15, 43, 73–75].

## C Detectors cheatsheet

In the following figures, we show Pauli webs of all detectors in the unrepeated $X^1Y^1Z^1$ and $X^1Z^1$ codes and the repeated $X^2Y^2Z^2$ and $X^2Z^2$ codes.

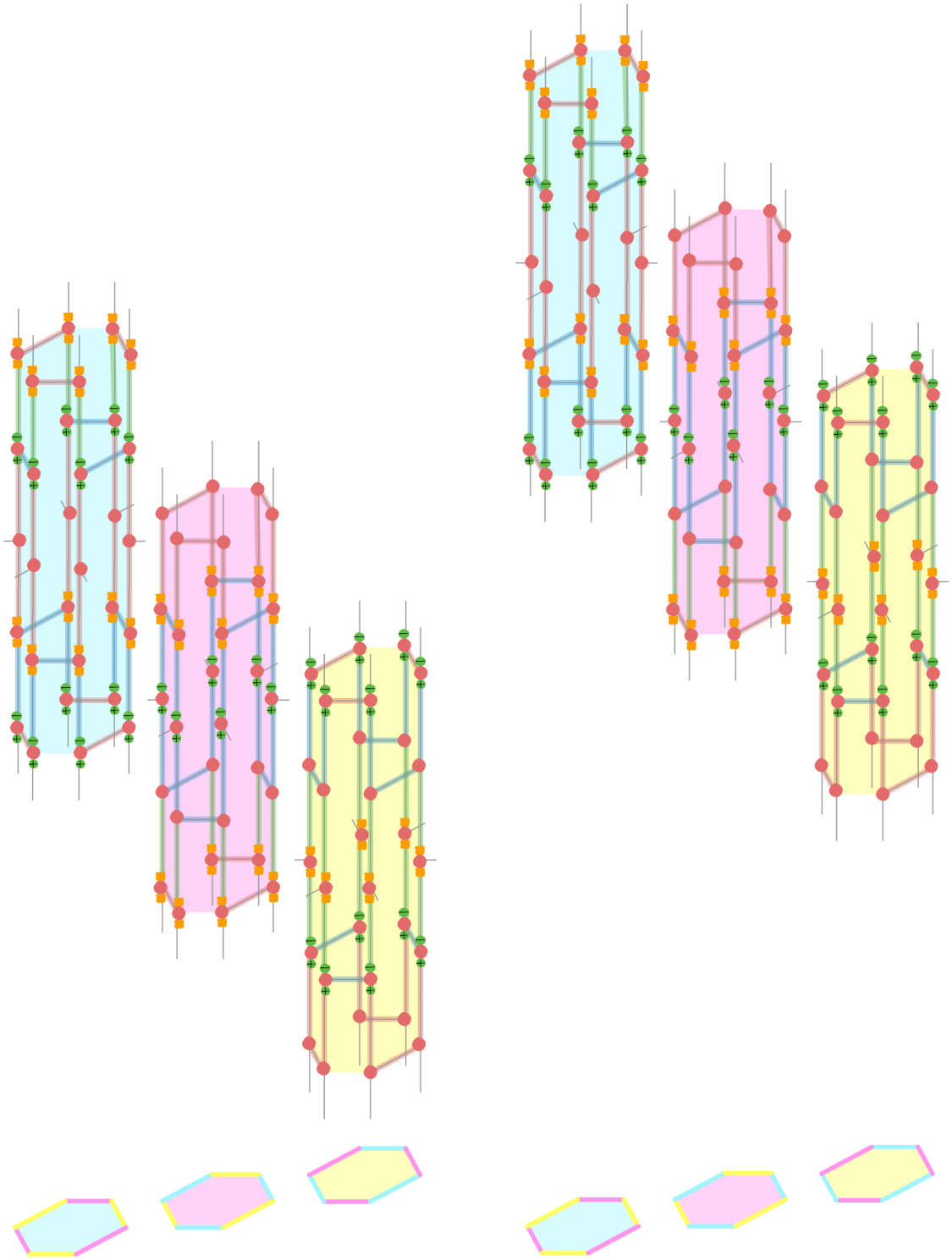


**Figure 27:** $E$ (left) and $M$ (right) detectors of the $X^1Y^1Z^1$ code.

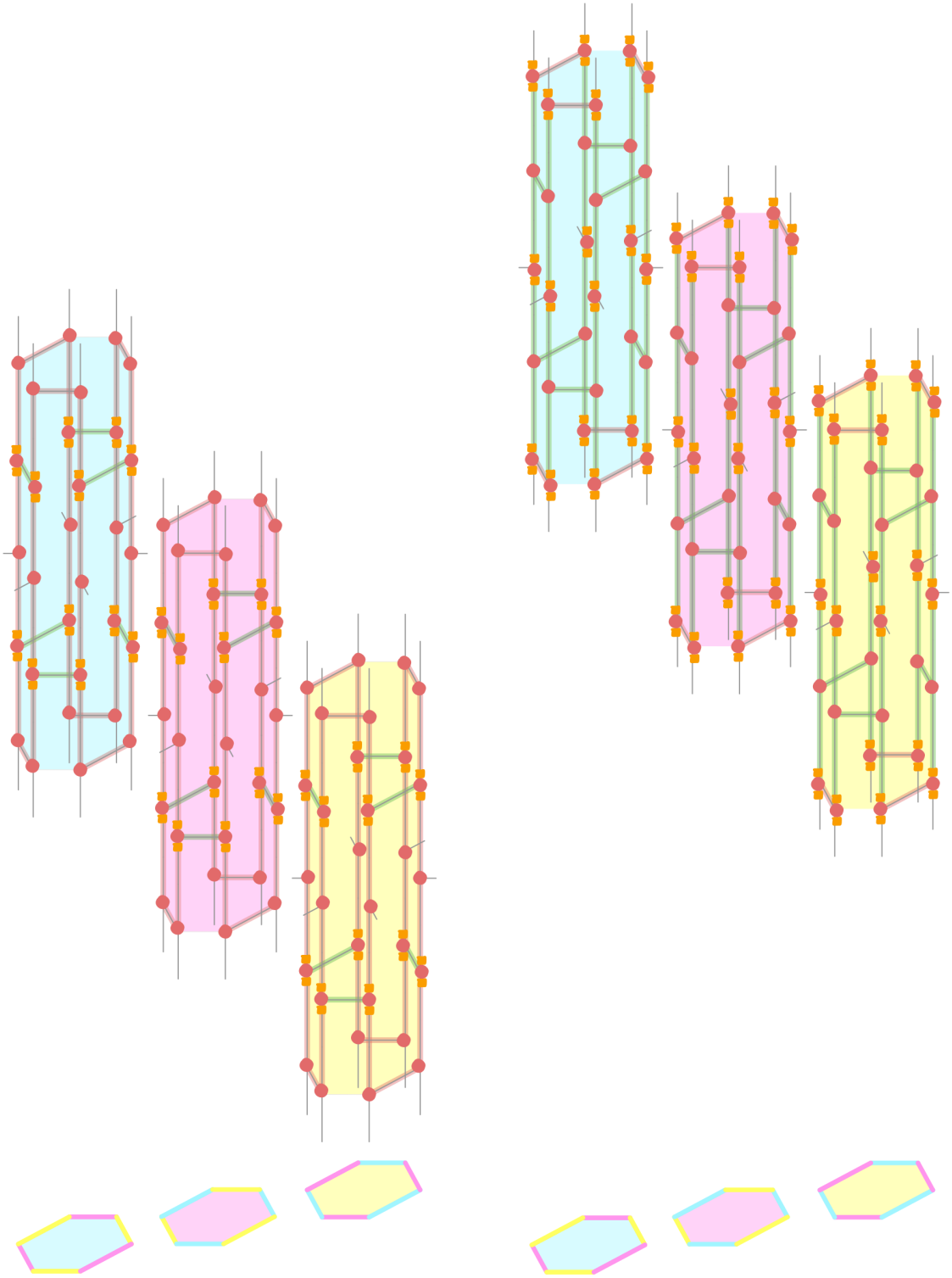


**Figure 28:** $E$ (left) and $M$ (right) detectors of the $X^1Z^1$ code.

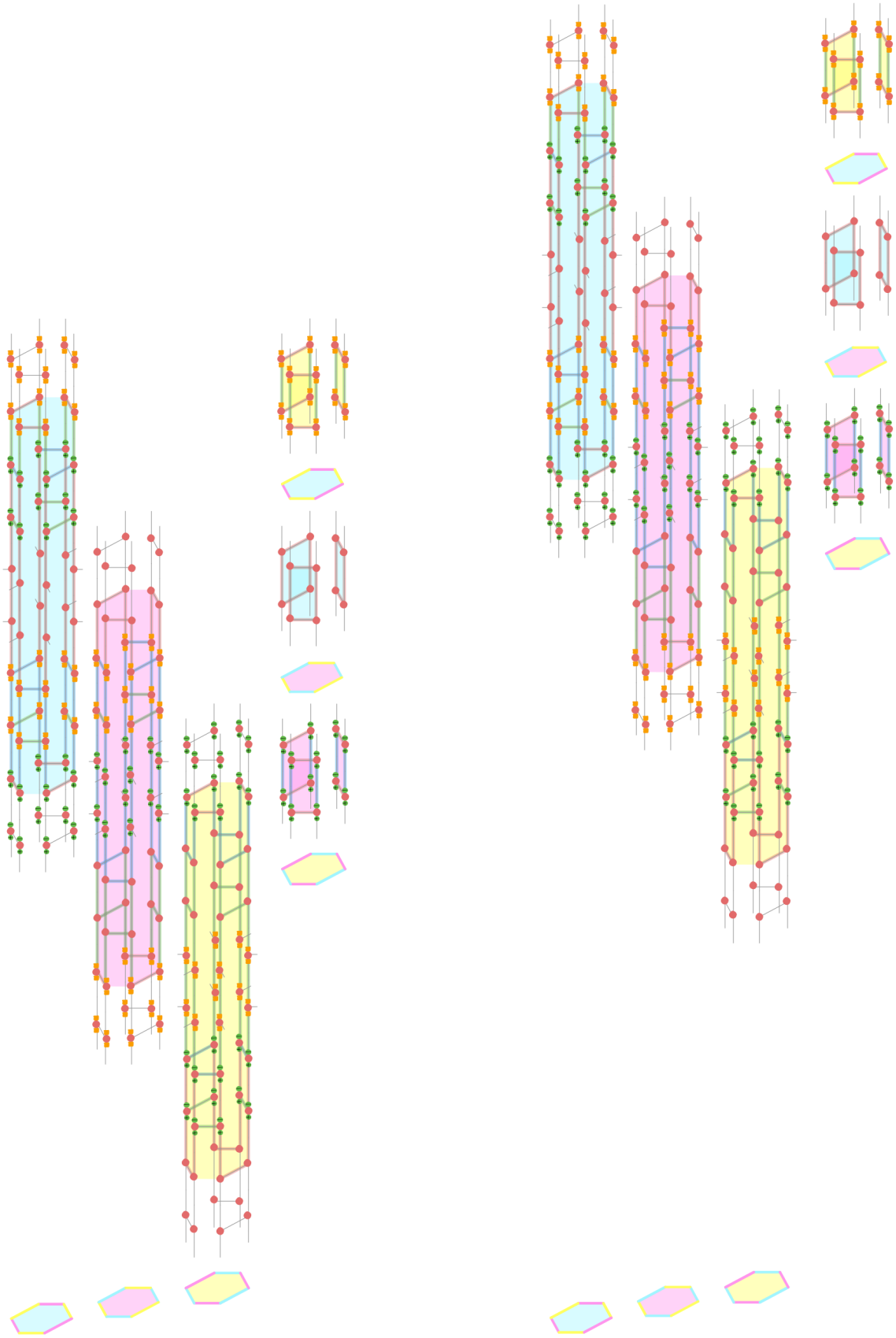


**Figure 29:** $E$ (left) and $M$ (right) detectors of the $X^2Y^2Z^2$ code.

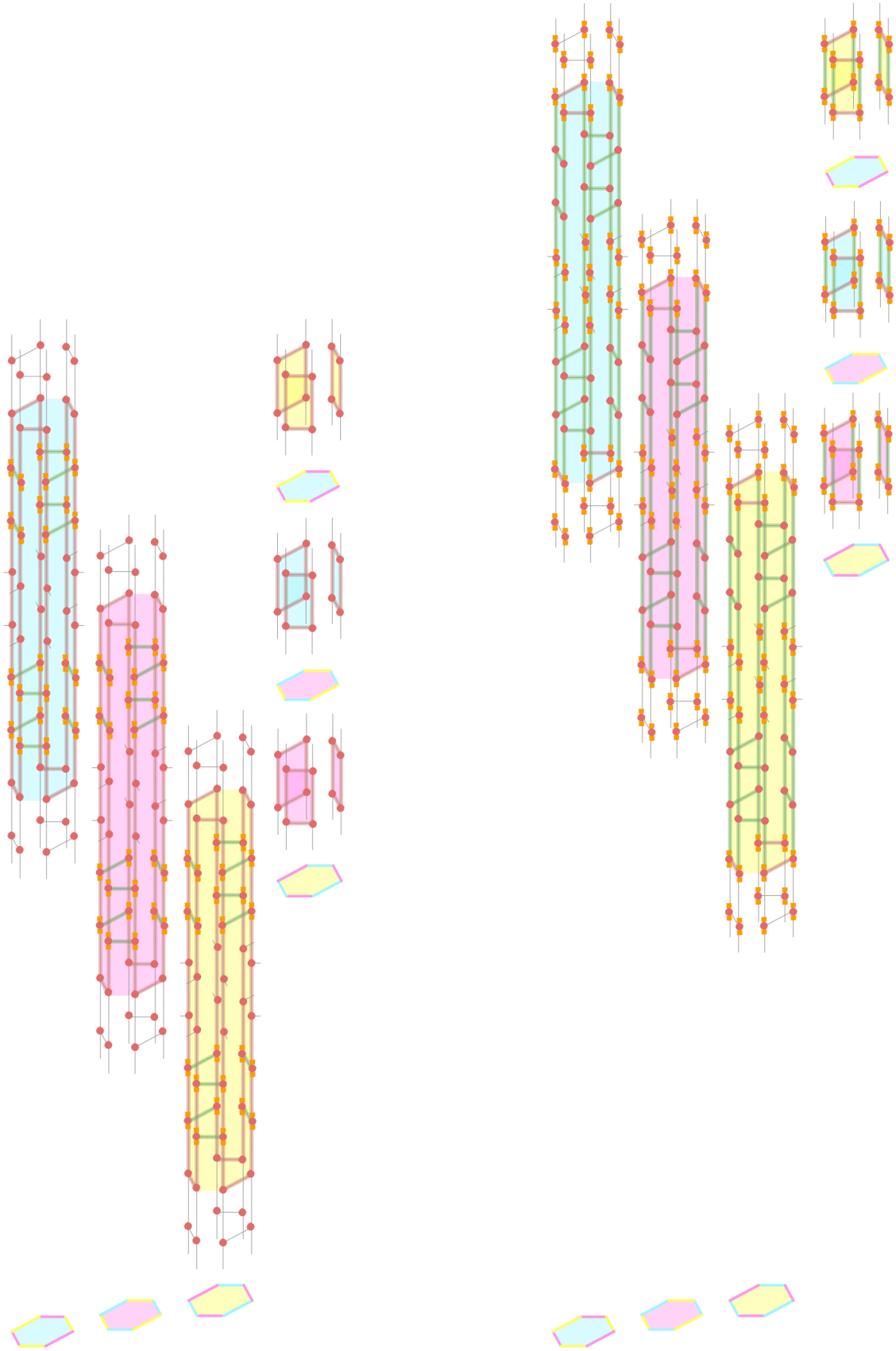


**Figure 30:** $E$ (left) and $M$ (right) detectors of the $X^2Z^2$ code.