

Optimization Tools for Distance-Preserving Flag Fault-Tolerant Error Correction


Balint Pato^{1,2,*}, Theerapat Tansuwannont^{1,2,†,§}, Shilin Huang^{1,2,||} and
Kenneth R. Brown^{1,2,3,4,‡}

¹*Duke Quantum Center, Duke University, Durham, North Carolina 27701, USA*

²*Department of Electrical and Computer Engineering, Duke University, Durham, North Carolina 27708, USA*

³*Department of Physics, Duke University, Durham, North Carolina 27708, USA*

⁴*Department of Chemistry, Duke University, Durham, North Carolina 27708, USA*

 (Received 22 August 2023; revised 25 January 2024; accepted 2 April 2024; published 16 May 2024)

Lookup-table decoding is fast and distance preserving, making it attractive for near-term quantum computer architectures with small-distance quantum error-correcting codes. In this work, we develop several optimization tools that can potentially reduce the space and time overhead required for flag fault-tolerant quantum error correction (FTQEC) with lookup-table decoding on Calderbank-Shor-Steane (CSS) codes. Our techniques include the compact lookup-table construction, the meet-in-the-middle technique, the adaptive time decoding for flag FTQEC, the classical processing technique for flag information, and the separate X - and Z -counting technique. We evaluate the performance of our tools using numerical simulation of hexagonal color codes of distances 3, 5, 7, and 9 under circuit-level noise. Combining all tools can result in an increase of more than an order of magnitude in the pseudothreshold for the hexagonal color code of distance 9, from $(1.34 \pm 0.01) \times 10^{-4}$ to $(1.43 \pm 0.07) \times 10^{-3}$.

DOI: [10.1103/PRXQuantum.5.020336](https://doi.org/10.1103/PRXQuantum.5.020336)

I. INTRODUCTION

Inside a future large-scale quantum computer, there will be a continuous battle against unwanted interactions with the environment. The main goal of fault-tolerant quantum error-correction (FTQEC) protocols [1] is to create a robust channel to transfer quantum information from the past to the future. The threshold theorem states that it is possible to suppress the failure rate of this channel (the logical error rate) to an arbitrarily small value given that the physical error rate of the constituent operations are below the accuracy threshold [2–7]. It is essential to reduce both space and time overhead (the numbers of qubits and gates) for scalable quantum computing, as decreasing logical error

rates requires increasing overhead [8–11], and the current leading proposals for FTQEC schemes have daunting requirements [12,13].

An FTQEC scheme is designed to be robust against propagating errors that emerge from faulty gates during the execution of the protocol [1]. The scheme also has to protect against ancilla preparation and measurement errors, usually through repeated syndrome measurements. For an $[[n, k, d]]$ stabilizer code [14], which encodes k logical qubits into n physical qubits and has minimum distance d , Shor's solution [1] has been to utilize a cat-state ancilla register that requires w ancilla qubits and $(d+1)^2/4$ rounds of syndrome measurements, where w is the maximum weight of the stabilizer generators. In Steane-style syndrome extraction [15], the ancilla register requires n qubits and is encoded with the same quantum error-correcting code (QECC) as the data qubits. Similarly, in Knill-style error correction [16], the ancilla register consists of two blocks of n qubits encoded in the same QECC as the data qubits.

In contrast to complex ancilla structures, bare ancillas can also be used to fault-tolerantly extract the syndrome while preserving the minimum distance for some specific families of stabilizer codes [17–19] and subsystem codes [20–22] or by tolerating some loss of distance [23–25]. For a general stabilizer code, however, generator measurements with bare ancillas might not be possible. A series of works aiming to reduce the size of the ancilla register has resulted in increasingly lighter-weight constructions

*balint.pato@duke.edu

†Corresponding author: t.tansuwannont.qiqb@osaka-u.ac.jp

‡ken.brown@duke.edu

§Present address: Center for Quantum Information and Quantum Biology, Osaka University, Toyonaka, Osaka 560-0043, Japan.

||Present address: Department of Applied Physics, Yale University, New Haven, Connecticut 06511, USA.

¶These authors contributed equally to this work.

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

[26,27], which have also led to the *flag FTQEC* schemes for perfect codes of distance 3 that use only two ancillas per generator [28], one flag qubit and one syndrome qubit. The flag schemes later generalized to arbitrary codes of distance d require $d + 1$ ancillas per generator [29], while the schemes for some specific families of codes require fewer [30–35].

FTQEC schemes based on extraction circuits with cat states and flag FTQEC schemes both require repetition of syndrome measurements that can result in a large number of gates. Adaptive syndrome-measurement schemes in which the subsequent measurement procedures depend on the previous syndrome-measurement outcomes have been explored to reduce the number of rounds required for FTQEC schemes with Shor-type extraction circuits [36–38].

During the execution of an FTQEC protocol, faults can occur at any gate on any round of the syndrome measurements. The only information about the error on data qubits that we can obtain is a sequence of error syndromes and we want to find an appropriate recovery operator from this information. An ideal strategy would be to use all syndrome bits from all rounds, i.e., the whole measurement outcomes in space-time. For some codes with a nice structure, such as surface codes, an efficient space-time decoder exists [39]. However, constructing a space-time decoder for a general stabilizer code is not simple. To simplify the problem, we will consider an error decoder, which is composed of two parts: the space and the time decoders. Under the assumption that the syndrome measurements can be faulty, the time decoder finds a round of syndrome measurements that has no faults and gives a correct syndrome. The space decoder then uses the correct syndrome to construct a recovery operator.

Conventionally, flag FTQEC uses a lookup-table decoder as a space decoder and relies on Shor-style repeated syndrome measurements as a time decoder, although there are instances in which this is not the case [40]. These decoders have pros and cons. The lookup-table decoder is fast and distance preserving. However, building a lookup table requires an exhaustive search over all possible fault combinations up to a certain number of faults and the table requires a lot of memory to store. Thus, it might not work well with a code of high distance (unless code concatenation is applied). The Shor-style time decoder is simple and compatible with any space decoder. However, the large time overhead required in the repetition can result in a lower threshold.

In this work, we build several optimization tools for both space and time decoders for the purpose of reducing the overhead of both to obtain better-performing protocols for flag-qubit-based FTQEC. Most of our tools are applicable to general stabilizer codes but we primarily focus on self-orthogonal CSS codes (CSS codes in which X - and Z -type generators are of the same form) in which

the number of physical qubits is odd, the number of logical qubits is 1, and the logical X and Z operators are transversal for simplicity. Our main results are the following. (1) We develop a technique to build a lookup table more efficiently. Our compact lookup table can leverage the structure of a self-orthogonal CSS code and requires 87.5% less memory footprint compared to a lookup table designed for a generic stabilizer code. Our method also efficiently verifies whether a configuration of the flag circuits preserves the code distance. The development also leads to the notion of a *fault code*, which can be useful in error sampling for the circuit-level noise model. (2) We introduce the meet-in-the-middle (MIM) technique, which can help the lookup-table decoder correct faults when the number of faults occurring is more than the number of errors correctable by the underlying code. Although the correction is not always successful, the higher success probability can significantly increase the pseudothreshold in our simulations. (3) We generalize previous work [38] on adaptive syndrome-measurement schemes to flag FTQEC and introduce one-tailed and two-tailed adaptive time decoders, which are useful in different circumstances. We also develop a classical processing technique on flag information that makes our FTQEC protocols compatible with any fault-tolerant Clifford computation. (4) We use our optimization tools and perform numerical simulations on the hexagonal color codes [41] of distances 3, 5, 7, and 9. The results show that each of our tools can significantly reduce the logical error rates and increase the pseudothreshold for each code while preserving the code distance. For the hexagonal color code of distance 9, the pseudothreshold is improved by one order of magnitude, from $(1.34 \pm 0.01) \times 10^{-4}$ to $(1.43 \pm 0.07) \times 10^{-3}$, when all techniques are applied.

This paper is organized as follows. In Sec. II, we define the noise model in this work, review flag FTQEC, and provide definitions of fault-tolerant error correction. In Sec. III, we develop optimization tools for space decoder, including an efficient method to build a compact lookup table and the MIM technique. In Sec. IV, we develop optimization tools for time decoder, including the one-tailed and two-tailed adaptive time decoder, and other extended techniques for CSS codes. In Sec. V, we provide numerical results for the hexagonal color codes and observe the effects of the MIM, the adaptive time decoding, and the separate X - and Z -counting techniques on the logical error rates. We discuss and conclude our results in Sec. VI.

II. BACKGROUND

Quantum systems are fragile and can lose their properties easily when interacting with the environment. To protect quantum information, one can use a QECC to encode the quantum data. Quantum error correction (QEC) is a process that identifies an error when it occurs and

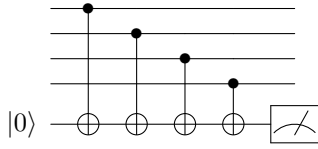


FIG. 1. A syndrome-extraction circuit with bare ancilla for measuring a stabilizer generator of the form $ZZZZ$.

then applies an appropriate error-correction (EC) operator to remove the error. However, quantum operations in the process can be faulty and may introduce more errors to the system. For this reason, we want to make sure that the QEC process is *fault tolerant*, which provides robustness guarantees against the impact of noise on the error-correction implementations.

In this section, we first describe the noise model that will be used in this work and provide the conventional definition of fault-tolerant error correction in Sec. II A. We then review flag FTQEC and provide a revised definition of fault-tolerant error correction, which is more suitable for flag FTQEC in Sec. II B.

A. Noise model and conventional definition of fault-tolerant error correction

An $[[n, k, d]]$ stabilizer code [14] encodes k logical qubits using n physical qubits and can correct up to $\tau = \lfloor (d-1)/2 \rfloor$ errors, where d is the code distance. A stabilizer code is described by a *stabilizer group*, an Abelian group the elements of which are called stabilizers which is generated by $r = n - k$ commuting Pauli operators. The code space is the simultaneous $+1$ eigenspace of all elements in the stabilizer group.

The QEC process for a stabilizer code can be done by first measuring the eigenvalues of all stabilizer generators. An r -bit string of measurement outcomes is called the error syndrome (where bits 0 and 1 refer to $+1$ and -1 eigenvalues of each generator). An example of a circuit for measuring an eigenvalue of a stabilizer generator is displayed in Fig. 1. After the syndrome is obtained, an appropriate recovery operator will be found by a mapping called the *error decoder*. Finally, the recovery operator will be applied to the data qubits. For Calderbank-Shor-Steane (CSS) codes [15,42], it is possible to correct X - and Z -type errors separately. In this work, we follow *standard CSS decoding* [43], meaning independent recovery for X - and Z -type errors, thus not taking the effect of X/Z correlations such as Y errors into account.

If all gates in the syndrome-measurement process (with an example circuit in Fig. 1) are perfect, a stabilizer code of distance d should be able to correct up to τ errors as desired. However, the above process may not be fault tolerant under the circuit-level depolarizing noise. This is because a single faulty gate may lead to an error that

can propagate to multiple errors on the data qubits, often referred to as hook errors [39]. These errors can always be handled by complex ancilla [1,44,45] or flag circuits [28] and sometimes handled by the circuit order [17–19].

In this work, we use the *circuit-level depolarizing noise model*. After each gate, a *fault* occurs on the support of the gate. Every single-qubit gate is followed by a single-qubit Pauli operator $P \in \{X, Y, Z\}$ with probability $p/3$ each and every two-qubit gate is followed by a two-qubit Pauli operator $P_1 \otimes P_2 \in \{I, X, Y, Z\}^{\otimes 2} \setminus \{I \otimes I\}$ with probability $p/15$ each. In addition, a single-qubit preparation and measurement can also be faulty; this is modeled by a bit-flip channel after a single-qubit preparation or before a single-qubit measurement with error probability p .

One way to define FTQEC is to use the definition proposed by Aliferis, Gottesman, and Preskill.

Definition 1 (Fault-tolerant error correction [6]). Let $t \leq \lfloor (d-1)/2 \rfloor$, where d is the distance of a stabilizer code. An error-correction protocol is t -*fault tolerant* if the following two conditions are satisfied:

- (1) The error-correction correctness property (ECCP): for any input code word with an error of weight r , if s faults occur during the protocol with $r + s \leq t$, ideally decoding the output state gives the same code word as ideally decoding the input state.
- (2) The error-correction recovery property (ECRP): if s faults occur during the protocol with $s \leq t$, regardless of the weight of the error on the input state, the output state differs from any valid code word by an error of weight at most s .

When a QEC protocol satisfies Definition 1, it is guaranteed that the output error will have weight $\leq t$ whenever the weight of the input error plus the total number of faults in the protocol is $\leq t$. This means that if the next round of QEC has no faults, it can always correct the output error from the current round. Normally, we would like to construct an FTQEC protocol in which t is as close as possible to $\tau = \lfloor (d-1)/2 \rfloor$. If $t = \tau$, we say that the *FTQEC protocol preserves the code distance*.

B. Flag technique and revised definition of fault-tolerant error correction

Before describing the flag technique for FTQEC, let us consider a well-known Shor FTQEC [1] applied to a stabilizer code of distance d . In this scheme, a stabilizer generator of weight w is measured using a cat state of the form $\frac{1}{\sqrt{2}}(|0\rangle^{\otimes w} + |1\rangle^{\otimes w})$ and transversal controlled-NOT (CNOT) gates (see Fig. 2). A circuit of this kind will be called a *Shor-syndrome-extraction circuit*. When the cat state is prepared fault-tolerantly, a single fault in the circuit can lead to an error of weight no more than one on the data qubits, so the set of all possible errors arising from up

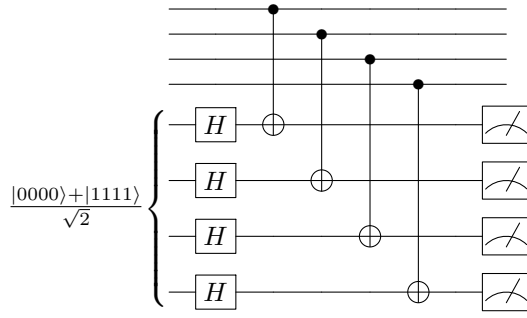


FIG. 2. A Shor-syndrome-extraction circuit for measuring a stabilizer generator of the form $ZZZZ$.

to t faults is exactly the same as a set of all possible errors on $\leq t$ qubits in this case. Therefore, any syndrome can uniquely identify the error (up to a multiplication of some stabilizer) when the number of faults in the protocol is $\leq t$.

One drawback of the Shor-syndrome-extraction circuit is that the number of required ancilla qubits is equal to the maximum weight of the stabilizer generators. Also, fault-tolerant preparation of the ancilla cat state requires verification [1] or an Divincenzo-Aliferis ancilla decoding circuit [26], which requires additional space and time overhead. One possible technique that can reduce the number of required ancillas for FTQEC is the flag technique [28], in which each syndrome-extraction circuit uses one ancilla qubit to keep the syndrome-measurement outcome and a few flag ancillas to find a location that a fault might have occurred. A circuit of this kind will be called a *flag circuit* (for an example, see Fig. 3). The flag-measurement outcomes give extra information that can be used to partition set of all possible errors from a certain number of faults. Therefore, it is possible to distinguish between two nonequivalent errors that correspond to the same syndrome if the flag-measurement outcomes associated with each error are different, making error correction easier.

Here, we define fault combination, the fault set, and the distinguishability of a fault set as follows.

Definition 2 (Fault combination, combined data error, and cumulative flag vector [35]). A fault combination

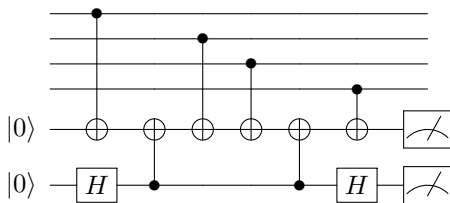


FIG. 3. A flag circuit for measuring a stabilizer generator of the form $ZZZZ$.

$\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_r\}$ is a set of r faults $\lambda_1, \lambda_2, \dots, \lambda_r$. Suppose that the Pauli error due to the fault λ_i can propagate through the circuit and lead to *data error* $E(\lambda_i)$ and *flag vector* $\vec{f}(\lambda_i)$. The *combined data error* $\mathbf{E}(\Lambda)$ and *cumulative flag vector* $\vec{F}(\Lambda)$ corresponding to the fault combination Λ are

$$\mathbf{E}(\Lambda) = \prod_{i=1}^r E(\lambda_i), \quad (1)$$

$$\vec{F}(\Lambda) = \sum_{i=1}^r \vec{f}(\lambda_i) \pmod{2}. \quad (2)$$

Definition 3 (Distinguishable fault set [35]). Let \mathcal{S} be the stabilizer group of a stabilizer code and let the *fault set* \mathcal{F}_t denote the set of all possible fault combinations arising from up to t faults during the measurement of stabilizer generators of \mathcal{S} . We say that \mathcal{F}_t is *distinguishable* if for any pair of fault combinations $\Lambda_p, \Lambda_q \in \mathcal{F}_t$, at least one of the following conditions is satisfied:

- (1) $\vec{s}(\mathbf{E}(\Lambda_p)) \neq \vec{s}(\mathbf{E}(\Lambda_q))$, or
- (2) $\vec{F}(\Lambda_p) \neq \vec{F}(\Lambda_q)$, or
- (3) $\mathbf{E}(\Lambda_p) = \mathbf{E}(\Lambda_q)M$ for some stabilizer $M \in \mathcal{S}$,

where $\vec{s}(\mathbf{E})$ is the error syndrome of a combined error \mathbf{E} . Otherwise, we say that \mathcal{F}_t is *indistinguishable*.

Note that the cases of faulty flag-qubit measurements are included when the fault set is calculated for verifying fault set distinguishability (see Sec. III A). Having a distinguishable fault set is key to successful error decoding. Given a set of syndrome-extraction circuits (with or without flags), we can calculate the fault set \mathcal{F}_t and check whether it is distinguishable. If it is, all possible errors arising from up to t faults that correspond to the same syndrome and cumulative flag vector are always logically equivalent. Therefore, if the syndrome measurements give a syndrome \vec{s} and a cumulative flag vector \vec{F} , we can pick any error that corresponds to the pair (\vec{s}, \vec{F}) to be a recovery operator. Using this idea, a decoding table and an FTQEC protocol can be constructed.

With the notion of fault distinguishability, it is possible to further generalize the definition of FTQEC as follows.

Definition 4 (Fault-tolerant error correction (revised) [35]). Let $t \leq \lfloor (d-1)/2 \rfloor$, where d is the distance of a stabilizer code. An error-correction protocol is *t-fault tolerant* if the following two conditions are satisfied:

- (1) ECCP: for any input code word with an error that can arise from r faults before the protocol and corresponds to the zero cumulative flag vector, if s

faults occur during the protocol with $r + s \leq t$, ideally decoding the output state gives the same code word as ideally decoding the input state.

- (2) ECRP: if s faults occur during the protocol with $s \leq t$, regardless of the number of faults that can cause the input error, the output state differs from any valid code word by an error that can arise from s faults and corresponds to the zero cumulative flag vector.

The main difference between these two definitions of FTQEC is that Definition 4 considers the *number of faults* that can cause the input (or the output) error instead of the *weight* of the error. An FTQEC protocol satisfying Definition 4 can be constructed if we can find syndrome-extraction circuits that give a distinguishable fault set (for more details, see previous results [35] by one of the authors of this work and the discussion in Sec. III). In fact, while the threshold theorem proved by Aliferis *et al.* [6] relies on the weight of the error to define fault tolerance (Definition 1), the theorem has been shown to hold [35] even if the definition of fault tolerance uses the number of faults (Definition 4) instead. For flag FTQEC, the use of Definition 4 can result in simpler FTQEC protocols, so we will use Definition 4 in the protocol development throughout this work.

III. OPTIMIZATION TOOLS FOR SPACE DECODING

In this work, the term “space decoder” refers to a process that finds a recovery operator from a given syndrome under the assumption that it is exactly the same as the syndrome of an error that occurred to the code word. The decoder succeeds if multiplying the error and the recovery operator gives a trivial logical operator (a stabilizer) and it fails if the multiplication gives a nontrivial logical operator. Our goal is to develop a space decoder such that whenever the total number of faults in the whole protocol is $\leq t$, the decoder always succeeds. In this work, we are interested in a lookup-table-based space decoder for flag FTQEC, so the decoder will use both syndrome and flag information obtained during the syndrome measurements. Note that the ability to correct faults for a certain code depends on the structure of the circuits for syndrome extraction, such as the ordering of gates.

In this section, we develop optimization tools for space decoding. In Sec. III A, we discuss how to efficiently construct a lookup table for error decoding for a distinguishable fault set \mathcal{F}_t and introduce the notion of a fault code. In Sec. III B, we discuss the meet-in-the-middle technique, an additional technique that can help improving our space decoders for both codes and increase the accuracy of the decoding.

A. Compact lookup table for minimum-weight decoding and fault code

In this section, we discuss how to construct the fault set \mathcal{F}_t , verify its distinguishability, and construct the lookup table for error decoding. With our method, we can reduce the memory-footprint requirement of the lookup table by 87.5% for self-orthogonal CSS codes compared to a lookup table designed for generic stabilizer codes. We also present the framework of *fault codes* that enables fast construction using streamlined Pauli-frame simulation represented as matrix-algebra operations over GF(2).

A brief summary of our methods is as follows. Let the *weight of a fault combination* be the number of faults that give rise to the fault combination. The decoding table maps each full syndrome, (\vec{s}, \vec{F}) , to a recovery operator that corresponds to the combined data error of the minimum-weight fault combination that results in the full syndrome. To construct the decoding table, we start by collecting all weight-1 fault combinations that may arise in the extraction circuits. We map each resulting full syndrome to its corresponding data error. At this point, we say that the *search radius* of the lookup table is 1. Afterward, we combine pairs of weight-1 fault combinations to create all possible weight-2 fault combinations. The combined data error of each weight-2 fault combination is obtained by simply taking the product of the data errors and the full syndrome is obtained by adding full syndromes of the weight-1 fault combination modulo 2. If the combining process leads to a new syndrome, we store it in the table. If the process leads to an existing syndrome, we have a collision and do one of the following. (1) If the stored combined data error and the new combined data error are the same up to a stabilizer, then we do nothing. (2) If the stored combined data error and the new combined data error differ by a logical operator (up to a stabilizer), then we raise an error; this implies that \mathcal{F}_2 is not distinguishable. At this point, if there is no combination that causes the second case (i.e., \mathcal{F}_2 is distinguishable), we say that the search radius of the lookup table is 2. We can gradually increase the search radius using similar ideas until we reach the maximum search radius in which the fault set is distinguishable. Here, we rely on an efficient representation of the combined data errors using a decomposition of Pauli operators to pure errors, stabilizers and logical operators [46].

During sampling, the decoder receives a full syndrome that has been measured. When the decoder finds this syndrome in the lookup table, it returns the corresponding *actual recovery operator* (ARO). However, when the decoder cannot find the syndrome in the lookup table, it only returns a so-called *canonical recovery operator* (CRO). Each syndrome has a unique canonical recovery operator, which guarantees that applying such an operator to the erroneous encoded state will map it back to the code space but with a possible logical error.

The full description of our methods is presented below.

1. Reducing the memory footprint

To decode an $[[n, k, d]]$ stabilizer code, we can construct a lookup table that, for all possible fault combinations of weight 0 to t (where $t = \lfloor (d-1)/2 \rfloor$), stores the full syndrome $\vec{\sigma} = (\vec{s}, \vec{F})$ as the key and maps the combined data error as the recovery operator. While this approach works, it is expensive. Let T_{stab} denote the number of distinct full syndromes for the fault combinations of weight 0 to t for a generic stabilizer code. As T_{stab} and thus the size of the lookup table grows exponentially in n , $n-k$ (the number of generators), and the number of circuit locations, we want to choose a representation to store data as efficiently as possible. For general stabilizer codes, $n-k$ bits are required for the syndrome bits and $n-k$ bits for the cumulative flag vector (assuming flag circuits with a single flag ancilla for simplicity). Meanwhile, the recovery operator requires $2n$ bits using the symplectic representation. Thus we have $T_{\text{stab}}(4n-2k)$ bits of data in the map.

Leveraging the structure of CSS codes, we can significantly improve the memory footprint. Assuming standard CSS decoding in which two separate lookup tables are used for X and Z decoding, we denote by r_X and r_Z the number of X - and Z -type stabilizer generators satisfying $r_X + r_Z = n-k$. The per-entry cost decreases, as the entries only need to cater for X - or Z -type operators and syndromes. Each entry for the X - and Z -type syndromes will have $2r_X$ and $2r_Z$ bits, respectively, for the syndrome and the cumulative flag vector and n bits for the recovery operator. A self-orthogonal CSS code needs only one table, further decreasing the cost (see more details on the total number of bits in Appendix A). Moreover, we can reduce the number of bits for the recovery operator to k using the following two key ideas:

- (1) In general, for an $[[n, k, d]]$ code, each Pauli operator $P \in \mathcal{P}_n$ can be decomposed as a product $P = EML$ of a pure error E , a stabilizer $M \in \mathcal{S}$, and a logical operator $L \in \overline{\mathcal{P}}_k$ (where $\overline{\mathcal{P}}_k$ is the k -dimensional logical Pauli group) [46]. We define a fixed set of pure errors called *canonical recovery operators* (CROs), with one CRO for each unique syndrome \vec{s} .
- (2) Given a syndrome $\vec{s}(E)$, the goal of decoding is to find a recovery operator R such that $RE \in \mathcal{S}$ and thus R converts the error into the logical identity operation. For any possible Pauli error, we only have to store its *logical class*, a value that indicates how the error is related to a CRO with the same syndrome. This enables the map value to be only $2k$ bits of information in general and k bits in the case in which the code is a self-orthogonal CSS code. In this latter case and with $k=1$, the logical class is 0 if the multiplication of the Pauli operator and

the CRO with the same syndrome is in the stabilizer group; otherwise, the logical class is 1.

Altogether, for a self-orthogonal CSS code with $n \gg k$, the size of the table can be as small as 12.5% of the table if we have viewed the code as a generic stabilizer code and stored the full recovery operators instead of the logical classes. For a CSS code that is not self-orthogonal, the gain is smaller but still significant (for detailed calculations around savings for the lookup table, See Appendix A). Note that if the lookup table is used for proving distinguishability, all unique syndromes are required. However, in a real-time decoding architecture, the entries corresponding to significantly low-probability fault combinations may be excluded, resulting in further reduction [47].

2. Constructing the lookup table

We now explicitly describe an algorithm to construct the lookup table. During the construction of the lookup table, we have a systemic way to enumerate fault combinations with their full syndromes and combined data errors instead of running through a circuit simulator for each case. The exhaustive enumeration of all possible fault combinations of weight 0 to t is done in two steps. First, we enumerate the single faults and capture their full syndrome and logical class in a single column of the *fault-check matrix*, H_f , using matrix algebra over $\text{GF}(2)$ to represent the propagation of errors in our syndrome-extraction circuits. Second, we combine these columns in all possible combinations of 0 to t faults ($\sum_{i=0}^t \binom{N}{i}$ combinations in total, where N is the number of possible single faults) while keeping track of the weight of each fault combination. This last step verifies whether \mathcal{F}_t is distinguishable (which is equivalent to verifying whether the protocol is distance preserving) and at the same time builds a lookup table for the decoder.

Enumerating weight-1 faults. From here on, we will only consider a self-orthogonal CSS code, and denote its parity-check matrix H . In order to list all possible single faults under the circuit-level depolarizing noise model, it is sufficient to consider all possible weight-1 faults within a single round of syndrome measurements. Each column of the fault-check matrix H_f describes, for each possible weight-1 fault, what its full syndrome and its logical class are. As the logical class of each fault depends on how its CRO is defined, we define the fault-check matrix relative to the right inverse H^{-1} of H (for which $HH^{-1} = I_{(n-k)/2}$).

The high-level structure of H_f consists of three major groups of rows and three major groups of columns. The three groups of rows are the $(n-k)/2$ *generator bits*, the $(n-k)/2$ *flag bits*, and the k bits for the logical class. Each single fault that is represented by a column of H_f can be put into one of the following three categories:

- (1) *Pure data-qubit errors* that result only in generator bits. They do not trigger flags, resulting in all-zero flag bits. The CRO R of each pure data-qubit error E can be described by each column of $H^{-1}H$ (since the syndromes of CROs are $H(H^{-1}H) = (HH^{-1})H = H$); thus the product RE of each E can be described by each column of $I_n \oplus H^{-1}H$ (where the matrix addition, denoted by \oplus , and multiplication are over $\text{GF}(2)$). If E is an X -type (or a Z -type) error, the logical class of RE is described by a k -bit string in which the i th bit indicates whether RE anti-commutes with \bar{Z}_i (or \bar{X}_i). That is, the logical classes of all pure data-qubit errors are described by

$$\begin{pmatrix} J_1^T(I_n \oplus H^{-1}H) \\ J_2^T(I_n \oplus H^{-1}H) \\ \vdots \\ J_k^T(I_n \oplus H^{-1}H) \end{pmatrix},$$

where J_i is the column vector representing \bar{Z}_i (or \bar{X}_i).

- (2) *Flag-ancilla-preparation or -measurement errors* that do not propagate to data qubits; thus, each single-flag error will result in a single flag bit. Therefore, all errors of this type have the all-zero syndrome and logical class 0, while the flag bits can be easily represented by the $(n-k)/2 \times (n-k)/2$ identity matrix.
- (3) *Gate faults* that cause errors on the syndrome ancilla that can propagate to data and flag qubits—we order these faults by top-down and left-right place of occurrence and capture their effect in syndrome bits, flag bits, and logical class. The part of the effective matrix corresponding to this type of fault is denoted by $H_{f,\text{gate}}$.

Note that single measurement and reset errors on the syndrome ancilla are ignored during this analysis, as their effects would be removed by the time decoder through the repetition of syndrome measurements.

Generalizing H_f for a non-self-orthogonal CSS code is straightforward. In that case, the parity-check matrices for X - and Z -type errors can be different, leading to different fault-check matrices. Generalizing H_f for a generic stabilizer code is more complicated but still doable, as all operators must be considered in the symplectic form. In that case, the number of rows for the logical class is $2k$. Also, instead of taking the inner product with J_i , whether each CRO commutes or anticommutes with each logical operator can be determined by the symplectic inner product between the symplectic bit strings representing the CRO and the logical operator.

In the case in which the code is a self-orthogonal CSS code, n is odd, $k = 1$, and the logical X and logical Z

operators are transversal, the fault-check matrix is

$$H_f = \left(\begin{array}{c|c|c} H & 0 & \\ \hline 0 & I_{(n-1)/2} & H_{f,\text{gate}} \\ \hline J^T(I_n \oplus H^{-1}H) & 0 & \end{array} \right),$$

where J is the all-one column vector of length n (representing $X^{\otimes n}$ or $Z^{\otimes n}$).

As an example, consider the first group of columns for the $[[7, 1, 3]]$ Steane code [15], the stabilizer generators of which can be defined by the parity-check matrix

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

One can pick its right inverse H^{-1} as follows:

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

We can see that each column of H^{-1} gives a Pauli operator for each syndrome bit. For a data error E of any weight, the syndrome $\vec{s}(E) = HE$ can be recovered with the CRO defined by $R(\vec{s}(E)) \equiv H^{-1}\vec{s}(E)$ as

$$\begin{aligned} \vec{s}(E \oplus R) &= H(E \oplus R) \\ &= HE \oplus HH^{-1}HE \\ &= HE \oplus HE = 0. \end{aligned}$$

For example, for $E = (0110000)^T$, $\vec{s}(E) = (001)^T$ and $R(\vec{s}(E)) = (1000000)^T$; thus $RE = (1110000)^T$, for which the syndrome is trivial (as RE is a logical operator).

For errors of weight 1 on the data qubits, the operator RE of each error can be represented by each column of $I_n \oplus H^{-1}H$. Since the logical class of RE can be determined by its weight parity, the logical classes of this type of errors are the row of $L \equiv J^T(I_n \oplus H^{-1}H)$, where J is the all-one column vector. That is, for the Steane code, the part of H_f corresponding to pure data-qubit errors is

$$\left(\begin{array}{c|c} H & \\ \hline 0 & \\ \hline J^T(I_n \oplus H^{-1}H) & \end{array} \right) = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Constructing $H_{f,\text{gate}}$. In this work, we focus on the case that any Z -type or X -type stabilizer generator of weight w is

measured using a flag circuit with a single flag ancilla similar to the circuit in Fig. 4 (with a slight modification, a similar construction can also be made for a general flag circuit). For $H_{f,\text{gate}}$, we are interested in how the errors propagate from the syndrome ancilla to the data qubits and the flag ancilla. The error propagation is represented via binary matrices, an idea closely related to the “gate matrix,” where the direction of propagation is the opposite way, toward the ancilla from the data qubits [45,48]. Given single-flag syndrome-extraction circuits for all stabilizer generators and the CNOT ordering for each circuit, $H_{f,\text{gate}}$ can be calculated via the *propagator matrix* P and the *aggregator matrix* A , defined as follows. For the error-correction protocol with n data qubits and r flag bits (which is the same number as the number of X - or Z -stabilizer generators), The matrix P has $n + r$ rows. The number of columns of P is $\sum_{i=1}^r (w(g_i) + 2)$, where $w(g_i)$ is the Hamming weight of the i th stabilizer generator (g_i). This is from the fact that for each CNOT gate in the single-flag syndrome-extraction circuits, the only fault that can lead to a unique data error after propagation is the fault that leads to a single Z error on the target qubit of the CNOT (which is the syndrome ancilla). To simplify the construction, we construct a submatrix P_i of size $(n + r) \times (w(g_i) + 2)$ for each row g_i of H (i.e., each stabilizer generator) and then concatenate the submatrices to obtain

$$P = (P_1 P_2 \dots P_r). \quad (3)$$

As the order of the CNOT gates matters in subtle ways, for a given stabilizer generator g_i , we represent the CNOT ordering by the permutation $\pi_i : \{1, 2, \dots, w(g_i)\} \rightarrow \text{supp}(g_i)$, where $\pi_i(j)$ indicates the control (data) qubit of the j th CNOT (the target qubit is always the syndrome ancilla). π_i

can also be represented by a list. For example, two possible permutations of CNOT gates in the syndrome-extraction circuit for measuring g_1 of the $[[7, 1, 3]]$ Steane code are $\pi_1 = [4, 5, 6, 7]$ and $\pi_1 = [4, 6, 5, 7]$.

To construct P_i , we iterate from $j = 1$ to $w(g_i)$ and create a column for each iteration with all zeros except for the 1 in row $\pi_i(j)$. We then insert an all-zero column on the second from the left and the second from the right positions (which represent the flag CNOT gates) and set its value to 1 at row $n + i$. In our running example of $g_1 = (0001111)$, for a permutation of $\pi_1 = [4, 6, 5, 7]$,

$$P_i = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \quad (4)$$

The aggregator matrix A plays the role of propagating the errors to the end of the syndrome-measurement circuits. For each g_i , we define A_i to be a square matrix of size $(w(g_i) + 2) \times (w(g_i) + 2)$ having a lower triangle set to all ones and define $A = \bigoplus_{i=1}^r A_i$ to be the direct sum of all the A_i . In our example case of g_1 ,

$$A_i = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \quad (5)$$

Multiplying the propagator and the aggregator matrices yields

$$PA = \begin{pmatrix} \Omega \\ \Phi \end{pmatrix} = \begin{pmatrix} \Omega_1 & \Omega_2 & \dots & \Omega_r \\ \Phi_1 & \Phi_2 & \dots & \Phi_r \end{pmatrix}, \quad (6)$$

where columns of the submatrices Ω_i are the final Pauli operators and columns of the submatrices Φ_i are the cumulative flag vectors after measuring g_i and having a fault propagated from the syndrome ancilla to the data qubits at the location corresponding to the given column.

Next, we find the syndromes for these Pauli operators by multiplying them with the parity-check matrix,

$$S = H\Omega. \quad (7)$$

Then, for each syndrome, we define the CRO based on the right inverse H^{-1} ,

$$\Theta = H^{-1}S. \quad (8)$$

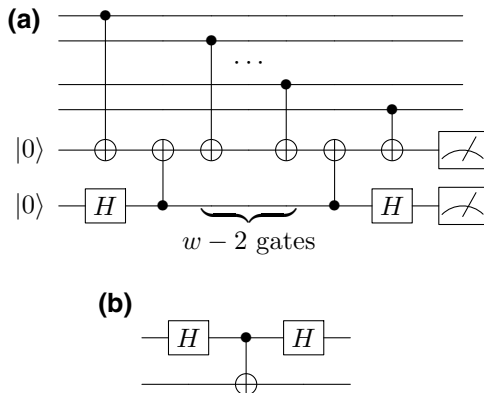


FIG. 4. (a) A flag circuit for measuring a Z -type stabilizer generator of weight w in this work. A flag circuit for measuring a X -type stabilizer generator of weight w can be obtained by replacing each CNOT gate that connects the data qubit to the syndrome ancilla with the gate in (b).

Finally, we determine the logical class L for each of the faults by adding the parity of the CRO and the propagated final data error,

$$L = J^T(\Omega \oplus \Theta). \quad (9)$$

As a result, the part of H_f corresponding to the gate faults is

$$H_{f,\text{gate}} = \begin{pmatrix} S \\ \Phi \\ L \end{pmatrix} = \begin{pmatrix} H\Omega \\ \Phi \\ J^T(I_n \oplus H^{-1}H)\Omega \end{pmatrix}. \quad (10)$$

The relationship between the full syndrome, the data error, and the CRO of each fault is as follows. Suppose that the i th column of H_f (which represents a single fault on the i th location) contains error syndrome \vec{s}_i , flag vector \vec{f}_i , and logical class l_i . The CRO of the fault is $H^{-1}\vec{s}_i$, while the data error of the fault is $l_i J \oplus H^{-1}\vec{s}_i$. That is, in the case of a single fault, the *actual recovery operator* (ARO) that we need to apply when finding the full syndrome (\vec{s}_i, \vec{f}_i) is $l_i J \oplus H^{-1}\vec{s}_i$.

Verifying distinguishability and building the lookup table. The fault set \mathcal{F}_t is distinguishable if and only if there is no fault combination from up to $2t$ faults that gives a nontrivial logical operator with trivial full syndrome [35]. As the fault-check matrix already contains all the possible single faults, in the case of $t = 1$, we only need to extend the matrix by a column with all zeros (which represents zero faults) and check whether there is a pair of columns that are the same except for the logical class. If there is, the combined data errors of one or two faults add up to an undetectable logical operator, meaning that \mathcal{F}_1 is not distinguishable.

When $t \geq 2$, we populate the cache with the logical classes of higher-weight fault combinations by simply combining all possible fault combinations of lower-weight fault combinations while keeping track of the weights of the fault combinations. We describe the i th fault combination as a key-value pair $[(\vec{s}_i, \vec{F}_i) : (l_i, w_i)]$, where (\vec{s}_i, \vec{F}_i) is the full syndrome, l_i is the logical class, and w_i is the weight of the fault combination. Combining the i th and the j th fault combinations gives $[(\vec{s}_i \oplus \vec{s}_j, \vec{F}_i \oplus \vec{F}_j) : (l_i \oplus l_j, w_i + w_j)]$. As we aim to check whether \mathcal{F}_t is distinguishable, we fill up the cache by combining any pair of fault combinations that satisfy $w_i + w_j \leq t$. In the case in which the process gives the new key (the full syndrome) that already exists in the cache, we have a key conflict. This can be one of the following cases:

- (1) The new and the existing fault combinations have the same full syndrome and the same logical class but have different weights. In this case, we store the fault combination with smaller weight in the cache.

TABLE I. The metrics of the lookup table. The number of columns of the fault-check matrix counted in the first row results from the three-part structure of data errors, flag errors, and gate faults. These columns are not necessarily unique, as can be seen in the second row, which counts the number of unique columns. The time to verify distinguishability for the different codes on a single thread with our C++ code depends on the number of unique columns; hence the verification of the higher distance code takes longer than shorter ones. All timings are reported using Intel Xeon Gold 6226R 2.90-GHz processors. Some fault combinations have the same full syndrome; hence the cache size is smaller than the full number of fault combinations. The cache size in memory is reported from actual usage, including the overhead of the hash table implementation.

	[[7, 1, 3]]	[[19, 1, 5]]	[[37, 1, 7]]	[[61, 1, 9]]
Number of columns of H_f	28	88	181	307
Number of unique columns	20	62	128	218
Number of fault combinations	20	1953	349 632	93 263 997
Cache size	20	1587	262 500	67 166 572
Memory	≤ 1 kB	≤ 1 kB	≈ 50 MB	≈ 1.38 GB
Verification time	≤ 1 ms	≤ 1 ms	≈ 720 ms	≈ 58.9 s

- (2) The new and the existing fault combinations have the same full syndrome but have different logical classes. As the sum of weights of these two fault combinations is $\leq 2t$, we raise an error—there exists a fault combination from up to $2t$ faults that gives a nontrivial logical operator with trivial full syndrome, i.e., \mathcal{F}_t is not distinguishable.

If, at the end, we find that \mathcal{F}_t is distinguishable, we can construct a lookup table of search radius t from the cache as follows: for each key-value pair $[(\vec{s}_i, \vec{F}_i) : (l_i, w_i)]$ in the cache, we store a new key-value pair $[(\vec{s}_i, \vec{F}_i) : l_i J \oplus H^{-1}\vec{s}_i]$ in the lookup table (the weights are not necessary for decoding, though they might be useful for estimating the number of faults that causes the full syndrome). That is, $l_i J \oplus H^{-1}\vec{s}_i$ is the ARO for the full syndrome (\vec{s}_i, \vec{F}_i) . When performing error decoding, the ARO is applied if the full syndrome obtained from measurements is found on the lookup table; otherwise, the CRO ($H^{-1}\vec{s}_i$) is applied.

The lookup table can then be stored in an efficient binary format on disk or memory as needed. In Table I, we display the metrics related to the lookup-table decoder obtained by the above algorithm.

In summary, we perform an exhaustive search of fault combinations which gives us a lookup table with search

radius t ; this is equivalent to verifying the distinguishability of \mathcal{F}_t . If we can construct the lookup table with $t = \tau = \lfloor (d-1)/2 \rfloor$, we have a minimum-weight decoder that is distance preserving under the circuit-level depolarizing noise model. As a hash table requires $\mathcal{O}(1)$ amortized complexity for lookup, this decoder is also relatively fast for numerical simulations or real-time decoding compared to more complicated algorithms such as MaxSAT decoding [49], neural-network-based decoding [50], or the restriction decoder [51] with minimum-weight perfect-matching decoding [52], all of which have at least $\mathcal{O}(n)$ complexity. However, the table size scales exponentially in the number of qubits, locations, and stabilizer generators, and thus constructing the lookup table may be impractical for a code of high distance.

3. The fault code

Any CSS code can be defined by its parity-check matrix H , which maps a bit string representing a combination of errors on the data qubits to the error syndrome of the error combination. In the case of flag FTQEC, where the circuit-level noise model is considered, we can use similar ideas and define a *fault code* by the fault-check matrix H_f , which maps a bit string representing a combination of possible faults to the full syndrome of the fault combination (which includes the error syndrome of the combined data error and cumulative flag vector) and the logical class relative to the CRO for the syndrome. It should be noted that the distance of the fault code might be lower than the distance of the underlying CSS code; this depends on the syndrome-extraction circuits, which affect the distinguishability of the fault set. We can define the *effective distance* d_{eff} to be the minimum number of faults that can give a fault combination with a nontrivial logical operator and the trivial full syndrome. The number of faults t_{eff} that the fault code can correct is $t_{\text{eff}} = \lfloor (d_{\text{eff}} - 1)/2 \rfloor$ (this is the maximum number of t in which \mathcal{F}_t is distinguishable). If the effective distance and the code distance are equal, we say that the error-correction protocol is distance preserving. Calculation of the distance of classical codes can be done by determining the spark of the parity-check matrix H , which is known to be NP-hard, in general [53]. However, the spark algorithm does not work in the case of degenerate CSS codes, as it reports only the minimum weight of the stabilizers, which is a lower bound on the code distance [42]. Our algorithm described in this section can be viewed as a modified spark algorithm that uses the logical class information to calculate the distance of the code (based on H) and also the effective distance of the fault code (based on H_f).

The perspective of the fault code can also be useful to extend a technique frequently used for error sampling (in, e.g., QECSIM [54]) to the circuit-level noise model beyond the code capacity noise model (memory errors

only) and phenomenological noise model (both memory and measurement errors). Here, a randomly generated column vector of Hamming weight w now represents faults on w locations instead of errors on w qubits. Suppose that the vector \vec{v} represents the fault combination and $H_f \vec{v}$ gives the full syndrome $(\vec{s}(\vec{v}), \vec{F}(\vec{v}))$ and the logical class $l(\vec{v})$. In an error-correction simulation, the decoder can predict the recovery operator \vec{r} based on the full syndrome. We will find that the predicted recovery operator causes a logical error if and only if $l(\vec{v})$ and $l(\vec{r})$ differ.

In principle, this method can lead to a better sampling rate compared to running the full circuit simulation for each sample. However, one needs to be aware of the probability distribution when generating vectors representing the fault combinations, as each possible single fault might not occur at the same rate.

B. Meet-in-the-middle technique

If the fault set \mathcal{F}_t of each code is distinguishable, the flag FTQEC protocol can correct up to t faults with certainty. However, whenever $t+1$ or more faults occur, the error correction is not guaranteed; our decoder can either remove the error or cause a logical error on the encoded state. Although the probability of having $t+1$ or more faults is $\mathcal{O}(p^{t+1})$, being able to correct more cases of faults can lead to a higher pseudothreshold. In this section, we introduce the meet-in-the-middle (MIM) technique, which can help to correct errors in the case in which there are more than t faults in our FTQEC protocol. Note that this technique is general and could help any FTQEC protocol with a table-based decoder to correct faults in excess of its capability if the stabilizer code being used is not a perfect (or a perfect CSS) code.

The MIM technique is inspired by the bidirectional-search algorithm [55] to improve the table-based decoder previously discussed in Sec. II B (see also Sec. III A) in the case in which the decoder cannot find, in its lookup table, the full syndrome obtained from measurements. Consider the case in which the fault set \mathcal{F}_t is distinguishable and a lookup table of search radius t can be constructed. Suppose that more than t faults occur and the full syndrome is (\vec{s}_m, \vec{F}_m) , which is not in the lookup table. The table-based decoder discussed in Sec. II B will return the canonical recovery operator, which may cause a logical error after correction. To make successful error correction more probable in such cases, one could, in principle, construct a lookup table with a search radius larger than t by relaxing the distinguishability requirement for fault combinations with weights higher than t . However, this can be impractical, as the number of fault combinations grows too quickly when the search radius increases.

To overcome this issue, we instead conduct a *search during decoding*, starting from the missing syndrome (\vec{s}_m, \vec{F}_m) . That is, we construct another decoding table,

called the MIM table, with search radius at most $\rho \leq t$ using ideas similar to the original lookup table but we also add (\vec{s}_m, \vec{F}_m) to the map key before storing the syndrome in the MIM table and check whether or not it is in the decoding lookup table. If a new map key in the MIM table is the same as some map key in the decoding lookup table, the search stops and the decoder constructs a recovery operator from two combined data errors from the MIM table and the decoding table that correspond to the two map keys. If the MIM search radius reaches ρ and no matching syndrome is found, the decoder returns the CRO for the full syndrome. Using the recovery operator obtained from this method, we can correct up to $t + \rho$ faults with a probability higher than that found using the CRO of the full syndrome only.

An example of error decoding using a lookup table and the MIM technique is illustrated in Fig. 5. In our FTQEC protocols for hexagonal color codes of distance 3, 5, 7, and 9, we find that constructing the MIM table with search radius $\rho = t$ is sufficiently fast to be used at run time. Note that the MIM technique does not guarantee successful error correction due to potential degeneracy in syndromes above the guaranteed number of correctable faults. However, we do find numerically that the MIM technique has a positive impact on the performance of our decoders for the hexagonal color codes for distances 3, 5, 7, and 9.

IV. OPTIMIZATION TOOLS FOR TIME DECODING

In general, faults can happen at any point during syndrome measurements and the syndrome obtained at each round of measurements may not be the *correct syndrome* (the syndrome of the combined data error at the end of that round). In particular, measurement errors can lead to a syndrome that differs from the correct syndrome by some bits. Errors on the data or ancilla qubits that happen in the middle of the syndrome extraction can also result in a syndrome that only captures some parts of the correct syndrome. Applying a space decoder to a faulty syndrome can lead to an incorrect recovery operation. For this reason, one must perform multiple rounds of syndrome measurements.

The goal of a time decoder is to find a round with a correct syndrome at least at one point in the whole syndrome-measurement process. If this can be done, an FTQEC protocol satisfying both conditions in Definition 4 can be constructed. Note that according to the definition, it is sufficient to consider only the case in which the total number of faults in the whole protocol is no more than t , where t is the number of errors that a stabilizer code being used can correct. This is because the failure probability of the FTQEC protocol (the probability of having $t + 1$ or more faults in the protocol) will be $\mathcal{O}(p^{t+1})$ similar to the failure probability of an ideal error correction

with the same stabilizer code. (Nevertheless, in terms of better decoding accuracy, it is beneficial to consider correcting some cases of $t + 1$ or more faults, as suggested by the MIM technique in Sec. III B.)

In this section, we develop several types of time decoders for flag FTQEC, building on the ideas of adaptive decoders for Shor-style error correction [38]. Different time decoders use different fault-count-estimation procedures. In Sec. IV A, we describe a conventional way to perform repeated syndrome measurements for flag FTQEC in terms of difference vectors, which will be useful for the development in later sections. In Sec. IV B, we develop one-tailed and two-tailed adaptive time decoders that utilize flag information in the protocols. The one-tailed adaptive decoder is applicable to a larger family of codes, while the two-tailed adaptive decoder is more optimized to self-orthogonal CSS codes but needs to be used with an extended technique so that it becomes fully fault tolerant when applied to quantum computation. In Sec. IV C, we develop two extended techniques that can further improve the performance of our adaptive time decoders for FTQEC, given that the code being used is a self-orthogonal CSS code.

A. Shor time decoder for flag FTQEC

In Shor's original approach [1], the syndrome extraction is repeated until the same syndrome appears $t + 1$ times in a row. Observe that for R repeated but untrustworthy syndromes, at least R faults are required to make them the same (we can think of, e.g., having exactly the same measurement errors). Therefore, to make sure that a round with a correct syndrome exists when considering the case with up to t faults, it is sufficient to wait for $t + 1$ repeated measurements. A time decoder with this stop condition will be referred to as a *Shor time decoder*.

It is possible to rephrase the Shor time decoder using the notion of a difference vector. For a syndrome history $(\vec{s}_1, \vec{s}_2, \dots, \vec{s}_m)$ of length m , we define a *difference vector* $\vec{\delta}$ to be an $(m - 1)$ -bit string in which δ_i is 0 if $\vec{s}_{i+1} = \vec{s}_i$, or δ_i is 1 if $\vec{s}_{i+1} \neq \vec{s}_i$. As two repeated syndrome measurements are represented by a zero in the difference vector, Shor's method can be reformulated as waiting for t consecutive zeros in $\vec{\delta}$.

As we aim to correct no more than t faults, the analysis of our time decoders can be made easier by thinking about the budget of t faults. Shor's method spends all of this budget on counting consecutive zeros in the difference vector and is completely oblivious to other parts of the syndrome history (because the counter is reset whenever bit one appears). We call the parts of the syndrome history outside of the zero substring the *context* of the zero substring. As Shor's method does not take the context into account, we call this strategy "context unaware." In the worst-case scenario for the Shor time decoder, $(t + 1)^2$

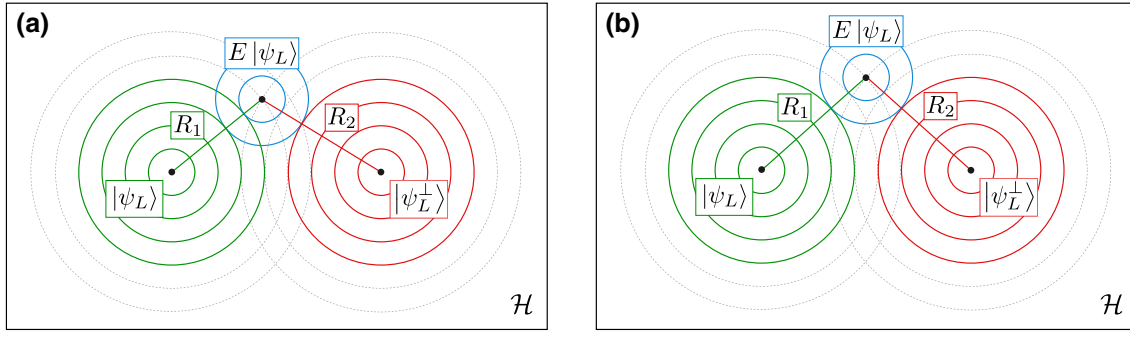


FIG. 5. An illustration of the error decoding using a lookup table and the MIM technique on the Hilbert space $\mathcal{H} = \mathbb{C}^{\otimes n}$ of the physical qubits. A code of distance 9 is considered in this example. Using a lookup table with search radius 4 only, any erroneous states lying on the green (or red) circles, which are up to four faults (circles) away from the logical state $|\psi_L\rangle$ (or $|\psi_L^\perp\rangle$), will be recovered to the logical state $|\psi_L\rangle$ (or $|\psi_L^\perp\rangle$). Consider the erroneous state $E|\psi_L\rangle$, which is not on any green or any red circle. In (a), $E|\psi_L\rangle$ is five faults away from $|\psi_L\rangle$ and six faults away from $|\psi_L^\perp\rangle$. Using the MIM table of radius 1, the recovery operator found by the decoder is R_1 . Since $R_1 E$ is a stabilizer, R_1 brings the state back to the original state $|\psi_L\rangle$. In (b), $E|\psi_L\rangle$ is six faults away from both $|\psi_L\rangle$ and $|\psi_L^\perp\rangle$. Using the MIM table of radius 2, the recovery operator found by the decoder is either R_1 such that $R_1 E$ is a stabilizer, or R_2 such that $R_2 E$ is a nontrivial logical operator. In this case, the state after recovery can be either $|\psi_L\rangle$ or $|\psi_L^\perp\rangle$.

rounds of syndrome measurements are done before the stopping condition is satisfied. The context of the zero substrings contains useful information and not counting the faults in the context results in underestimating the number of faults that can cause a given syndrome history. Context-aware strategies that have a better estimate of the number of faults can stop earlier and execute fewer measurements, resulting in higher pseudothresholds.

As flag circuits are used in the syndrome extraction, we also obtain a flag-vector history $(\vec{f}_1, \vec{f}_2, \dots, \vec{f}_m)$ from m rounds of syndrome measurements, which also leads to a cumulative-flag-vector history $(\vec{F}_1, \vec{F}_2, \dots, \vec{F}_m)$. Note that the calculation of a difference vector *does not* involve flag vectors; since the cases of faulty flag-qubit measurements are considered when we evaluate the distinguishability of a fault set, all flag-measurement outcomes are considered correct and can be used for error decoding. Our goal is to find a round such that all syndrome bits are correct.

The correct syndrome will be used in conjunction with the flag information obtained *right before* the measurements of the correct syndrome. Suppose that the code being used is a CSS code and X -type generator measurements at round i (which lead to $\vec{s}_{i,x}$, $\vec{f}_{i,x}$, and $\vec{F}_{i,x}$) are done before Z -type generator measurements (which lead to $\vec{s}_{i,z}$, $\vec{f}_{i,z}$, and $\vec{F}_{i,z}$). If the syndrome from round l is correct according to the Shor time decoder, Z -type (or X -type) error correction will be done using $\vec{s}_{l,x}$ and $\vec{F}_{l-1,z}$ (or $\vec{s}_{l,z}$ and $\vec{F}_{l,x}$). We also use similar ideas for error correction with other time decoders.

Suppose that a table-based space decoder for flag FTQEC can be constructed (as discussed in Sec. III). Then, a flag FTQEC protocol with a Shor time decoder is as follows.

Protocol 1 (Flag FTQEC protocol with Shor time decoder). Let $t = \lfloor (d-1)/2 \rfloor$ be the number of errors that a stabilizer code of distance d can correct. Let $\vec{s}_i = (\vec{s}_{i,x}, \vec{s}_{i,z})$ and $\vec{f}_i = (\vec{f}_{i,x}, \vec{f}_{i,z})$ be the syndrome and flag vector obtained from the i th round of full syndrome measurements with flag circuits. Let the cumulative flag vector at the i th round be $\vec{F}_i = (\vec{F}_{i,x}, \vec{F}_{i,z}) = \sum_{j=1}^i \vec{f}_j \pmod{2}$. After the i th round with $i \geq 2$, calculate δ_{i-1} . Repeat the syndrome measurements until the last t bits of δ reach zero or the total number of rounds reaches $(t+1)^2$. Suppose that the latest round is round l . Perform Z -type error correction using $(\vec{s}_{l,x}, \vec{F}_{l-1,z})$ and perform X -type error correction using $(\vec{s}_{l,z}, \vec{F}_{l,x})$.

B. Adaptive time decoder for flag FTQEC

Recently, FTQEC protocols with adaptive syndrome-measurement techniques have been proposed by some of the authors of this work [38]. Instead of using flag qubits, in that work, each stabilizer generator is measured using a syndrome-extraction circuit with a cat state (similar to Shor's original circuits [1]). The authors show that using the *adaptive strong decoder*, it is possible to reduce the number of syndrome-measurement rounds in the worst-case scenario from $(t+1)^2$ rounds to $(t+3)^2/4 - 1$ rounds. The resulting FTQEC protocol satisfies the error-weight-based definition of FTQEC (Definition 1) and is applicable to any stabilizer code. In this work, we extend the adaptive strong-decoder-based measurement techniques to flag FTQEC and develop protocols satisfying the revised FTQEC conditions that use the number of faults instead of the weight of errors (Definition 4). The main difference from Ref. [38] is that this work also uses flag information to estimate the number faults occurring in the

protocol, leading to a faster procedure to find a syndrome suitable for error correction. We start by describing the key ideas of Ref. [38] in terms of correlated and uncorrelated bit histories, which is useful for bounding the number of faults occurring from below. Afterward, we explain how each technique in Ref. [38] could be improved using the flag information.

1. Counting faults in correlated and uncorrelated bit histories

Let us first consider a way to estimate the number of faults occurring from a given difference vector $\vec{\delta}$. A single fault can cause either one or two consecutive bits of ones in $\vec{\delta}$ [38]. Thus, for each substring \vec{k} in $\vec{\delta}$, the number of faults that can cause such a substring is bounded from below by the number of 11 sequences plus the number of remaining ones in \vec{k} .

Suppose that the difference vector is of the form $\vec{\delta} = \eta_1 1 \eta_2 1 \dots 1 \eta_c$, where $\eta_j = 00 \dots 00$ are zero substrings and $1 \leq j \leq c$. For each η_j of length $\gamma_j \geq 1$ with $2 \leq j \leq c - 1$, we define α to be the total number of nonoverlapping 11 sequences plus the total number of remaining ones before the substring $1\eta_j 1$ and we define β_j similarly but for the substring after $1\eta_j 1$ (for η_1 and η_c , β_1 and α_c are defined similarly to those of other values of η_j and we let $\alpha_1 = 0$ and $\beta_c = 0$). The zero substring η_j of length γ_j corresponds to $\gamma_j + 1$ consecutive rounds with the same syndrome, so the number of rounds that can cause these rounds to give incorrect syndromes is at least $\gamma_j + 1$. Therefore, under the assumption that there are at most t faults in the whole protocol, if we find that there exists γ_j such that $t - \alpha_j - \beta_j < \gamma_j + 1$, the syndromes of the $\gamma_j + 1$ rounds that give rise to η_j cannot all be incorrect. For this reason, at least one syndrome corresponding to η_j is correct and can be used for error correction (for more details, see the full analysis in Ref. [38]).

For example, assume that the total number of faults in the protocol is $t = 4$ and that ten rounds of syndrome measurements give the following $\vec{\delta}$:

Round: 1 2 3 4 5 6 7 8 9 10

$\vec{\delta}$: 1 1 0 | 1 0 0 1 | 0 1

Focusing on the substring $1\eta_j 1 = 1001$, we find that $\alpha_j = 1$ and $\beta_j = 1$, meaning that the patterns of $\vec{\delta}$ on the left and the right sides of 1001 arise from at least two faults and the number of remaining faults is at most two. We can see that $\gamma_j = 2$ because of the two zeros in the substring 1001 and this corresponds to three rounds with the same syndrome. Since the number of remaining faults that can cause the pattern 1001 is less than the number of rounds with the same syndrome, the syndrome of at least one round in these three rounds must be correct and can be used for error correction.

There are multiple increasing fine-grained ways of estimating the number of faults in the context around each zero substring in the difference vector. Here, we use the term *bit history* as a general term for a series of syndrome bits (measurement outcomes) from a given stabilizer generator, a given flag bit (the measurement outcome of a flag qubit), or bits in a difference vector (taken as the difference history of a group of bits). A key element in this discussion is the notion of correlated and uncorrelated bit histories.

Under the assumed error model, two-bit histories are uncorrelated if they are independent of each other. For example, in our case, the circuit-level depolarizing channel is memoryless and each fault can cause either one or two consecutive bits of ones. Thus, different sections of the same syndrome bit history that are at least two bits apart are uncorrelated, as they are independent in time. Similarly, in space, if there are no shared qubits between two generators, then their syndrome bit histories are completely independent. Also, flag qubits are always reset between rounds of measurements and thus all flag bits are independent. However, when two stabilizer generators share at least one qubit, their syndrome bit histories are correlated. Similarly, due to hook errors, the bit history of the flag qubit and the syndrome bit history of that same stabilizer generator are correlated.

Our goal is to estimate the number of faults that have occurred from a given bit history in the case of flag FTQEC. Estimates from uncorrelated histories can be summed together. When two or more estimates are from correlated histories, the best we can do is to take the maximum of those estimates. Note that the total estimates must not exceed the actual total number of faults occurring in any case; otherwise, the error-correction protocol will not be fault tolerant.

For the estimation in the previous work [38], which is discussed previously in this section, the bits of the syndrome history before and after each substring $1\eta_j 1$ are uncorrelated to the bits within η_j under the memoryless depolarizing channel assumption. This means that α_j and β_j , which are the minimum numbers of faults that can cause the substring before and after $1\eta_j 1$, can be independently estimated. The estimated number of faults in the context outside of the zero substring η_j is, therefore, $\alpha_j + \beta_j$.

In this work, we further extend the fault-counting idea to flag FTQEC in which flag circuits with a single flag qubit are used for syndrome extraction. Below, we will discuss two types of adaptive time decoders with different stop conditions, namely, *one-tailed* and *two-tailed adaptive time decoders*. Both protocols are applicable to any stabilizer code as long as flag circuits for the code that give a distinguishable fault set can be found. The flag FTQEC protocol with one-tailed adaptive time decoder satisfies the FTQEC conditions in Definition 4; thus it is

applicable to any fault-tolerant quantum computation as long as the fault-tolerant implementation of other operations (gate, state preparation, or measurement) also satisfies the revised definition of fault tolerance, which considers the number of faults instead of the weight of the error [35]. Meanwhile, the flag FTQEC protocol with two-tailed adaptive time decoder does not satisfy the FTQEC conditions in Definition 4, as the output error may correspond to a nontrivial cumulative flag vector; hence it is only applicable to quantum memory. Nevertheless, for a self-orthogonal CSS code, the FTQEC protocol with the two-tailed adaptive time decoder can be applied to any fault-tolerant Clifford computation if the cumulative flag vector is processed appropriately. An analysis of this extension will be discussed in Sec. IV C.

2. Two-tailed adaptive time decoder

For the substring $1\eta_j 1$ in $\vec{\delta}$, suppose that bit one on the left of η_j is the i_1 th bit of $\vec{\delta}$, and bit one on the right of η_j is the i_2 th bit of $\vec{\delta}$. Let α_j , β_j , and γ_j be defined as before and let μ_j and ν_j be the total numbers of nonzero flag bits obtained from round 1 to round i_1 and from round $i_2 + 1$ onward. Also, let ω_j be the sum of the numbers of flag bits that *exceed one bit per round* during round $i_1 + 1$ to round i_2 . For example, consider the substring $1\eta_j 1 = 1001$ in the example below:

Round: 1 2 3 4 5 6 7 8 9 10
 Number of flag bits: 1 0 2 0|0 2 1|0 0 1
 $\vec{\delta}$: 1 1 0|1 0 0 1|0 1

In this example, $\alpha_j = 1$, $\beta_j = 1$, $\gamma_j = 2$, $\mu_j = 3$, $\nu_j = 1$, and $\omega_j = 1$.

Since a single fault can cause both nontrivial flag bits and syndrome differences (i.e., syndrome bits and flag bits are correlated), one has to make sure that the number of faults is not over-counted. The numbers of faults that can cause bit histories before and after $1\eta_j 1$ are bounded from below by $\tilde{\alpha}_j = \max(\alpha_j, \mu_j)$ and $\tilde{\beta}_j = \max(\beta_j, \nu_j)$, respectively. So an estimate of the number of faults for the context outside of η_j is $\tilde{\alpha}_j + \tilde{\beta}_j$.

Next, let us consider η_j of length γ_j , which corresponds to $\gamma_j + 1$ consecutive rounds with the same syndrome. To make all syndromes in this region incorrect requires at least one fault per round. So if we find a round with more than one flag bit, the number of flag bits that exceed one bit per round can be a part of the total estimate. That is, for each η_j , the total estimate is $\tilde{\alpha}_j + \tilde{\beta}_j + \omega_j$.

Under the assumption that there are at most t faults in the whole protocol, if we find that there exists γ_j such that $t - \tilde{\alpha}_j - \tilde{\beta}_j - \omega_j < \gamma_j + 1$ (or, equivalently, $\tilde{\alpha}_j + \tilde{\beta}_j + \gamma_j + \omega_j \geq t$), we know that a syndrome of at least one round in the $\gamma_j + 1$ rounds that give rise to η_j must be correct.

Another way to find a correct syndrome is to estimate the total number of faults that can cause the whole syndrome and flag-bit histories. Let N_{11} be the total number of nonoverlapping 11 sequences in the whole $\vec{\delta}$. Assuming that there are at most t faults in the whole protocol, if $N_{11} \geq t$, the last round must have a correct syndrome.

Suppose that a table-based space decoder for flag FTQEC can be constructed. Then, a flag FTQEC protocol with two-tailed adaptive time decoder is as follows.

Protocol 2 (Flag FTQEC protocol with two-tailed adaptive time decoder). Let $t = \lfloor (d-1)/2 \rfloor$ be the number of errors that a stabilizer code of distance d can correct. Let $\vec{s}_i = (\vec{s}_{i,x}, \vec{s}_{i,z})$ and $\vec{F}_i = (\vec{F}_{i,x}, \vec{F}_{i,z})$ be the syndrome and cumulative flag vector obtained from the i th round of full syndrome measurements with flag circuits. After the i th round with $i \geq 2$, calculate δ_{i-1} . Repeat the syndrome measurements until one of the following conditions is satisfied and then perform error correction using the error syndrome corresponding to each condition:

- (1) For each η_j in $\vec{\delta}$, calculate $\tilde{\alpha}_j, \tilde{\beta}_j, \gamma_j, \omega_j$. If at least one η_j with $\tilde{\alpha}_j + \tilde{\beta}_j + \gamma_j + \omega_j \geq t$ is found, stop the syndrome measurements. Let l be the last round of the $\gamma_j + 1$ rounds that correspond to η_j . Perform Z-type error correction using $(\vec{s}_{l,x}, \vec{F}_{l-1,z})$ and perform X-type error correction using $(\vec{s}_{l,z}, \vec{F}_{l,x})$.
- (2) Calculate N_{11} from the whole syndrome and flag-bit histories. If $N_{11} \geq t$, stop the syndrome measurements. Suppose that the latest round is round l . Perform Z-type error correction using $(\vec{s}_{l,x}, \vec{F}_{l-1,z})$ and perform X-type error correction using $(\vec{s}_{l,z}, \vec{F}_{l,x})$.

The two-tailed adaptive time decoder for flag FTQEC developed in this work uses similar ideas to the adaptive strong decoder presented in the previous work [38]. Therefore, the number of syndrome-measurement rounds in the worst-case scenario is $(t+3)^2/4 - 1$ when t is odd and is $(t+2)(t+4)/4 - 1$ when t is even. This can be proved by assuming that none of the faults causes a nonzero flag bit, then the rest of the proof follows the proof of Theorem 2 of the previous work [38].

If the syndrome \vec{s}_l and cumulative flag vector $\vec{F}_l = \sum_{i=1}^l \vec{f}_i \pmod{2}$ of round l are used for error correction, any faults that have happened up to round l will be corrected. However, because round l may correspond to some η_j in the middle of $\vec{\delta}$, an output error may correspond to a nontrivial cumulative flag vector. Therefore, Protocol 2 may not satisfy the FTQEC conditions in Definition 4 and cannot be applied to fault-tolerant quantum computation. Nevertheless, Protocol 2 is still applicable to a quantum memory. To do so, one needs to pass the *remaining cumulative flag vector* of the current FTQEC routine (the sum of the flag vectors from round $l+1$ onward) to the next FTQEC routine and use it as an initial flag vector.

3. One-tailed adaptive time decoder

One-tailed and two-tailed decoders use similar ideas to estimate the number of faults, except that in the one-tailed case, the syndrome and cumulative vector for error correction must be from the very last zero substring in $\vec{\delta}$ (this is to ensure that the output error satisfies both conditions in Definition 4). Suppose that $\vec{\delta} = \eta_1 1 \eta_2 1 \dots 1 \eta_c$ for some positive integer c , η_c has length $\gamma_c \geq 1$, and bit one on the left of η_c is the i_1 th bit of $\vec{\delta}$. We define α_c as usual and define μ_c to be the total number of nonzero flag bits obtained from round 1 to round i_1 . Also, we define ω_c to be the sum of the numbers of flag bits that exceed one bit per round during round $i_1 + 1$ onward. Let $\tilde{\alpha}_c = \max(\alpha_c, \mu_c)$. In this case, the total estimate of the number of faults occurring is $\tilde{\alpha}_c + \omega_c$.

Assuming that there are at most t faults in the whole protocol, if we find that $\tilde{\alpha}_c + \gamma_c + \omega_c \geq t$, at least one round in the $\gamma_c + 1$ rounds that give rise to η_c must have a correct syndrome. This is the first possible stop condition.

The second possible stop condition is similar to what we have for the two-tailed decoder. Let N_{11} be the total number of nonoverlapping 11 sequences in the whole $\vec{\delta}$. If $N_{11} \geq t$, the last round must have a correct syndrome.

Suppose that a table-based space decoder for flag FTQEC can be constructed. Then, a flag FTQEC protocol with the one-tailed adaptive time decoder is as follows.

Protocol 3 (Flag FTQEC protocol with one-tailed adaptive time decoder). Let $t = \lfloor (d-1)/2 \rfloor$ be the number of errors that a stabilizer code of distance d can correct. Let $\vec{s}_i = (\vec{s}_{i,x}, \vec{s}_{i,z})$ and $\vec{F}_i = (\vec{F}_{i,x}, \vec{F}_{i,z})$ be syndrome and cumulative flag vector obtained from the i th round of full syndrome measurements with flag circuits. After the i th round with $i \geq 2$, calculate δ_{i-1} . Repeat the syndrome measurements until one of the following conditions is satisfied:

- (1) $\tilde{\alpha}_c, \gamma_c, \omega_c$ satisfy $\tilde{\alpha}_c + \gamma_c + \omega_c \geq t$.
- (2) $N_{11} \geq t$.

Suppose that the latest round when any condition is satisfied is round l . Perform Z -type error correction using $(\vec{s}_{l,x}, \vec{F}_{l-1,z})$ and perform X -type error correction using $(\vec{s}_{l,z}, \vec{F}_{l,x})$.

The number of rounds of full syndrome measurements in the worst-case scenario for Protocol 3, which is also the minimum number of rounds required to guarantee that error correction can be done, can be found using the following theorem.

Theorem 1. Suppose that the flag circuits being used in Protocol 3 give a distinguishable fault set \mathcal{F}_t , where $t = \lfloor (d-1)/2 \rfloor$ and d is the distance of the stabilizer code. Performing $\lceil t(t+3)/2 \rceil + 2$ rounds of full syndrome

measurements is sufficient to guarantee that Protocol 3 is strongly t -fault tolerant; i.e., both conditions in Definition 4 are satisfied.

Proof. Suppose that $\vec{\delta} = \eta_1 1 \eta_2 1 \dots 1 \eta_c$ and $\gamma_c \geq 1$. We will show that if none of $\eta_1, \eta_1 1 \eta_2, \eta_1 1 \eta_2 1 \eta_3, \dots, \eta_1 1 \eta_2 1 \dots 1 \eta_c$ satisfies any condition in Protocol 3, the maximum length of such $\vec{\delta}$ is $t(t+3)/2$. In the worst-case scenario, flag-measurement results do not help in estimating the number of faults occurring, so we can assume that $\tilde{\alpha}_c = \alpha_c$ and $\omega_c = 0$. Below are the results from analyzing $\eta_1, \eta_1 1 \eta_2$, and $\eta_1 1 \eta_2 1 \eta_3$:

- (1) For $\eta_1, \alpha_c = 0$ and $\gamma_c = \gamma_1$, so the maximum length of η_1 such that $\tilde{\alpha}_c + \gamma_c \geq t$ is not satisfied is $t-1$.
- (2) For $\eta_1 1 \eta_2, \alpha_c = 0$ and $\gamma_c = \gamma_2$, so the maximum length of η_2 such that $\tilde{\alpha}_c + \gamma_c \geq t$ is not satisfied is $t-1$.
- (3) For $\eta_1 1 \eta_2 1 \eta_3, \alpha_c = 1$ and $\gamma_c = \gamma_3$, so the maximum length of η_3 such that $\tilde{\alpha}_c + \gamma_c \geq t$ is not satisfied is $t-2$.

By induction, the maximum length of $\vec{\delta} = \eta_1 1 \eta_2 1 \dots 1 \eta_c$ such that $\tilde{\alpha}_c + \gamma_c \geq t$ is not satisfied is $(t-1) + 1 + (t-1) + 1 + (t-2) + 1 + \dots + 1 + 1 + 0 + 1$, which is $t(t+3)/2$. Here, $\vec{\delta}$ is of the form

$$\underbrace{00 \dots 00}_{t-1} \underbrace{1 00 \dots 00}_{t-1} \underbrace{1 00 \dots 00}_{t-2} \underbrace{1 00 \dots 00}_{t-3} 1 \dots 1001011. \quad (11)$$

The number of rounds that gives $\vec{\delta}$ of the maximum length is $t(t+3)/2 + 1$. By performing one more round of syndrome measurements, $\vec{\delta}$ is extended by one bit, which must be bit zero if the total number of faults is no more than t . In that case, $\tilde{\alpha}_c + \gamma_c \geq t$ will be satisfied. Therefore, $\lceil t(t+3)/2 \rceil + 2$ rounds of full syndrome measurements are sufficient to guarantee that flag FTQEC can be performed.

Note that there are other forms of $\vec{\delta}$ in which none of $\eta_1, \eta_1 1 \eta_2, \eta_1 1 \eta_2 1 \eta_3, \dots, \eta_1 1 \eta_2 1 \dots 1 \eta_c$ satisfies any condition in Protocol 3 and the length of $\vec{\delta}$ is $\lceil t(t+3)/2 \rceil - 1$. For example, suppose that $t = 3$. Possible forms of such $\vec{\delta}$ are 001101011 and 001001111. In any case, one of the conditions in Protocol 3 will be satisfied if one more round of syndrome measurements is done, so the number of rounds to guarantee fault tolerance is still $\lceil t(t+3)/2 \rceil + 2$. ■

Note that the number given by Theorem 1 is worse than that of the two-tailed decoder because we are not allowed to check whether the syndrome of any round in the middle can be used for error correction.

An advantage of the FTQEC protocol with the one-tailed adaptive time decoder is that it is applicable to any kind of fault-tolerant quantum computation as long as the corresponding fault-tolerant implementation satisfies the

revised definitions of fault tolerance, which consider the number of faults instead of the weight of errors [35]. This is possible because when the syndrome and cumulative flag vector for error correction are from the last zero substring in $\vec{\delta}$, it is guaranteed that the output error corresponds to a zero cumulative flag vector.

C. Extended techniques for CSS codes

In this section, we discuss two additional techniques that can further improve our flag FTQEC protocols with adaptive time decoding. The first technique is separate X and Z counting, which is applicable to any CSS code. This technique is based on the ideas from Refs. [37,38] and can be used to improve the pseudothreshold. The main difference from the technique developed in Ref. [38] is that this work also uses flag information to estimate the number of faults occurring, making the procedure to obtain a syndrome for error correction terminate faster. The second technique is the classical processing of the remaining cumulative flag vector. This technique allows our flag FTQEC protocol with the two-tailed adaptive time decoder to be applicable to any fault-tolerant Clifford computation.

1. Separate X and Z counting

For any CSS code, Z -type and X -type errors can be corrected separately. It is possible to improve the number of measurements by separating the X - and Z -type syndrome-measurement rounds (which correspond to X - and Z -type stabilizer generators). In this section, we introduce the XZ and ZX decoding strategies. In the XZ strategy, first, we execute a time decoder (which can be a Shor, one-tailed, or two-tailed decoder) using only the X -type syndromes. The difference vector for this process is denoted by $\vec{\delta}_x$. After the decoder returns the X -type syndrome and the cumulative flag vectors for Z -type error correction, we estimate the number of faults t_x that could cause $\vec{\delta}_x$; we define $\alpha_{\text{all},x}$ to be the total number of nonoverlapping 11 sequences plus the total number of remaining ones in $\vec{\delta}_x$, define $\mu_{\text{all},x}$ to be the total number of nontrivial flag bits in $\vec{\delta}_x$, and let $t_x = \max(\alpha_{\text{all},x}, \mu_{\text{all},x})$. Given that we spend this number of faults from our fault budget t , we can reduce the target number of faults in the stop condition for the Z -type syndrome measurements. Afterward, we run a time decoder for Z -type syndromes with the target number of faults $t_z = t - t_x$. The ZX strategy is similar to the XZ strategy, except that the Z -type generators are measured first.

When the separate X - and Z -counting technique is applied to a flag FTQEC protocol, one can find syndromes for Z -type and X -type error corrections faster compared to a conventional method where the target numbers of faults for both types of error corrections are t . However, a drawback is that the flag FTQEC protocol will only be compatible with quantum memory. This is because each

type of error correction requires flag information of the opposite type. In particular, suppose that the time decoder for X -type syndrome measurements gives syndrome \vec{s}_x and cumulative flag vector \vec{F}_x and the time decoder for Z -type syndrome measurements gives syndrome \vec{s}_z and cumulative flag vector \vec{F}_z . Z -type error correction will be done by applying a space decoder to \vec{s}_x and the zero cumulative flag vector, while X -type error correction will be done by applying a space decoder to \vec{s}_x and \vec{F}_z . The cumulative flag vector \vec{F}_x that has not been used will be treated as the remaining cumulative flag vector of the current FTQEC routine and used as an initial flag vector for Z -type error correction in the next FTQEC routine.

2. Classical processing of the remaining cumulative flag vector

One drawback of a flag FTQEC protocol that uses the two-tailed adaptive time decoder or the separate X - and Z -counting technique is that it is only applicable to a quantum memory, not general fault-tolerant quantum computation. This is because the output error at the end of each FTQEC routine may correspond to a nontrivial cumulative flag vector. To correct such an error, one needs to pass the flag information from each FTQEC routine (the remaining cumulative flag vector) to the next FTQEC routine. However, if there is some quantum computation between two FTQEC routines (as in an extended rectangle [6]), the error will be transformed and may not be correctable if the corresponding flag information is not processed properly.

Nevertheless, for any self-orthogonal CSS code, a flag FTQEC protocol with a two-tailed adaptive time decoder or separate X and Z counting (or both) can be made applicable to any fault-tolerant Clifford computation. For example, let us consider an application of a logical Hadamard gate \bar{H} between two FTQEC routines. Suppose that the first FTQEC routine causes an output error $E_x E_z$ and the remaining cumulative flag vector is (\vec{F}_x, \vec{F}_z) . Without a logical Hadamard gate, E_x and E_z can be corrected using \vec{F}_z and \vec{F}_x , respectively. A logical Hadamard gate transforms an X -type error to a Z -type error of the same form and vice versa. Because the X - and Z -type generators are of the same form, possible fault combinations for both types of errors are also of the same form. To correct the transformed error $\bar{H}(E_x E_z)\bar{H}^\dagger$ in the second FTQEC routine, one needs to swap the X - and Z -type cumulative flag vector; i.e., the initial flag vector for the second FTQEC routine must be (\vec{F}_z, \vec{F}_x) .

We can apply similar ideas for flag information processing to logical S and logical CNOT gates. The summary of the classical processing operations for logical H , S , and CNOT gates is provided in Table II. Because $\{H, S, \text{CNOT}\}$ generates the Clifford group, a flag FTQEC protocol with two-tailed adaptive time decoder or separate X and Z

TABLE II. A list of the required classical processing operations on the remaining cumulative flag vector in the case in which a logical Clifford gate is performed between two FTQEC routines. With these operations, a flag FTQEC protocol with two-tailed adaptive time decoder or separate X and Z counting is applicable to any fault-tolerant Clifford computation, given that the CSS code is self-orthogonal.

Remaining cumulative flag vector of the current FTQEC routine	Logical Clifford operation	Initial flag vector of the next FTQEC routine
(\vec{F}_x, \vec{F}_z)	\bar{H}	(\vec{F}_z, \vec{F}_x)
(\vec{F}_x, \vec{F}_z)	\bar{S}	$(\vec{F}_x, \vec{F}_x \oplus \vec{F}_z)$
$(\vec{F}_{x,1}, \vec{F}_{z,1} \vec{F}_{x,2}, \vec{F}_{z,2})$	$\overline{\text{CNOT}}_{1,2}$	$(\vec{F}_{x,1}, \vec{F}_{z,1} \oplus \vec{F}_{z,2} \vec{F}_{x,1} \oplus \vec{F}_{x,2}, \vec{F}_{z,2})$

counting is applicable to any fault-tolerant Clifford computation given that the CSS code is self-orthogonal. Note that magic state distillation and injection [56,57] use only Clifford operations. Thus, our techniques are also applicable to fault-tolerant universal quantum computation given that high-fidelity magic states are provided.

V. NUMERICAL RESULTS

A. Methods

Our optimization tools for space and time decoders, including the compact lookup-table construction, the MIM technique, and the adaptive time decoders for flag FTQEC are applicable to any stabilizer code. However, we focus on a specific family of codes where the aforementioned tools can be simplified and extended techniques, including separate X and Z decoding and classical processing of flag information, are applicable—the family of self-orthogonal CSS codes in which the number of physical qubits is odd, the number of logical qubits is 1, and logical X and Z operators are transversal. To evaluate the performance of our tools, we simulate FTQEC protocols on the $[(3d^2 + 1)/4, 1, d]$ hexagonal color codes [41] of distance 3, 5, 7, and 9. These codes are planar topological codes with configurations displayed in Fig. 6. For each code, stabilizer generators are measured using the syndrome-extraction circuits with single flag ancilla, as depicted in Fig. 4. It has been proven that for the hexagonal code of any distance, the use of flag circuits of this form preserves the code distance regardless of the gate orderings [32,35]. The simulation is implemented under the circuit-level depolarizing noise model specified in Sec. II A. As there is no idling noise in our error model, the syndromes can be extracted sequentially.

To construct a lookup table for space decoding and to verify that our circuit configurations preserve the code distance, we implement the algorithm described in Sec. III A

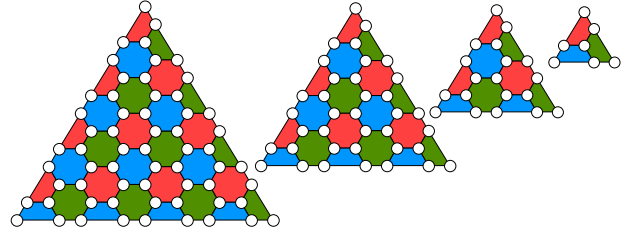


FIG. 6. The studied members of the hexagonal color code, for distances 3, 5, 7, and 9 (right to left). Qubits are on the vertices and stabilizer generators are the plaquettes. As the codes are self-orthogonal CSS codes, both the X - and Z -stabilizer generators are described by the same layout.

using C++. The timing for verification alongside the statistics of the lookup table can be found in Table I. The lookup table for these codes can be generated on the fly before the sampling starts, as the required time is low enough.

Here, we simulate the storage (i.e., the result of the logical identity operation) of the logical state $|\bar{0}\rangle$. We use the Pauli-frame simulator in STIM [58] to collect measurement samples and we use CIRQ [59] for constructing the circuits with the given noise model. After a perfect preparation of $|\bar{0}\rangle$, we perform noisy error correction and recovery. In the error-correction process, full rounds of syndrome measurements are repeated until the stop condition of the time decoder is satisfied. The time decoder returns an accepted full syndrome (consisting of error syndrome and cumulative flag vector) and then the space decoder determines the recovery operation based on the accepted full syndrome. This recovery operation is applied to the data qubits afterward. Finally, we apply an ideal error correction and determine whether the output error is a logical X error (which corresponds to having $|\bar{1}\rangle$ as the output state).

B. The overall effect of optimization tools

We first compare two protocols: (1) the FTQEC protocol with a Shor time decoder without the MIM technique (the protocol in which none of our optimization tools are applied) and (2) the FTQEC protocol with the MIM technique and the two-tailed adaptive time decoder with the ZX strategy (the best FTQEC protocol in this work, which is compatible with any Clifford computation on a self-orthogonal CSS code). The logical error rate p_L versus the physical error rate p for hexagonal color codes of distance 3, 5, 7, and 9 is plotted in Fig. 7. Our results show that for each code, applying the optimization tools can significantly improve the pseudothreshold (the intersection between each plot and the $p_L = 2p/3$ line). Furthermore, the optimized decoder yields orders-of-magnitude improvements in the logical error rate in the $p = 10^{-4}$ error regime.

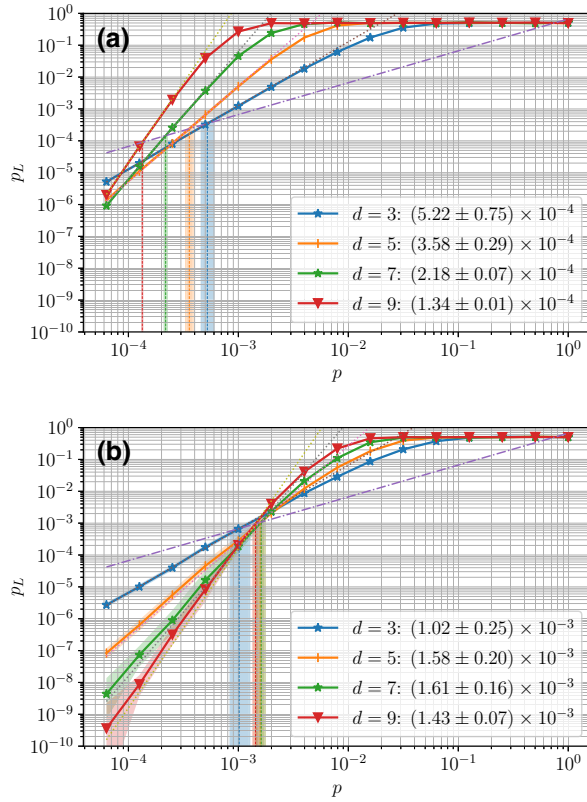


FIG. 7. (a) The curve of the logical error rate p_L versus the physical error rate p for the hexagonal color code family without any of our optimization techniques, using the Shor time decoder without MIM. (b) The use of the best-performing combination of our techniques, including MIM and the two-tailed adaptive time decoder with ZX strategy. Pseud thresholds for each curve (the p_{th} error rate, which gives $p_L(p_{th}) = 2p_{th}/3$) are included in the labels and marked with vertical lines. The data points represent the number of logical errors divided by the total number of samples at that p error rate and thus estimate the true logical error rates, which should lie within the shaded areas with high confidence. The dotted helper lines, which are αp^{t+1} , where $\alpha = \frac{2}{3}p_{th}^{-t}$ retroactively calculated for each curve from its pseud threshold, show good agreement with distance preservation.

Under a noise model parametrized by a single parameter p , the fault-tolerant threshold p_{th} is the error probability under which the logical error rate is guaranteed to decrease with increasing code distance for a specific code family and decoder. Our decoders can yield a p_{th} for concatenated code families using a level-by-level decoder but they will not yield a threshold for topological code families for two reasons. The practical reason is that our space decoder, which uses a lookup table, is not scalable to the large d limit. The fundamental reason is that the time decoder will always take δ in which all bits are one when d is large, because δ_j for each round will be 1 with a probability exponentially close to 1 for finite p . The space decoder then acts on the final state but lacks the information about

TABLE III. The effect of different time decoders (rows) and the MIM technique (columns) on the pseud threshold of the distance-9 hexagonal color code (for more details on the underlying data, see Figs. 8–10).

Time decoder	Without MIM ($\times 10^{-4}$)	With MIM ($\times 10^{-4}$)
Shor	1.34 ± 0.01	2.79 ± 0.07
One-tailed	2.11 ± 0.05	3.91 ± 0.26
Two-tailed	3.38 ± 0.17	6.30 ± 0.45
Two-tailed XZ	—	6.09 ± 0.47
Two-tailed ZX	—	14.3 ± 0.7

correlations to properly correct it. This is why an efficient space-time decoder is critical for achieving p_{th} for topological codes.

We can define an effective threshold \tilde{p}_{th} as the error rate below which increasing the code distance improves the logical error rate for this finite set of codes. The optimized protocol yields a $\tilde{p}_{th} = 1.5 \times 10^{-3}$, while the unoptimized protocol yields $\tilde{p}_{th} = 4.5 \times 10^{-5}$. We also note that the crossing point between the codes of distances d and $d-2$ is dropping quickly with the unoptimized decoder, while it is stable for the optimized decoder over this code set. The effect of each optimization technique to the crossing points can be found in Fig. 13 in Appendix C. In Table III, we summarize the effects of different optimization tools on the pseud threshold of the $d=9$ color code. In the following sections, we further discuss the effect of each technique that can contribute to this improvement.

C. The effect of the meet-in-the-middle technique

In this section, we evaluate the performance of simulated storage that uses the space decoder with and without the MIM technique. We explore the effect for distances 3, 5, 7, and 9 and compare the effect when the time decoder is a Shor, one-tail, or two-tail time decoder. We observe a significant decrease in logical error rates and an improvement in pseud threshold when the MIM technique is applied. We also find that the benefit increases with the code distance. In Fig. 8, we show the improvement for the code of distance 9, where the benefit is the largest. The results for codes of other distances are provided in Figs. 14–16.

It is clear that both nonadaptive (Shor) and adaptive time decoders benefit from the MIM technique. The pseud threshold for the Shor time decoder increases by more than 100%, from $(1.34 \pm 0.01) \times 10^{-4}$ to $(2.79 \pm 0.07) \times 10^{-4}$. The pseud threshold for the one-tailed adaptive time decoder increases by 85%, from $(2.11 \pm 0.05) \times 10^{-4}$ to $(3.91 \pm 0.26) \times 10^{-4}$. Finally, the pseud threshold for the two-tailed adaptive time decoder gets a boost of 86%, from $(3.38 \pm 0.17) \times 10^{-4}$ to $(6.30 \pm 0.45) \times 10^{-4}$.

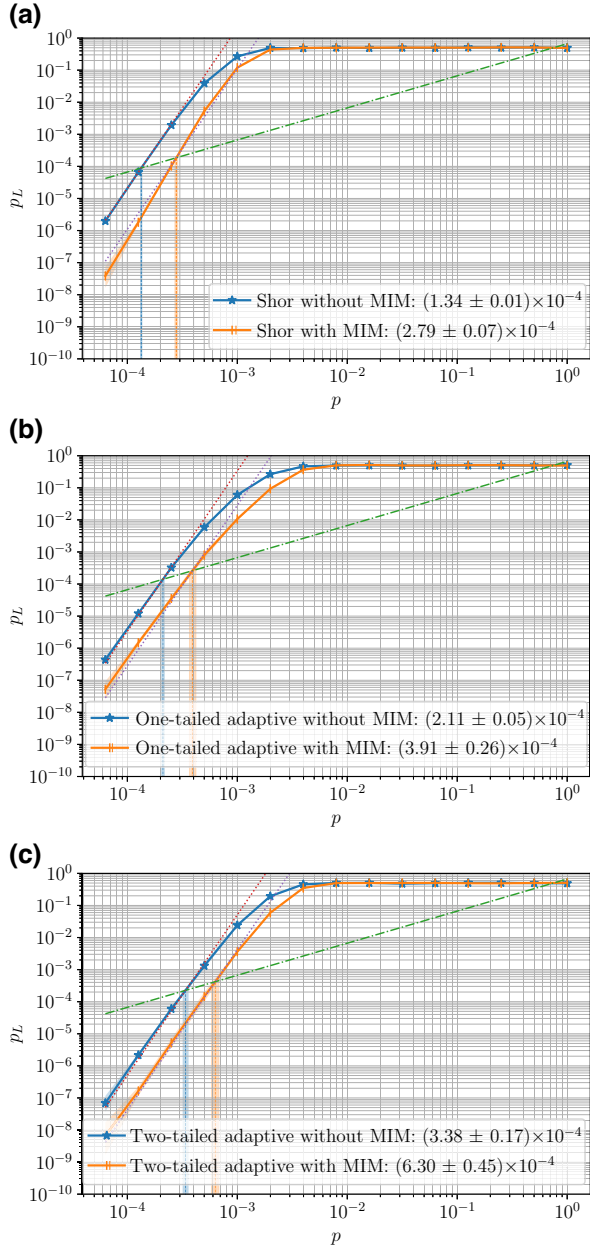


FIG. 8. The effect of the MIM technique on different time decoders at distance $d = 9$: (a) Shor time decoder; (b) one-tailed adaptive time decoder; (c) two-tailed adaptive time decoder. The effect is largest for the Shor time decoder, more than doubling the pseudethreshold. The MIM technique also gives at least a significant 70% improvement on the adaptive time decoders.

D. The effect of the adaptive time decoders

In this section, we compare the performance of the simulated storage numerical experiments that use different time decoders when the MIM technique is applied. The results are displayed in Fig. 9 for the hexagonal code of distance 9 and we refer the reader to Fig. 17 in Appendix C for the results for the codes of other distances.

For the code of distance 9, in comparison with the Shor time decoder, the one-tailed adaptive time decoder improves the pseudethreshold by 40% from $(2.79 \pm 0.07) \times 10^{-4}$ to $(3.91 \pm 0.26) \times 10^{-4}$. The two-tailed method achieves $(6.30 \pm 0.45) \times 10^{-4}$ pseudethreshold, which is more than a 125% increase compared to the Shor time decoder. However, this gain vanishes at lower error rates and the performances of the Shor and the one-tailed decoder become similar at around $p = 10^{-4}$. This is not surprising, as we expect all adaptive time decoders to converge to a Shor time decoder at lower error rates. The main reason for this convergence is that the performance gains for the adaptive techniques come from a decrease in the average number of rounds for syndrome measurements and the decrease converges to zero at low error rates. How fast the decrease converges matters and, in contrast to the one-tailed approach, the two-tailed time decoder preserves its performance gain over the Shor time decoder in the observed error-rate regime as low as 5×10^{-5} .

We also provide the plots of the average numbers of full rounds of measurements for all decoders. In the low-error-rate regime, all decoders have the same minimum number of measurement rounds, $t + 1$, which corresponds to the case in which all bits in the difference vector are zeros. We can see the separation more clearly when the physical error rate is in the 10^{-3} range; the two-tailed time decoder requires the fewest rounds, followed by the one-tailed decoder, and the Shor time decoder performs the worst. In the high-error-rate regime, all bits in the difference vector tend to be ones. In this case, the Shor time decoder requires $(t + 1)^2$ rounds, while both the one-tailed and two-tailed decoders require $2t + 1$ rounds.

E. The effect of the separate X - and Z -counting technique

In this section, we observe the performance gains when the separate X - and Z -counting technique is applied. Here, we compare the FTQEC protocols that use the two-tailed adaptive time decoder with joint X - and Z -generator measurements (as in Sec. IV B 2), the two-tailed adaptive time decoder with the XZ strategy, and the two-tailed adaptive time decoder with the ZX strategy (as in Sec. IV C 1). The logical error rate is calculated from the number of samples in which the output error is a logical X error. The p_L versus p plots for the code of distance 9 are shown in Fig. 10 (the results for codes of other distances can be found in Fig. 18 in Appendix C).

In terms of the pseudethreshold, we observe that the decoder with separate X and Z counting performs the best when Z -type generators are measured before X -type generators. Compared to the two-tailed decoder with joint X - and Z -generator measurements, the separate two-tailed ZX decoder improves the pseudethreshold by 127%, from $(6.30 \pm 0.45) \times 10^{-4}$ to $(1.43 \pm 0.07) \times 10^{-3}$. This

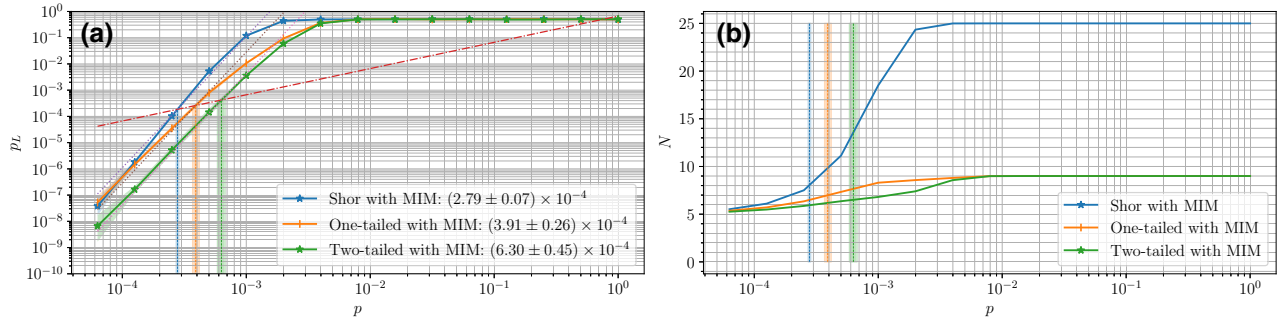


FIG. 9. The logical error rates of the one-tailed and two-tailed adaptive time decoders compared to the Shor time decoder with the corresponding average number of rounds for the hexagonal color code of distance 9: (a) logical error rate p_L versus physical error rate p ; (b) average number of rounds N versus physical error rate p .

is mainly because measuring generators of the first type (X or Z) requires more rounds and it is more probable that the measurements can cause correlated errors of the same type as the generators being measured (which are more difficult to correct than uncorrelated errors, since they require flag information). Because in our simulations we measure the performance of storing the logical $|0\rangle$ state (thus, a logical X error is counted), the decoder that measures X -type generators first performs worse. We also observe that there is no significant difference between the two-tailed decoder with joint measurements and the two-tailed decoder with the XZ strategy.

We also provide plots of the average number of full rounds of measurements for all decoders (where the full round of single-type generator measurements is counted as half a round of total measurements). In the low-error-rate regime, all decoders require $t + 1$ rounds. For the original two-tailed decoder, the average number of rounds increases as the physical error rate increases and it reaches $2t + 1$ rounds in the high-error-rate regime. For both two-tailed decoders with separate X and Z counting, we find that the average number of rounds increases near the pseudthreshold, then there are the dips after the

pseudthreshold, and the numbers reach $t + 1$ rounds in the high-error-rate regime. The dips come from the fact that the measurements of generators of the first type (either X or Z) can stop at less than $(2t + 1)/2$ rounds but the estimate of the number of faults occurring can be t , which then causes the measurements of generators of the second type to stop at $1/2$ rounds. In the high-error-rate regime, the decoders with separate X and Z counting require $t + 1$ rounds, since measuring generators of the first type requires $(2t + 1)/2$ rounds while measuring generators of the second type requires $1/2$ rounds on average. Overall, the decoder that measures Z -type generators first performs better than the decoder that measures X -type generators first.

VI. DISCUSSION AND CONCLUSIONS

In this work, we focus on flag FTQEC with lookup-table decoding and improvements to a decoder consisting of a time decoder and a space decoder. For the space decoder, we first develop a technique to build the lookup table more efficiently in Sec. III A. With our lookup-table construction method, the lookup table for a self-orthogonal CSS code requires at least 87.5% less memory compared to the

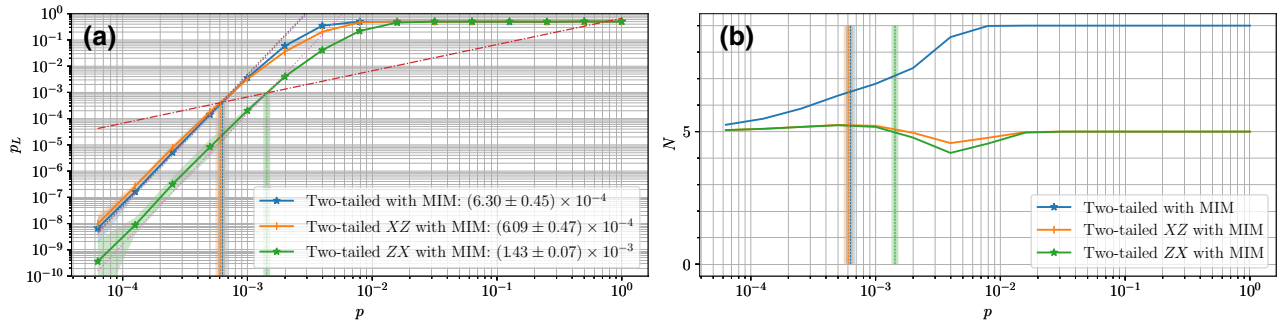


FIG. 10. The logical error rates of the two-tailed time decoder with XZ and ZX strategies in comparison with the two-tailed adaptive time decoder with joint X and Z measurements and the corresponding average number of rounds for the hexagonal color code of distance 9: (a) logical error rate p_L versus physical error rate p ; (b) average number of rounds N versus physical error rate p .

lookup table for a generic stabilizer code. The construction method also verifies the distinguishability of the fault set corresponding to flag circuits for syndrome measurements. Our construction also leads to the notion of the fault code, a linear code corresponding to the faults under circuit-level noise, which simplifies the verification of the distance of the protocol. More efficient decoding schemes for the fault code can be an interesting avenue to explore in future work.

Another optimization tool for space decoding is the MIM technique in Sec. III B, which could improve the decoding accuracy when the number of faults in the protocol is greater than t (where $t = \lfloor (d-1)/2 \rfloor$ for the code of distance d). The effect of the MIM technique on the simulated storage of the hexagonal color codes is discussed in Sec. V C (see also Fig. 14). We find that for any kind of time decoder, the logical error rates are reduced and the pseudothresholds are improved when applying the MIM technique, with greater improvements at larger distances.

For the time decoder, we generalize the adaptive syndrome-measurement technique from the previous work [38] (which is applicable to Shor-style error correction [1]) to flag FTQEC and develop one-tailed and two-tailed adaptive time decoders in Sec. IV B. For a general stabilizer code in which flag FTQEC is possible, the one-tailed decoder is preferable, as it is compatible with any fault-tolerant quantum computation, while the two-tailed decoder is applicable to quantum memory only. Nevertheless, for self-orthogonal CSS codes, the two-tailed decoder is applicable to any fault-tolerant computation built from Clifford gates and application of T gates by gate teleportation using high-fidelity magic states with the help of the classical processing technique on cumulative flag vectors developed in Sec. IV C. The effect of the adaptive time decoders on the simulated storage is discussed in Sec. V D. We observe that our adaptive time decoders can improve the pseudothresholds compared to the non-adaptive (Shor) time decoder while preserving the code distance. The two-tailed decoder also outperforms the one-tailed decoder.

The two-tailed adaptive decoder without MIM in this work is similar to the adaptive strong decoder in the previous work [38], except that this work uses flag circuits instead of syndrome-extraction circuits with cat states. The numerical results show that using flag circuits results in a 20–35% increase of the pseudothreshold for the hexagonal color codes of distances 3, 5, 7, and 9. This is mainly because flag circuits have fewer state-preparation and qubit-measurement locations, although they have more gates. The previous work [38] also assumes fault-tolerant preparation of cat states, which requires verification [1] or an ancilla decoding circuit [26], which can result in higher space and time overhead. Thus, the pseudothresholds could be worse in that case if additional requirements are also

considered. It should be noted that flag circuits may not outperform syndrome-extraction circuits with cat states in general, as flag FTQEC for other codes may require more complicated flag circuits.

We can further improve the performance of adaptive time decoders on self-orthogonal CSS codes by using the separate X - and Z -counting technique described in Sec. IV C. Here, we estimate the number of faults occurring from the measurement of generators of the first type (either X or Z) and then use that information in the measurement of generators of the second type. The effect of this technique can be found in Sec. V E. When the logical $|0\rangle$ state is stored, we find that the protocol that measures Z -type generators before X -type generators performs the best. We see no significant difference between the protocol that measures X -type generators before Z -type generators and the protocol that measures X - and Z -type generators jointly. Thus, the separated X and Z counting provides an advantage only for certain input states, depending on the measurement order.

Combining all techniques together, we find a significant improvement in the pseudothreshold while the code distance is still preserved. For example, on the hexagonal color code of distance 9, the pseudothreshold goes up from $(1.34 \pm 0.01) \times 10^{-4}$ to $(1.43 \pm 0.07) \times 10^{-3}$. We also find that in comparison with the unoptimized decoder, the crossing points between the codes of distances d and $d-2$ come much closer when all techniques are applied (as shown in Fig. 7), leading to a higher effective threshold \tilde{p}_{th} for this set of codes.

While our techniques are applicable to a broader family of codes, it would be interesting to see how our results compare with other works that study error decoding on the hexagonal color codes under circuit-level noise. For example, Baireuther *et al.* [60] have reported a pseudothreshold above 2×10^{-3} (against $p_L = p$ instead of $p_L = 2p/3$) with a neural-network decoder, which also preserves the code distance empirically. However, it has also been reported that training decoders for $d > 7$ have become too expensive. By adapting efficient color-decoding algorithms known as the restriction decoder [61] and the projection decoder [51], Chamberland *et al.* [32] and Beverland *et al.* [24] have reported threshold values of 2×10^{-3} and 3.7×10^{-3} , respectively. The difference between the threshold values is mostly contributed by different choices of syndrome-extraction circuits: for each weight-6 stabilizer generator, Ref. [32] has used three flag qubits for connectivity considerations, while Ref. [24] has not used any flag qubits. However, both the restriction decoder and the projection decoder can only correct up to $d/3$ errors (for examples of failure modes, see Fig. 15 in Sahay and Brown [62]) on the color-code family considered in this paper [63]. Recent preprints have reported distance-losing schemes to decode the color code with even higher

thresholds of 4.7×10^{-3} [25], and between 5×10^{-3} to 7×10^{-3} [64] without using flag qubits.

In contrast to the constructions that utilize the restriction decoder [32] and the projection decoder [24], our adaptive decoding method preserves the code distance (although the lookup table is not scalable to codes with larger distances). It is expected that our method could become advantageous for the codes of interest when the physical error rate is below a certain value. However, the noise models in Refs. [24,25,32,60,64] also consider idling noise, while our noise model does not. Sequential syndrome extraction is expected to perform poorly in architectures where idling noise is dominant (for an analysis on the $[[7, 1, 3]]$ code, see Appendix B). To improve performance, our methods need to be combined with optimized schedules specific to the given code family. CNOT schedule optimization is involved, requiring an enumeration of valid CNOT schedules satisfying basic constraints and finding the best-performing one using exhaustive search, similar to how Beverland *et al.* [24] have found a well-performing schedule for hexagonal color codes and bare ancillas. It is thus an open question what the error regime is where our flag-qubit-based adaptive methods are advantageous in comparison to the non-distance-preserving decoders. This analysis will require evaluation using code-specific optimizations under different-strength idling-noise scenarios, which we leave for future work.

Hierarchical decoding approaches also provide an interesting avenue to explore with lookup-table-based and adaptive techniques [65,66]. We conjecture that our techniques may result in efficient predecoders. The lookup tables and the adaptive syndrome algorithms would have to be restricted to local sections of topological codes or sparsely connected modules of other codes. Then, when the lookup-table decoders cannot decode the local problems, the more expensive and accurate decoder can attempt to decode the nonlocal problem.

It should be noted that this work uses the adaptive syndrome-measurement technique, which assumes fast qubit preparation and measurement. For the architectures on which qubit measurement and reset are slow, however, our method may require a large number of ancillas or may not be possible. In that case, one may consider using flag schemes that do not require fast qubit measurement and reset, such as the flag scheme for any distance-3 code [67] or the flag scheme in which the flag gadgets are constructed from the classical Bose-Chaudhuri-Hocquenghem (BCH) codes [68].

ACKNOWLEDGMENTS

The work was supported by the Office of the Director of National Intelligence—Intelligence Advanced Research

Projects Activity through an Army Research Office (ARO) contract (Grant No. W911NF-16-1-0082), the ARO Multidisciplinary University Research Initiative (MURI) program (Grant No. W911NF-16-1-0349), the ARO (Grant No. W911NF-21-1-0005), and the National Science Foundation Institute for Robust Quantum Simulation (QLCI Grant No. OMA-2120757).

APPENDIX A: MEMORY-FOOTPRINT SAVINGS IN THE LOOKUP TABLE

In this appendix, we detail how much saving each of the ideas in the main text contributes relative to the lookup-table memory cost of

$$M_{\text{stab}} = T_{\text{stab}}(4n - 2k) \text{ bits}, \quad (\text{A1})$$

when we look at a code as a generic stabilizer code. If CROs and the logical class are used instead of full Pauli operators for recovery, then instead of storing the full $2n$ bits for recovery, only $2k$ bits are required. Thus, $M_{\text{stab,CRO}} = T_{\text{stab}}(2n)$ leading to

$$M_{\text{stab,CRO}}/M_{\text{stab}} = \frac{T_{\text{stab}}(2n)}{T_{\text{stab}}(4n - 2k)} = \frac{1}{2 - k/n} = \frac{1}{2 - R}, \quad (\text{A2})$$

where $R = k/n$ is the encoding rate. Thus, for codes with $R \rightarrow 0$ as $n \rightarrow \infty$, this alone saves up to 50% in storage.

For CSS codes, as mentioned in the main text, we assume independent recovery for X - and Z -type errors. Let T_X and T_Z denote the number of unique X - and Z -type nontrivial syndromes (which are obtained by measuring X - and Z -type generators, respectively). Then, the CSS codes have two lookup tables mapping pure X - and Z -type syndromes to purely Z - and X -type recovery operators, with $T_X + 1$ and $T_Z + 1$ entries. Thus, they require only $2r_X$ and $2r_Z$ bits for the map key (a factor of 2 for flags and data bits) and n bits for the recovery operator. This results in the following memory cost for the lookup tables of a CSS code:

$$\begin{aligned} M_{\text{CSS}} &= (T_X + 1)(2r_X + n) + (T_Z + 1)(2r_Z + n) \text{ bits} \\ &= 2r_X T_X + 2r_X + nT_X + n + 2r_Z T_Z \\ &\quad + 2r_Z + nT_Z + n \text{ bits} \\ &= 2(r_X T_X + r_Z T_Z) + 2(r_X + r_Z) \\ &\quad + n(T_X + T_Z) + 2n \text{ bits} \\ &= 2(r_Z T_Z + r_X T_X) + (4n - 2k) + n(T_Z + T_X) \text{ bits}. \end{aligned} \quad (\text{A3})$$

To compare this with M_{stab} , we need to take care of the trivial syndrome and introduce a variable, T_{XZ} , for the number

of unique mixed X/Z syndromes that a generic stabilizer code representation would yield:

$$T_{\text{stab}} = T_X + T_Z + 1 + T_{XZ}. \quad (\text{A4})$$

Thus, from Eq. (A1),

$$M_{\text{stab}} = (T_X + T_Z + 1 + T_{XZ})(4n - 2k) \text{ bits}. \quad (\text{A5})$$

The ratio between the storage cost for a CSS code versus the cost when the same code is viewed as a generic stabilizer code is hard to bound precisely. Nevertheless, at least we know that $M_{\text{stab}} > M_{\text{CSS}}$, as the savings are

$$\begin{aligned} M_{\text{stab}} - M_{\text{CSS}} &= (T_X + T_Z + 1 + T_{XZ})(4n - 2k) \\ &\quad - (T_X + 1)(2r_X + n) + (T_Z + 1)(2r_Z + n) \\ &= 2(T_X r_Z + T_Z r_X) + n(T_X + T_Z) \\ &\quad + 2T_{XZ}(2n - k) \text{ bits}. \end{aligned} \quad (\text{A6})$$

If we use CROs, we can reduce the size of the map values to k bits from n :

$$\begin{aligned} M_{\text{CSS,CRO}} &= (T_X + 1)(2r_X + k) + (T_Z + 1)(2r_Z + k) \text{ bits} \\ &= 2r_X T_X + 2r_X + kT_X + k + 2r_Z T_Z \\ &\quad + 2r_Z + kT_Z + k \text{ bits} \\ &= 2(r_X T_X + r_Z T_Z) + 2(r_X + r_Z) \end{aligned}$$

$$+ k(T_X + T_Z) + 2k \text{ bits}$$

$$= 2(r_X T_X + r_Z T_Z) + 2n + k(T_Z + T_X) \text{ bits}. \quad (\text{A7})$$

And thus, not surprisingly, the decrease in bits is

$$\begin{aligned} M_{\text{CSS}} - M_{\text{CSS,CRO}} \\ = (n - k)(2 + T_Z + T_X) = (n - k)T_{\text{CSS}}, \end{aligned} \quad (\text{A8})$$

where T_{CSS} is the total number of entries of the two tables.

If the code is self-orthogonal, then the two tables coincide, $T := T_X = T_Z$, $r := r_X = r_Z$. Thus,

$$\begin{aligned} M_{\text{CSS,CRO,SO}} &= \frac{1}{2} (2(r_Z T_Z + r_X T_X) + 2n + k(T_Z + T_X)) \\ &= \frac{1}{2} (2(rT + rT) + 2n + k(T + T)) \\ &= \frac{1}{2} (4rT + 2n + 2kT) \\ &= 2rT + n + kT \\ &= (n - k)T + n + kT \\ &= nT + n \\ &= n(T + 1), \end{aligned} \quad (\text{A9})$$

which is consistent with having $T + 1$ entries, with a map key of $n - k$ bits (with $(n - k)/2$ for flags and for generator bits) and a value with k bits. If we were to view the

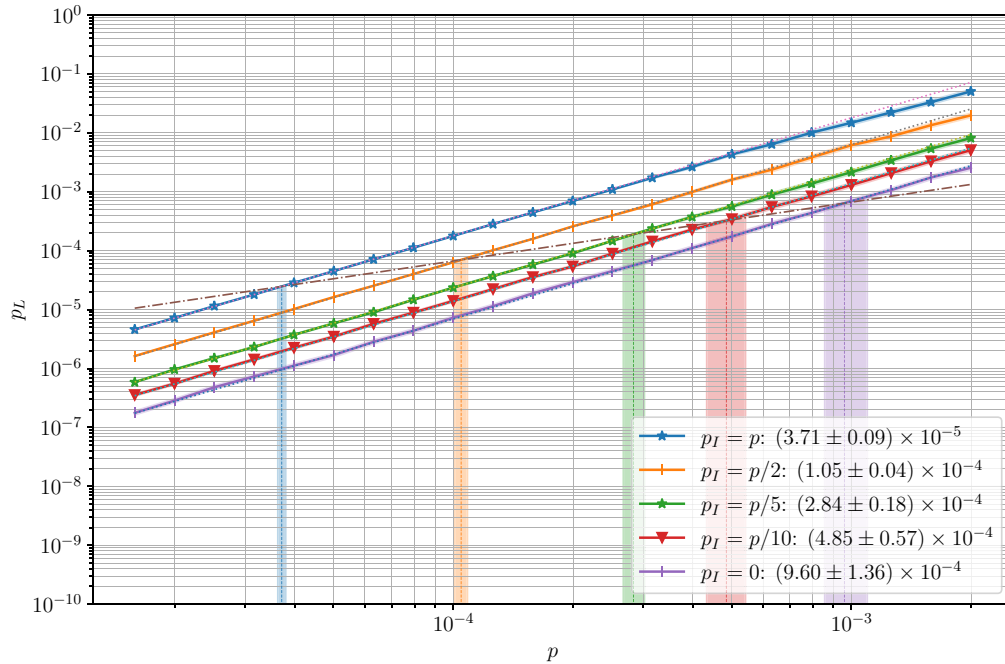


FIG. 11. The effect of idling noise on a naive CNOT schedule for the $[[7, 1, 3]]$ code at different idling-noise strengths p_I relative to the gate errors p . In this setup, $p_I = p$ is the full standard depolarizing noise model and $p_I = 0$ is the one that we have used to evaluate our methods in the main text, while $p_I = p/2$, $p_I = p/5$, and $p_I = p/10$ are between those two extremes.

self-orthogonal CSS code as a generic stabilizer code, we would obtain $T_{\text{stab}} = 1 + 2T + T_{XZ}$ and thus the ratio is

$$\begin{aligned} M_{\text{CSS,CRO,SO}}/M_{\text{stab}} &= \frac{n(T+1)}{T_{\text{stab}}(4n-2k)} \\ &= \frac{n(T+1)}{(1+2T+T_{XZ})(4n-2k)} \\ &= \frac{T+1}{(1+2T+T_{XZ})(4-2R)} \leq \frac{1}{8-4R}, \end{aligned} \quad (\text{A10})$$

where we have used the fact that T_{XZ} must be at least 1 for $t > 0$ and a nontrivial encoding. This upper bound means that at a zero rate code, leveraging the structure of a self-orthogonal CSS code and the CROs can create a memory footprint less than 12.5% that of the memory footprint of a lookup table if we view the code as a stabilizer code.

APPENDIX B: THE EFFECT OF IDLING NOISE

To demonstrate the effect of idling noise, we evaluate the $[[7, 1, 3]]$ code under a naive interleaved schedule, depicted in Fig. 12(a) without noise terms and in Fig. 12(b) with gate-noise terms with strength $p = 0.02$ and idling-noise terms with strength $p_I = 0.01$. Note that further improvements are possible to reduce idling in the circuit by doubling the number of flag qubits and ancilla qubits

and measuring X - and Z -stabilizer generators in parallel, similar to the scheme by Beverland *et al.* [24]. This will, however, be only possible for the two-tailed adaptive decoder and the separate X/Z decoder will not work by definition. Also, protocol-specific CNOT schedule optimization might be possible, depending on the underlying quantum code. As we are not aiming to find tools on the code level, this investigation is beyond the scope of this paper. It is also interesting to point out that the use of a single flag ancilla and a single syndrome ancilla forces sequential execution of the gates within a generator, while multi-flag-based schemes such as in the work of Chamberland *et al.* [32] allow for multiple CNOT gates to be executed in the same time step. While our methods here use single-flag-qubit-based protocols, that angle can be relaxed if the strength of the idling noise requires it.

Our numerical evaluation results, displayed in Fig. 11, show that at idling-noise strength $p_I = p$, the pseudothreshold is 20–25 times smaller than the case without idling noise, $p_I = 0$. However, as the relative strength of the depolarizing noise p/p_I increases, the performance rapidly approaches the ideal case. Furthermore, we can see that our decoders still preserve the distance, which is expected given that the single-qubit depolarizing noise terms do not change the set of errors to be corrected but only change the strength of some terms.

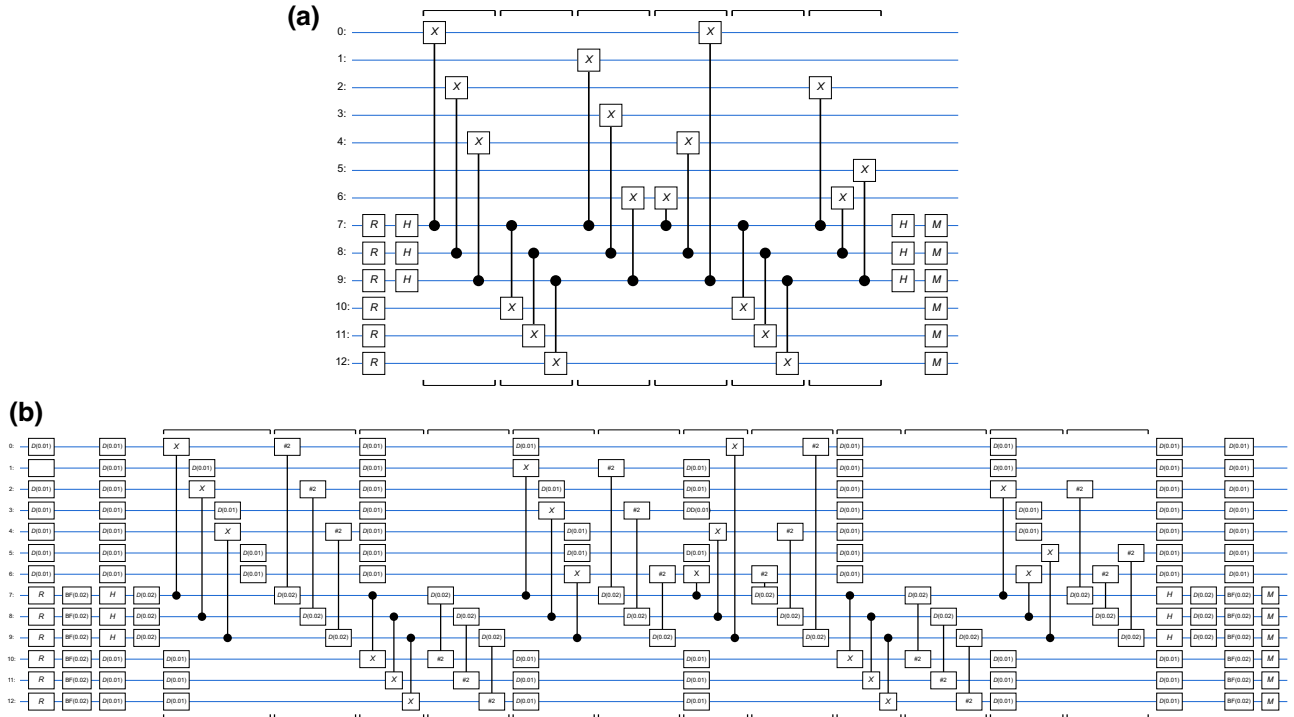


FIG. 12. An interleaved schedule for extracting the Z syndrome of the $[[7, 1, 3]]$ code (a) without and (b) with noise terms at gate depolarizing strength $p = 0.02$ and idling-noise strength $p_I = p/2 = 0.01$. The data qubits are 0–6, the ancilla qubits are 7–9, and the flag qubits are 10 and 11. Brackets above and below the circuit group together gates that are executed during the same time step. X -type syndrome extraction is similar.

APPENDIX C: FIGURES FOR ALL DISTANCES

In this appendix, we provide more details on the plots of logical error rate p_L versus physical error rate p and the plots of average number of rounds N versus physical error rate p for hexagonal color codes of distance 3, 5, 7, and 9 when different combinations of optimization tools are used.

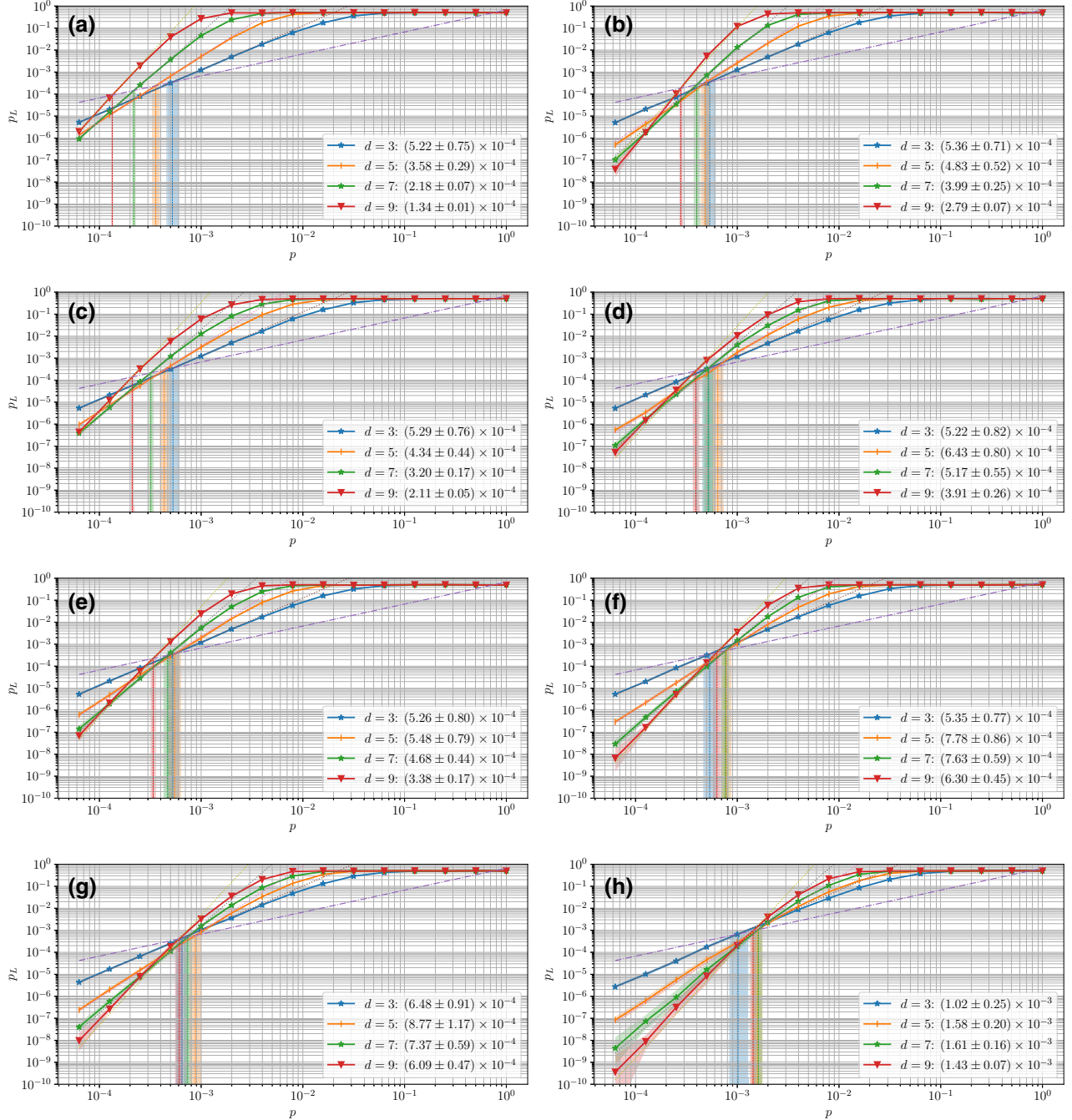


FIG. 13. The threshold-formation effect of increasingly better space and time decoders: (a) Shor without MIM; (b) Shor with MIM; (c) one-tailed without MIM; (d) one-tailed with MIM; (e) two-tailed without MIM; (f) two-tailed with MIM; (g) two-tailed XZ with MIM; (h) two-tailed ZX with MIM. Both space-decoding improvements (MIM) and time-decoding improvements (from Shor to two-tailed ZX strategy) help in making the intersections of the p_L versus p curves more focused.

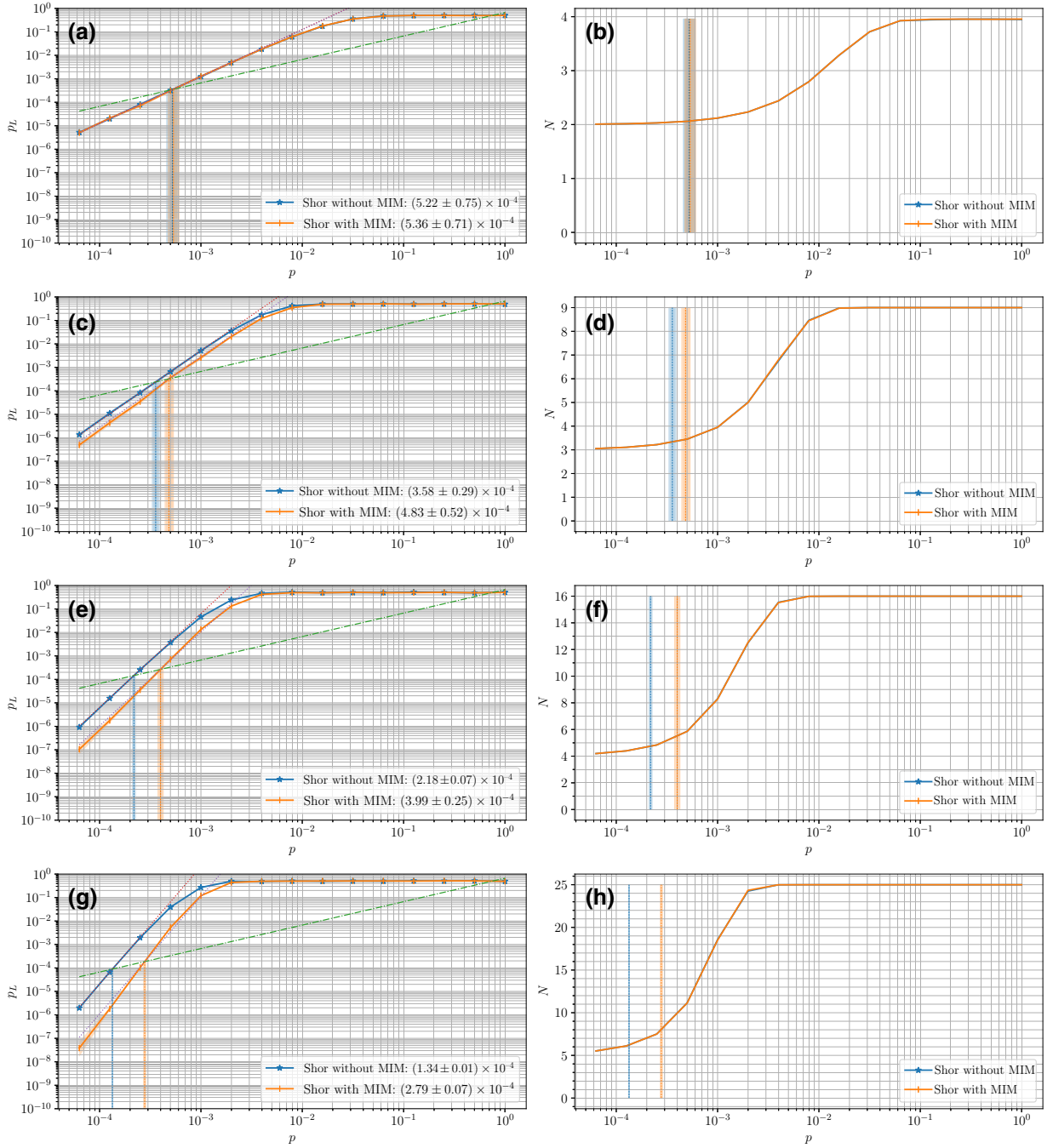


FIG. 14. The effect of the MIM technique on the Shor time decoder for hexagonal color codes of various distances: (a),(c),(e),(g) logical error rate p_L versus physical error rate p ; (b),(d),(f),(h) average number of rounds N versus physical error rate p ; (a),(b) $d = 3$; (c),(d) $d = 5$; (e),(f) $d = 7$; (g),(h) $d = 9$. The improvement is increasing with the code distance, with no improvement at $d = 3$ and the biggest one at $d = 9$.

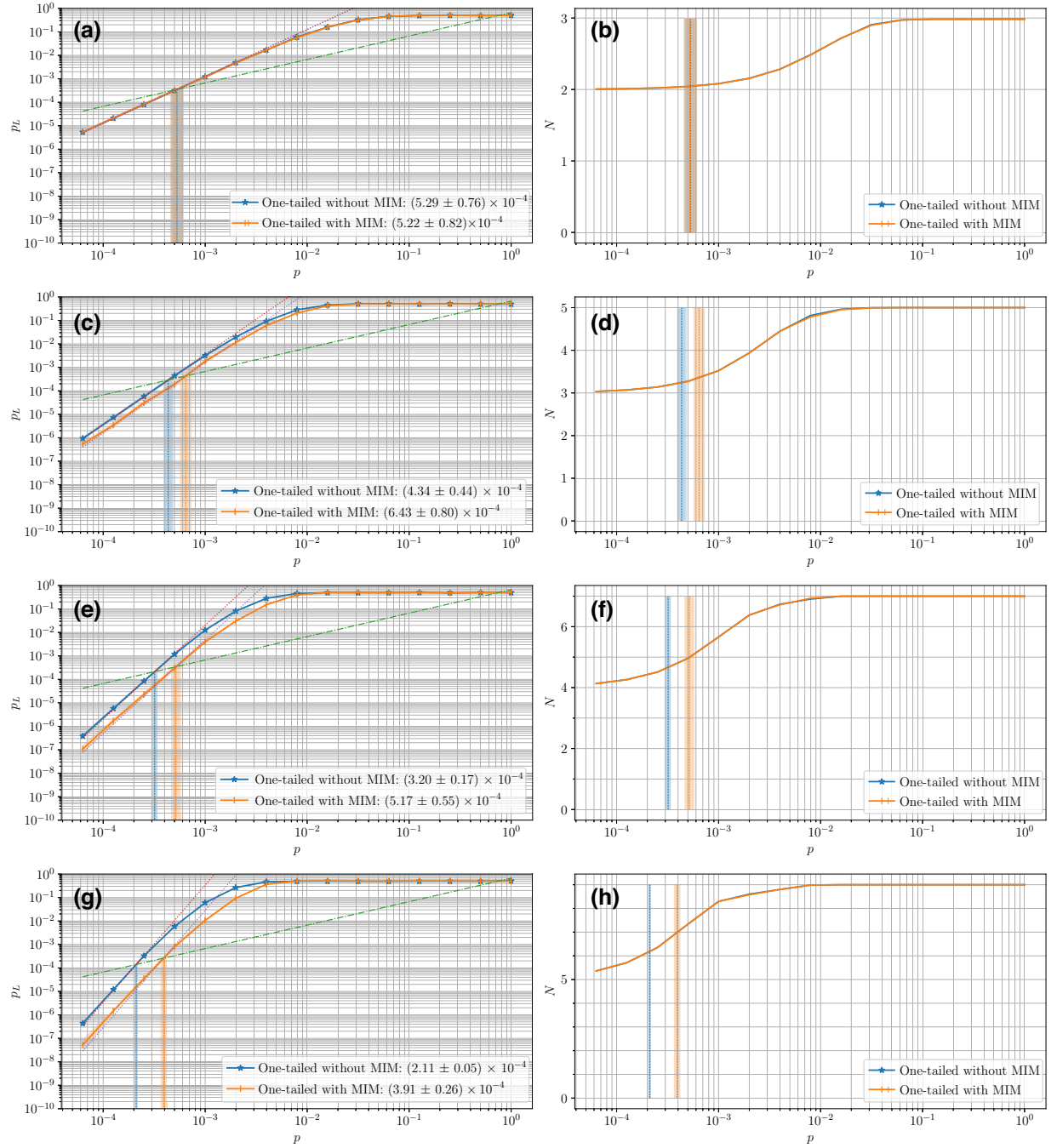


FIG. 15. The effect of the MIM technique on the one-tailed adaptive time decoder for hexagonal color codes of various distances: (a),(c),(e),(g) logical error rate p_L versus physical error rate p ; (b),(d),(f),(h) average number of rounds N versus physical error rate p ; (a),(b) $d = 3$; (c),(d) $d = 5$; (e),(f) $d = 7$; (g),(h) $d = 9$. The improvement is increasing with distance, with no improvement at $d = 3$ and the biggest one at $d = 9$.

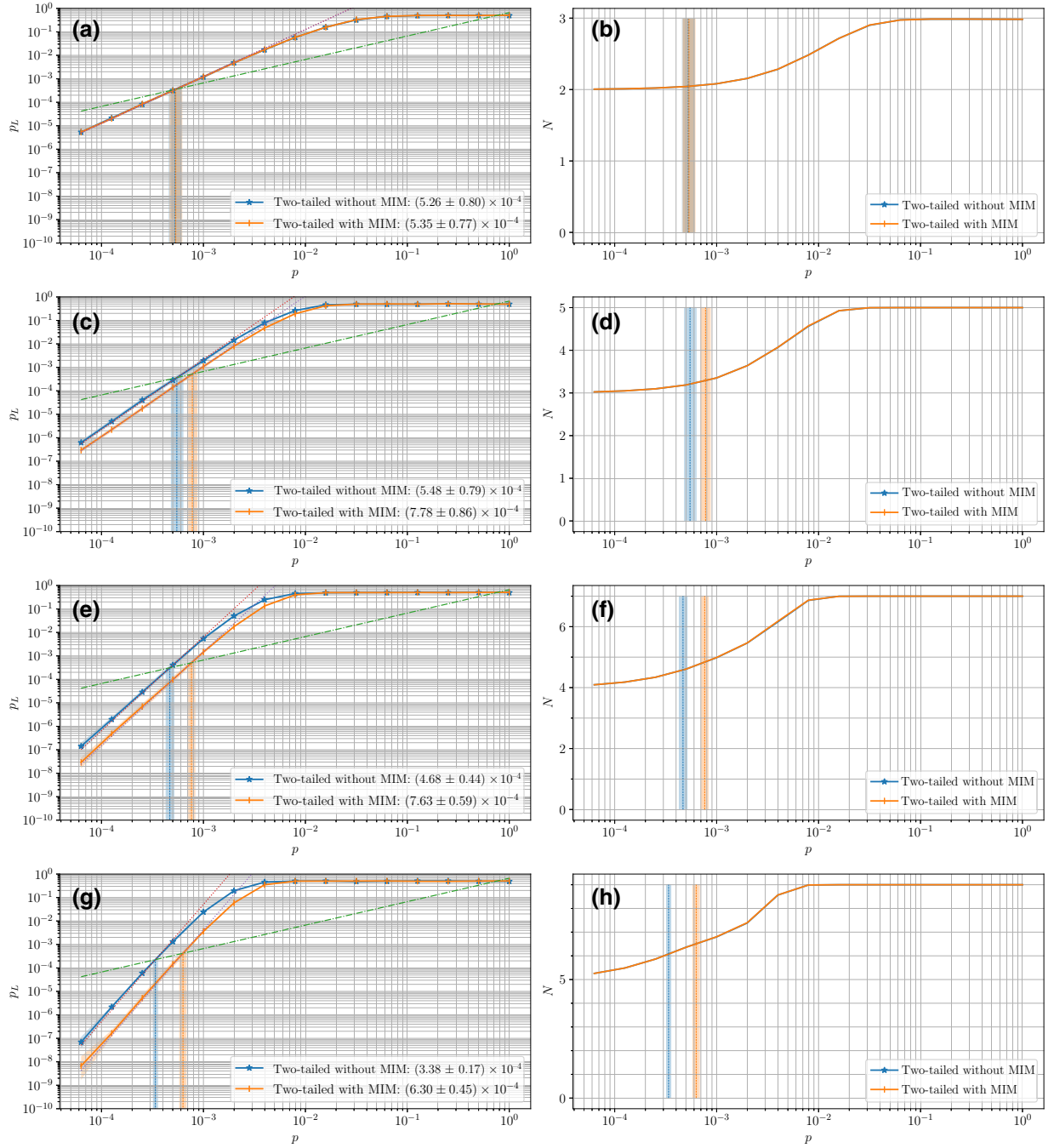


FIG. 16. The effect of the MIM technique on the two-tailed decoder for hexagonal color codes of various distances: (a),(c),(e),(g) logical error rate p_L versus physical error rate p ; (b),(d),(f),(h) average number of rounds N versus physical error rate p ; (a),(b) $d = 3$; (c),(d) $d = 5$; (e),(f) $d = 7$; (g),(h) $d = 9$. The improvement is increasing with distance, with no improvement at $d = 3$ and the biggest one at $d = 9$.

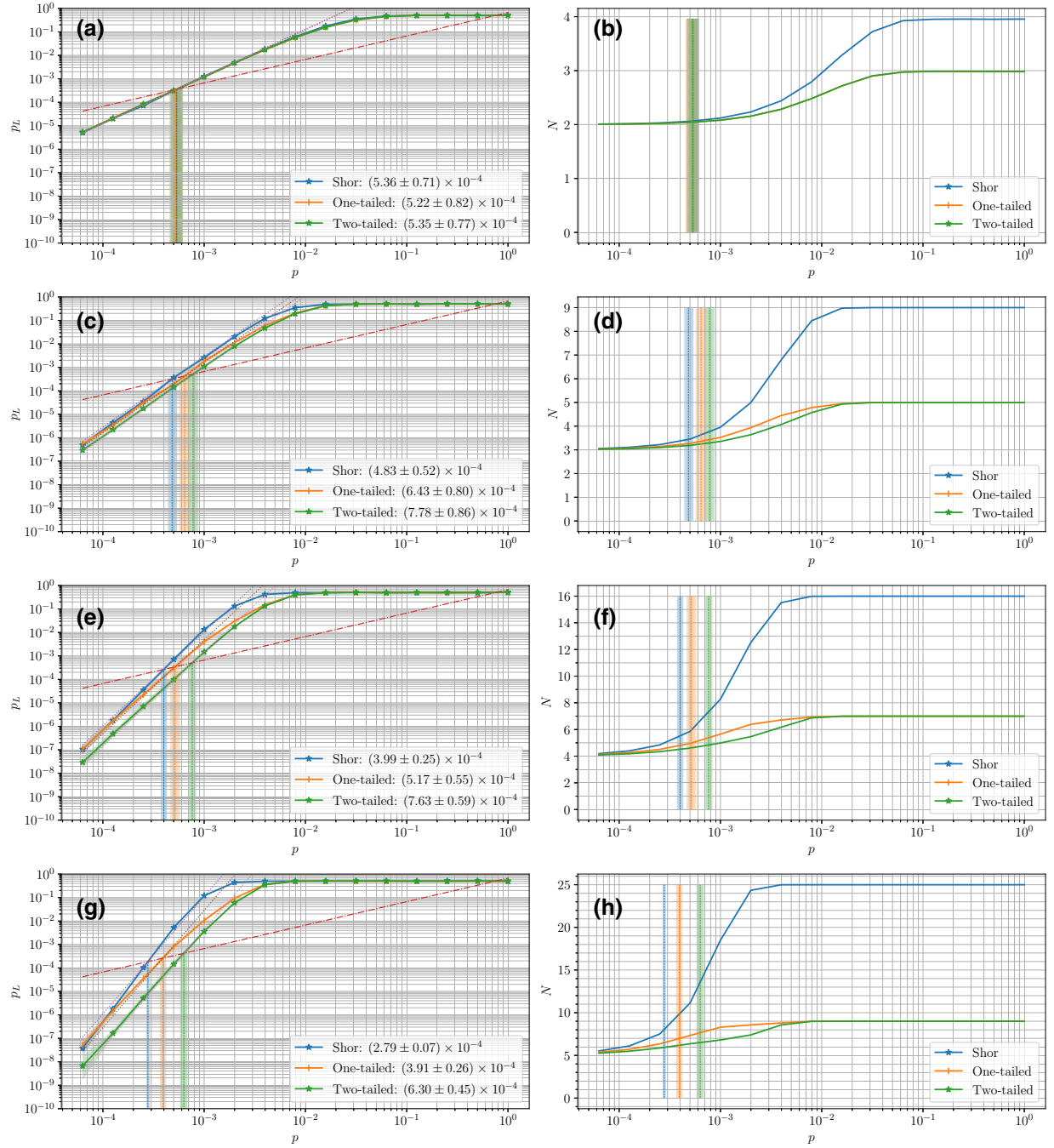


FIG. 17. A comparison of the one-tailed and two-tailed adaptive time decoders with the Shor time decoder for hexagonal color codes of various distances: (a),(c),(e),(g) logical error rate p_L versus physical error rate p ; (b),(d),(f),(h) average number of rounds N versus physical error rate p ; (a),(b) $d = 3$; (c),(d) $d = 5$; (e),(f) $d = 7$; (g),(h) $d = 9$. The improvement is increasing with distance, with no improvement at $d = 3$ and the biggest one at $d = 9$. Here, all decoders use MIM-enhanced space decoding.

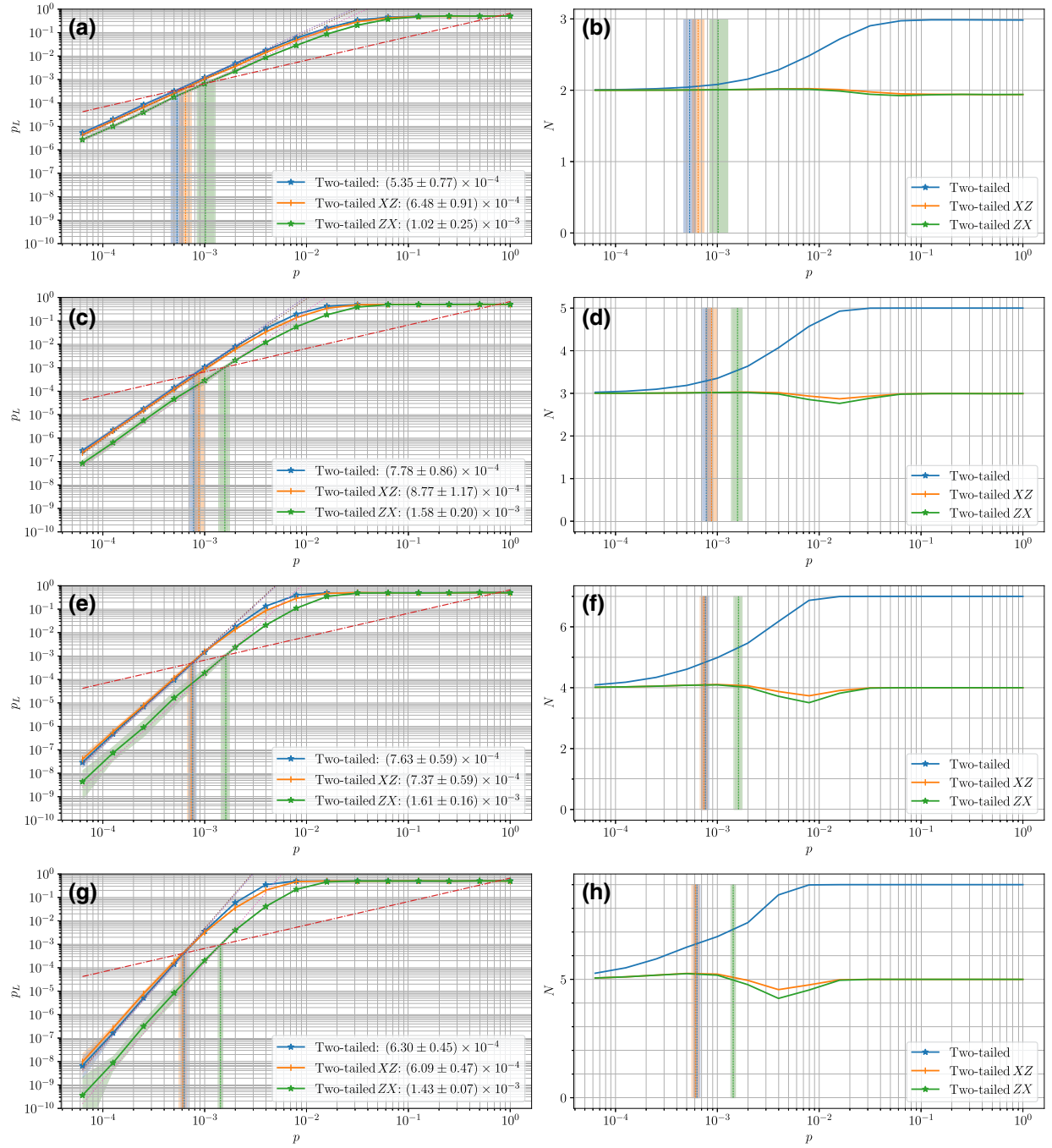


FIG. 18. A comparison of the two-tailed time decoder with joint measurements and the two-tailed time decoders with the XZ and ZX strategies for hexagonal color codes of various distances: (a),(c),(e),(g) logical error rate p_L versus physical error rate p ; (b),(d),(f),(h) average number of rounds N versus physical error rate p ; (a),(b) $d=3$; (c),(d) $d=5$; (e),(f) $d=7$; (g),(h) $d=9$. Here, all decoders use space decoding with MIM.

- [1] P. W. Shor, in *37th Annual Symposium on Foundations of Computer Science* (IEEE, Burlington, VT, USA, 1996), p. 56.
- [2] E. Knill, R. Laflamme, and W. H. Zurek, Resilient quantum computation: Error models and thresholds, *Proc. R. Soc. London, Ser. A* **454**, 365 (1998).
- [3] A. Y. Kitaev, Quantum computations: Algorithms and error correction, *Russ. Math. Surv.* **52**, 1191 (1997).
- [4] J. Preskill, Reliable quantum computers, *Proc. R. Soc. London, Ser. A* **454**, 385 (1998).
- [5] D. Aharonov and M. Ben-Or, Fault-tolerant quantum computation with constant error rate, *SIAM J. Comput.* **38**, 1207 (2008).
- [6] P. Aliferis, D. Gottesman, and J. Preskill, Quantum accuracy threshold for concatenated distance-3 codes, *Quantum Inf. Comput.* **6**, 97 (2006).
- [7] D. Aharonov, A. Kitaev, and J. Preskill, Fault-tolerant quantum computation with long-range correlated noise, *Phys. Rev. Lett.* **96**, 050504 (2006).
- [8] A. M. Steane, Overhead and noise threshold of fault-tolerant quantum error correction, *Phys. Rev. A* **68**, 042322 (2003).
- [9] A. Paetznick and B. W. Reichardt, Fault-tolerant ancilla preparation and noise threshold lower bounds for the 23-qubit Golay code, *Quantum Inf. Comput.* **12**, 1034 (2012).
- [10] C. Chamberland, T. Jochym-O'Connor, and R. Laflamme, Overhead analysis of universal concatenated quantum codes, *Phys. Rev. A* **95**, 022313 (2017).
- [11] R. Takagi, T. J. Yoder, and I. L. Chuang, Error rates and resource overheads of encoded three-qubit gates, *Phys. Rev. A* **96**, 042302 (2017).
- [12] C. Gidney and M. Ekerå, How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits, *Quantum* **5**, 433 (2021).
- [13] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vasilillo, Assessing requirements to scale to practical quantum advantage, [ArXiv:2211.07629](https://arxiv.org/abs/2211.07629).
- [14] D. Gottesman, Ph.D. thesis, California Institute of Technology, 1997, <https://doi.org/10.7907/rzr7-dt72>.
- [15] A. Steane, Multiple-particle interference and quantum error correction, *Proc. R. Soc. London, Ser. A* **452**, 2551 (1996).
- [16] E. Knill, Scalable quantum computing in the presence of large detected-error rates, *Phys. Rev. A* **71**, 042322 (2005).
- [17] Y. Tomita and K. M. Svore, Low-distance surface codes under realistic quantum noise, *Phys. Rev. A* **90**, 062320 (2014).
- [18] S. Huang and K. R. Brown, Fault-tolerant compass codes, *Phys. Rev. A* **101**, 042312 (2020).
- [19] M. Li, M. Gutiérrez, S. E. David, A. Hernandez, and K. R. Brown, Fault tolerance with bare ancillary qubits for a $[[7, 1, 3]]$ code, *Phys. Rev. A* **96**, 032341 (2017).
- [20] S. Bravyi, G. Duclos-Cianci, D. Poulin, and M. Suchara, Subsystem surface codes with three-qubit check operators, *Quantum Inf. Comput.* **13**, 963 (2013).
- [21] M. Li, D. Miller, and K. R. Brown, Direct measurement of Bacon-Shor code stabilizers, *Phys. Rev. A* **98**, 050301 (2018).
- [22] O. Higgott and N. P. Breuckmann, Subsystem codes with high thresholds by gauge fixing and reduced qubit overhead, *Phys. Rev. X* **11**, 031039 (2021).
- [23] A. J. Landahl, J. T. Anderson, and P. R. Rice, Fault-tolerant quantum computing with color codes, [ArXiv:1108.5738](https://arxiv.org/abs/1108.5738).
- [24] M. E. Beverland, A. Kubica, and K. M. Svore, Cost of universality: A comparative study of the overhead of state distillation and code switching with color codes, *PRX Quantum* **2**, 020341 (2021).
- [25] J. Zhang, Y.-C. Wu, and G.-P. Guo, Facilitating practical fault-tolerant quantum computing based on color codes, [ArXiv:quant-ph/2309.05222](https://arxiv.org/abs/2309.05222).
- [26] D. P. DiVincenzo and P. Aliferis, Effective fault-tolerant quantum computation with slow measurements, *Phys. Rev. Lett.* **98**, 020501 (2007).
- [27] T. J. Yoder and I. H. Kim, The surface code with a twist, *Quantum* **1**, 2 (2017).
- [28] R. Chao and B. W. Reichardt, Quantum error correction with only two extra qubits, *Phys. Rev. Lett.* **121**, 050502 (2018).
- [29] R. Chao and B. W. Reichardt, Flag fault-tolerant error correction for any stabilizer code, *PRX Quantum* **1**, 010302 (2020).
- [30] C. Chamberland and M. E. Beverland, Flag fault-tolerant error correction with arbitrary distance codes, *Quantum* **2**, 53 (2018).
- [31] T. Tansuwanont, C. Chamberland, and D. Leung, Flag fault-tolerant error correction, measurement, and quantum computation for cyclic Calderbank-Shor-Steane codes, *Phys. Rev. A* **101**, 012342 (2020).
- [32] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, Triangular color codes on trivalent graphs with flag qubits, *New J. Phys.* **22**, 023019 (2020).
- [33] C. Chamberland, G. Zhu, T. J. Yoder, J. B. Hertzberg, and A. W. Cross, Topological and subsystem codes on low-degree graphs with flag qubits, *Phys. Rev. X* **10**, 011022 (2020).
- [34] T. Tansuwanont and D. Leung, Fault-tolerant quantum error correction using error weight parities, *Phys. Rev. A* **104**, 042410 (2021).
- [35] T. Tansuwanont and D. Leung, Achieving fault tolerance on capped color codes with few ancillas, *PRX Quantum* **3**, 030322 (2022).
- [36] C. Zalka, Threshold estimate for fault tolerant quantum computation, [ArXiv:quant-ph/9612028](https://arxiv.org/abs/quant-ph/9612028).
- [37] N. Delfosse and B. W. Reichardt, Short Shor-style syndrome sequences, [ArXiv:2008.05051](https://arxiv.org/abs/2008.05051).
- [38] T. Tansuwanont, B. Pato, and K. R. Brown, Adaptive syndrome measurements for Shor-style error correction, *Quantum* **7**, 1075 (2023).
- [39] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys.* **43**, 4452 (2002).
- [40] C. Chamberland, A. Kubica, T. J. Yoder, and G. Zhu, Triangular color codes on trivalent graphs with flag qubits, [ArXiv:1911.00355v3](https://arxiv.org/abs/1911.00355v3).
- [41] H. Bombin and M. A. Martin-Delgado, Topological quantum distillation, *Phys. Rev. Lett.* **97**, 180501 (2006).
- [42] A. R. Calderbank and P. W. Shor, Good quantum error-correcting codes exist, *Phys. Rev. A* **54**, 1098 (1996).

- [43] N. Delfosse and J.-P. Tillich, in *2014 IEEE International Symposium on Information Theory* (IEEE, Honolulu, HI, USA, 2014), p. 1071.
- [44] A. M. Steane, Active stabilization, quantum computation, and quantum state synthesis, *Phys. Rev. Lett.* **78**, 2252 (1997).
- [45] S. Huang and K. R. Brown, Between Shor and Steane: A unifying construction for measuring error syndromes, *Phys. Rev. Lett.* **127**, 090505 (2021).
- [46] D. Poulin, Optimal and efficient decoding of concatenated quantum block codes, *Phys. Rev. A* **74**, 052333 (2006).
- [47] P. Das, A. Locharla, and C. Jones, in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (ACM, Lausanne, Switzerland, 2022), p. 541.
- [48] S. Huang and K. R. Brown, Constructions for measuring error syndromes in Calderbank-Shor-Steane codes between Shor and Steane methods, *Phys. Rev. A* **104**, 022429 (2021).
- [49] L. Berent, L. Burgholzer, P.-J. H. S. Derks, J. Eisert, and R. Wille, Decoding quantum color codes with MaxSAT, *ArXiv:2303.14237*.
- [50] N. Maskara, A. Kubica, and T. Jochym-O'Connor, Advantages of versatile neural-network decoding for topological codes, *Phys. Rev. A* **99**, 052351 (2019).
- [51] N. Delfosse, Decoding color codes by projection onto surface codes, *Phys. Rev. A* **89**, 012317 (2014).
- [52] O. Higgott and C. Gidney, Sparse Blossom: Correcting a million errors per core second with minimum-weight matching, *ArXiv:2303.15933*.
- [53] A. M. Tillmann and M. E. Pfetsch, The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing, *IEEE Trans. Inf. Theory* **60**, 1248 (2013).
- [54] D. Tucket, QECSIM is a PYTHON 3 package for simulating quantum error correction using stabilizer codes (2021), <https://github.com/qecsim/qecsim>.
- [55] R. E. Korf, in *Encyclopedia of Information Systems*, edited by H. Bidgoli (Elsevier, New York, 2003), p. 31.
- [56] S. Bravyi and A. Kitaev, Universal quantum computation with ideal Clifford gates and noisy ancillas, *Phys. Rev. A* **71**, 022316 (2005).
- [57] S. Bravyi and J. Haah, Magic-state distillation with low overhead, *Phys. Rev. A* **86**, 052329 (2012).
- [58] C. Gidney, STIM: A fast stabilizer circuit simulator, *Quantum* **5**, 497 (2021).
- [59] CIRQ Developers, CIRQ v.1.1.0, Full list of authors: <https://github.com/quantumlib/Cirq>, Zenodo (2022), <https://doi.org/10.5281/zenodo.10247207>.
- [60] P. Baireuther, M. Caio, B. Criger, C. W. Beenakker, and T. E. O'Brien, Neural network decoder for topological color codes with circuit level noise, *New J. Phys.* **21**, 013003 (2019).
- [61] A. Kubica and N. Delfosse, Efficient color code decoders in $d \geq 2$ dimensions from toric code decoders, *ArXiv:1905.07393*.
- [62] K. Sahay and B. J. Brown, Decoder for the triangular color code by matching on a Möbius strip, *PRX Quantum* **3**, 010310 (2022).
- [63] See Appendix A of the third arXiv version of Ref. [32].
- [64] C. Gidney and C. Jones, New circuits and an open source decoder for the color code, *ArXiv:2312.08813*.
- [65] N. Delfosse, Hierarchical decoding to reduce hardware requirements for quantum computing, *ArXiv:2001.11427*.
- [66] S. C. Smith, B. J. Brown, and S. D. Bartlett, Local predecoder to reduce the bandwidth and latency of quantum error correction, *Phys. Rev. Appl.* **19**, 034050 (2023).
- [67] P. Prabhu and B. W. Reichardt, Fault-tolerant syndrome extraction and cat state preparation with fewer qubits, *ArXiv:2108.02184*.
- [68] B. Anker and M. Marvian, Flag gadgets based on classical codes, *ArXiv:2212.1073*.