

Soft information decoding with superconducting qubits

Maurice D. Hanisch^{1,2}, Bence Hetényi^{1,*}, and James R. Wootton^{1,3,†}

¹IBM Quantum, *IBM Research—Zürich*, 8803 Rüschlikon, Switzerland

²Institute for Theoretical Physics, *ETH Zürich*, 8093 Zürich, Switzerland

³Center for Quantum Computing and Quantum Coherence (QC2),
University of Basel, Petersplatz 1, 4001 Basel, Switzerland



(Received 26 June 2025; accepted 22 April 2026; published 8 May 2026)

Quantum error correction promises a viable path to fault-tolerant computations, enabling exponential error suppression when the device's error rates remain below the protocol's threshold. However, this threshold strongly depends on the classical method used to decode the measurements of the syndrome. These classical algorithms traditionally only interpret binary data, ignoring valuable information contained in complete analog measurement data. We present the first large-scale experimental verification of soft decoding by implementing repetition codes up to distance 51 on superconducting hardware. Unlike previous demonstrations limited to small code distances, this extensive range allows us to confirm a fundamental improvement in the exponential scaling of error suppression rather than a mere constant-factor reduction in error rates. We observe a 25% increase in the threshold, which yields logical error rates up to 30 times lower than standard hard decoding. Analyzing the trade-off between information volume and decoding performance, we show that a single byte of information per measurement suffices to reach optimal decoding. This underscores the effectiveness and practicality of soft decoding on hardware, including in time-sensitive contexts such as real-time decoding.

DOI: [10.1103/y9fh-4x6n](https://doi.org/10.1103/y9fh-4x6n)

I. INTRODUCTION

Quantum error correction (QEC) is fundamentally recognized as a central strategy for deploying fault-tolerant computations at scale. It promises exponential error suppression, closing the gap between attainable error rates and those required for practical quantum computing. In recent years, significant progress has been achieved in quantum error correction experiments, demonstrating its emerging capabilities [1–7].

These advances are particularly evident in the implementation of the surface code, a leading candidate for fault-tolerant quantum computation. Progress has been marked by an upward path from the implementation of a distance-three ($d = 3$) surface code [1,2], advancing to a distance-five ($d = 5$) surface code [3], culminating in a distance-seven ($d = 7$) surface code that has also shown evidence for improvements in the fidelities of logical entangling gates with increasing code distance [4]. Furthermore, recent experiments claim to have achieved breakeven error correction

*Contact author: bence.hetenyi@ibm.com

†Contact author: james.wootton@unibas.ch

performance, matching or surpassing their physical counterparts. Key milestones include achieving breakeven fidelity in the initialization of logical qubits within a color code [4], preparing logical Bell pairs with error rates 800 times lower than their physical level [5], and successfully distilling magic states with fidelity beyond the breakeven point [6]. Furthermore, in recent experiments, below surface threshold performance was achieved on a distance-seven ($d = 7$) code [7]. These claims highlight quantum error correction’s growing effectiveness and practical relevance.

The success of quantum error correction experiments hinges on several critical factors: the choice of quantum code, the error rates of gate operations and midcircuit measurements, and the performance of the classical decoding algorithm used to process measurement outcomes and determine corrective operations. Usually, outcomes from midcircuit measurements, inherently analog, are converted into binary data indicating the presence or absence of an error before being analyzed by the decoder. This conversion inevitably results in a loss of information that could affect the accuracy of decoding.

Pattison *et al.* developed a method to avoid this loss by incorporating the full range of analog data, also known as soft information, into the decoding process [8] with a weighting algorithm. Their simulations indicated that using soft information in surface code decoding can significantly raise the code’s threshold compared to decoders that utilize only binary, or “hard,” information. This result reveals a substantial improvement in the exponential scaling of error suppression, which goes beyond just a constant benefit. The findings underscore the importance of utilizing all available information from the experimental setup, thus optimizing quantum error correction strategies.

In addition to the work by Pattison *et al.*, the benefit of employing soft information has been validated through simulations using neural network decoders [9,10]. On the hardware front, soft decoding has been implemented in different settings: to improve readout fidelity on a Si/SiGe two-qubit system [11], and to superconducting qubits using both a distance-three ($d = 3$) heavy-hexagon code [12] and, recently, a distance-three ($d = 3$) two-dimensional repetition code [13]. However, these hardware experiments were limited to fixed, small code distances (at most $d = 3$). Crucially, such small-scale experiments cannot distinguish between a constant-factor improvement in fidelity and a true improvement in the threshold. To verify the latter, one must observe the logical error rate across a wide range of code distances to demonstrate a change in the exponential scaling behavior. Consequently, the anticipated threshold benefit of soft decoding has remained experimentally unverified in large-scale systems.

The experiments of Ref. [12] involving $T = 10$ rounds of stabilizer measurements managed to postselect for leakage errors—transitions to higher-energy states detrimental to quantum error correction [14]. This postselection was feasible because the low number of measurement rounds helped restrict the accumulation of leakage, which typically increases with the duration of the experiment. Furthermore, the experiments of Ref. [13] that used Pattison *et al.*’s weighting method did not explicitly treat leakage. Consequently, the leakage issue and its impact on soft decoding with weighting algorithms, also not addressed in the initial studies by Pattison *et al.* or subsequent simulation efforts, remain a critical area for further investigation.

This work provides the first large-scale experimental verification of the soft decoding threshold benefit. By implementing repetition codes ranging from $d = 3$ to $d = 51$, we move beyond constant-factor comparisons and successfully extract the exponential suppression of logical error rates. This allows us to confirm that soft information fundamentally alters the scaling of error suppression on physical hardware. Our experiments were conducted on IBM Quantum’s 127-qubit device, IBM Sherbrooke. We used this device via the standard IBM Quantum Cloud interface without specific calibrations to improve the accuracy of the specified error correction experiment. Rather than focusing on the lowest logical error rates the hardware is capable of, we sought to compare how incorporating soft information affects the decoding process and improves overall logical error rates. Additionally, given that soft information error correction involves processing and transmitting a greater volume of information, we studied how the performance of soft decoding is influenced when we retain less information from measurements. This analysis addresses the practicality of soft information decoding within a realistic quantum error correction framework such as real-time

decoding. Finally, we explore the impact of significantly increasing the number of rounds, up to 100 rounds of stabilizer measurements, to assess how leakage affects the error rates.

II. DECODING REPETITION CODES WITH SOFT INFORMATION

A. Repetition codes

Our study concerns the distance- d one-dimensional repetition code. We implement the code on a linear chain of qubits, utilizing every second qubit as an ancilla to measure the parity between neighboring code qubits. Hence, a repetition code of distance d requires $2d - 1$ qubits. We measure the $Z_i Z_{i+1}$ stabilizers by applying a pair of CNOT gates between the data qubits and the ancilla. If a bit flip occurs on adjacent data qubits, the CNOT gates flip the corresponding ancilla qubit and we detect the error by measuring the ancilla in the Z basis.

We do not reset the ancillary qubits to avoid idling errors and the complexity of active reset operations [15]. Instead, we compute the changes in outcomes in the measurement to simulate a scenario as if resets had been performed. Let $\mathbf{m}_t^{-r}$ be the results of the measurement at time t without resetting the ancilla qubits. We can then write the *corrected* measurement results at time t as

$$\mathbf{m}_t = \mathbf{m}_t^{-r} \oplus \mathbf{m}_{t-1}^{-r}. \quad (1)$$

B. Decoding

We decode the repetition code using the minimum-weight perfect matching (MWPM) algorithm [16, 17]. Under circuit-level Pauli noise, the relationship between possible errors and syndromes can be represented as a *decoding graph*, where detectors are nodes and potential errors are edges. The matching decoder finds the most likely set of errors by solving for the minimum-weight set of edges on a graph where edge weights $w = \ln((1 - p)/p)$ are calculated using the probability p of the corresponding error event.

We typically evaluate the merits of a quantum code based on its *threshold value* p_{th} , defined as the upper limit of error probability at which the code can still suppress errors. Well below the threshold, the logical error rate ϵ_L decreases exponentially with the distance of the code as

$$\epsilon_L \propto \left(\frac{p}{p_{\text{th}}} \right)^{\lceil d/2 \rceil}. \quad (2)$$

However, the code in question does not determine only the threshold value, as it is also significantly influenced by the decoding method [18] and the potentially different error parameters that describe the different operations in the syndrome extraction circuit [19].

Determining the threshold value for a specific noise model can be done by rescaling every noise parameter with the same value and plotting the logical error rate as a function of the physical error rate for different code distances. In experiments, one could, in principle, amplify the error channels to achieve a similar effect, but this would require precise knowledge about each error mechanism that affects the syndrome extraction circuit.

To overcome this issue, we employ the *error suppression factor* Λ as a practical metric to compare the performance of various codes and decoders [20]. For the general case, we define Λ_d as

$$\Lambda_d = \frac{\epsilon_L(d - 2)}{\epsilon_L(d)}, \quad (3)$$

where $\epsilon_L(d)$ is the logical error rate of the distance- d code for $d \geq 3$. For small physical error rates where Eq. (2) is valid, we can also define the distance-independent Λ through the scaling of the logical error rate ϵ_L as

$$\epsilon_L \propto \left(\frac{1}{\Lambda} \right)^{\lceil \frac{d}{2} \rceil}. \quad (4)$$

Therefore, $\Lambda = p_{\text{th}}/p$ effectively measures how far below the threshold the error rate is.

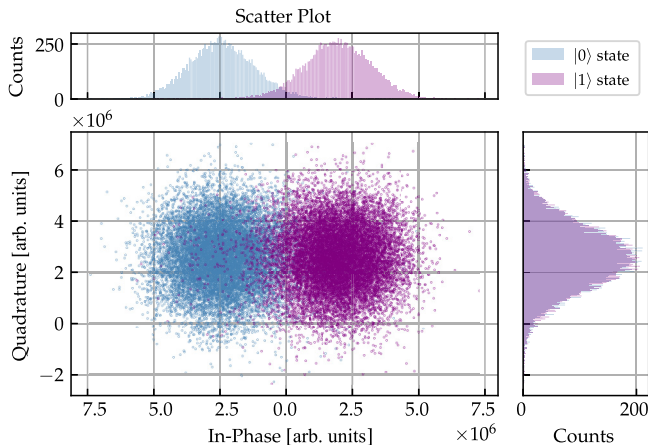


FIG. 1. IQ data for qubit 72 of the IBM Sherbrooke device prepared in the $|0\rangle$ (blue) and $|1\rangle$ (violet) states. The data show two distinct Gaussian distributions corresponding to the qubit’s computational states.

Comparing Eq. (4) with Eq. (2), we see that determining the Λ factor for different decoders allows us to effectively compare the threshold as the ratio of the Λ factors corresponds to the ratio of the thresholds:

$$\frac{p_{\text{th},1}}{p_{\text{th},2}} = \left(\frac{\Lambda_1}{\Lambda_2} \right). \quad (5)$$

Note that $\Lambda = \lim_{d \rightarrow \infty} \Lambda_d$ holds if every noise parameter can be rescaled with a single parameter p . However, in experiments the error rates of different qubits are not uniform. Therefore, increasing the code distance changes the noise model, making the relationship between the measurable Λ_d and the threshold parameter Λ far less obvious. While this complicates the strict interpretation of Λ as an absolute threshold, it remains a robust effective threshold metric for comparing the error suppression capabilities of different decoding strategies. In the following, we use Λ to denote the fitted decay rate of $\epsilon_L(d)$ and discuss the model violations in the experimental data in Sec. IV.

When decoding multiple rounds of syndrome measurements, we obtain the *total logical error rate* P_L rather than the previously discussed logical error rate *per round* ϵ_L . Assuming each round incurs a logical error with probability ϵ_L , the total logical error rate P_L after T rounds is given by [18]

$$P_L = \frac{1}{2}(1 - (1 - 2\epsilon_L)^T). \quad (6)$$

Using $(1 - 2\epsilon_L)^T \approx 1 - 2T\epsilon_L$ for $\epsilon_L \ll 1$, we can approximate the total logical error rate as

$$P_L \approx T\epsilon_L. \quad (7)$$

Consequently, to compute the Λ factor when decoding multiple rounds, we must first derive the logical error per round ϵ_L from the total logical error rate P_L and the number of rounds T . We then use Eq. (4) to determine the Λ factor. Utilizing this methodology, we will calculate the lambda factors to compare the performance of various decoders in the subsequent sections.

C. Soft information

In superconducting hardware, qubit readout is performed by measuring a microwave signal reflected from a resonator coupled to the qubit. The signal is integrated to extract the *in-phase* (I) and *quadrature* (Q) components, which form two distinct Gaussian distributions on the *IQ plane* corresponding to the qubit’s state. Figure 1 shows statistical IQ data obtained from a

superconducting qubit. Typically, the continuous IQ data are discretized into a binary outcome by determining which Gaussian distribution the point most likely belongs to. However, due to the overlap of the distributions, this process can lead to *assignment errors*, commonly called *soft flips*.

We model the measurement process by assuming that the qubit state is projected onto a true state $\bar{z} \in \{0, 1\}$, producing a soft outcome μ drawn from a distribution $f^{\bar{z}}(\mu)$. While leakage to higher-energy states (e.g., $|2\rangle$) is possible, we treat such events separately as discussed in Sec. II F. We estimate the state $\hat{z}$ using a maximum likelihood assignment. A *soft flip* occurs when $\hat{z} \neq \bar{z}$. In the absence of resets, after several stabilizer measurement rounds, the ancilla qubits are expected to be randomly in either the $|0\rangle$ or the $|1\rangle$ state. Therefore, we assume equal prior true-state probabilities $\mathbb{P}[\bar{z} = 0] = \mathbb{P}[\bar{z} = 1] = 0.5$. Under this assumption, the probability of a soft flip given an outcome μ is

$$\mathbb{P}[\text{soft} \mid \mu] = \left[1 + \frac{f^{(\hat{z})}(\mu)}{f^{(\hat{z} \oplus 1)}(\mu)} \right]^{-1}. \quad (8)$$

This device-independent model requires only the distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$, which can be obtained from calibration. This assumption generally holds for stabilizer measurements after sufficient rounds, as the ancilla qubits will likely be randomly in the $|0\rangle$ or $|1\rangle$ state. However, for the initial rounds and particularly for the final logical code readout, this assumption may not hold. Therefore, incorporating additional information derived from Pauli tracing [21] could improve the model's accuracy and, consequently, the efficacy of the methods discussed in this paper.

D. Decoding graph with soft flips

The repetition code is a one-dimensional code with an interleaved arrangement of qubits and stabilizers along a line. This layout forms a two-dimensional decoding graph, where a line of syndrome nodes connected by code qubit edges extends in the time direction. The hard flip channel connects consecutive syndrome nodes. Data qubit error channels connect neighboring nodes. An additional error channel links the diagonal syndromes; this error is caused by an error occurring between cnots during the stabilizer measurement. This channel's orientation depends on the error propagation direction, determined by the CNOT gate ordering. An example of the decoding graph for a repetition code with $d = 3$ is shown in Fig. 2.

When ancilla qubits are not reset, a soft flip (measurement misassignment) results in an incorrect calculation of the syndrome change, effectively creating a pair of syndromes separated by *two* time steps (see Fig. 3). This contrasts with a hard flip (qubit state flip), which creates syndromes separated by one time step. We represent this error channel in the decoding graph by adding a *distance-two time edge* (soft edge) connecting syndrome nodes separated by two time steps. Since soft flips still produce pairs of syndromes, they can be handled by the matching decoder.

For the final round of stabilizers, computed from code qubit measurements, soft flips on ancillas do not propagate to a second time step. Thus, soft and hard flips become indistinguishable, combining to a single measurement error probability $\mathbb{P}_E[\mathbf{E}_m] = p_h(1 - p_s) + (1 - p_h)p_s$.

E. Dynamic weighting using soft information

In this section, we will explore how we can improve decoding by using information contained in the analog nature of the signal.

We improve decoding by *dynamically adjusting the weights* in the decoding graph. Instead of using only the binary outcomes, we use the soft outcomes $\{\mu_t\}$ to compute the specific soft flip probability $\mathbb{P}[\text{soft} \mid \mu]$ for each measurement [Eq. (8)]. This allows us to refine the noise model on a *per-shot basis*, assigning lower weights to more ambiguous measurements. The procedure is outlined in Algorithm 1.

Using the soft outcomes directly is unique as it is the only additional information we can extract immediately from the stabilizer rounds. This extra information can remedy the loss of

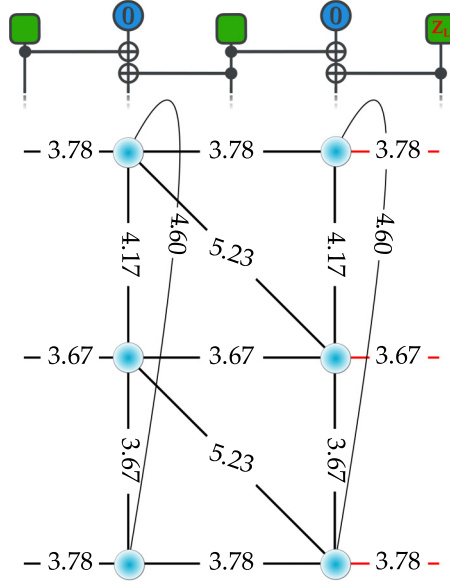


FIG. 2. Decoding graph for a repetition code of distance $d = 3$ with $T = 2$ stabilizer round, including the final code qubit readout. The graph represents the syndrome nodes (blue circles) connected by code qubit and measurement error edges. The weights of the edges are typical for the IBM Sherbrooke noise model. The red edges represent the qubit used to read out the logical state.

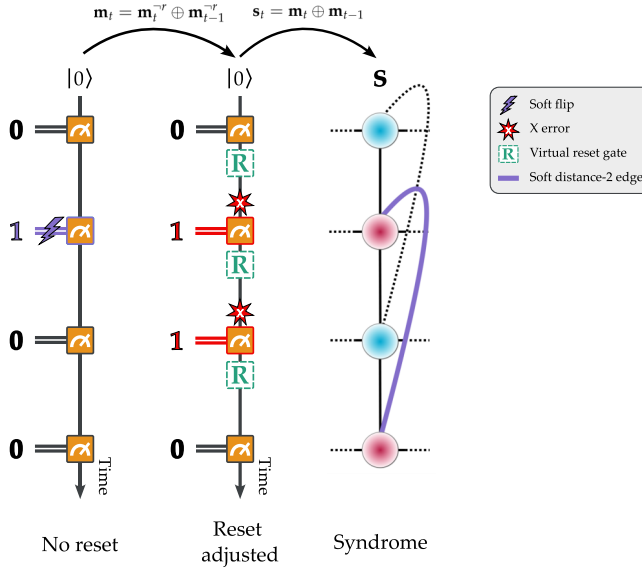


FIG. 3. Timeline of measurement outcomes without resets, the inferred corrected outcomes, and the corresponding single stabilizer column of the decoding graph G_T . A soft flip occurs at time $t = 2$, and the qubit state is wrongfully read out as $|1\rangle$. When calculating the reset adjusted measurement outcomes, the soft flip is equivalent to two consecutive hard flips. A hard flip can be seen as a bit-flip error on the ancilla before measurement. Due to this overcorrection, the column contains a pair of flipped syndromes separated by two time steps.

ALGORITHM 1. Dynamic reweighting of soft edges in the decoding graph.

Input:

- 1: Weighted decoding graph G_T for T rounds of measurements, with stabilizer nodes $\{n_{a,t} \mid a \in \{1, \dots, m\}, t \in \{1, \dots, T\}\}$.
- 2: Soft stabilizer measurement outcomes $\{\mu_t\}_{t=1, \dots, T}$.
- 3: Soft code measurement outcomes $\mu^{c_1}, \dots, \mu^{c_n}$.
- 4: Probability distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$.

Output:

- 5: A reweighted decoding graph $\tilde{G}_T$.
- 6: **for each** $t = 1$ **to** T **do**
- 7: **for each** stabilizer $a = 1$ **to** m **do**
- 8: Retrieve the soft outcome $\mu_{a,t}$ from μ_t .
- 9: Compute the soft flip probability $p_s(\mu_{a,t})$ using Eq. (8).
- 10: **if** $t \leq T - 1$ **then**
- 11: Calculate the edge weight $w = -\ln(p_s/(1 - p_s))$.
- 12: Reweight the distance-two edge $(n_{a,t}, n_{a,t+2})$.
- 13: **end if**
- 14: **if** $t = T$ **then**
- 15: Compute the total measurement error probability using Eq. (C1).
- 16: Calculate the edge weight $w = -\ln(p_{\text{tot}}/(1 - p_{\text{tot}}))$.
- 17: Reweight the distance-one edge $(n_{a,t}, n_{a,t+1})$.
- 18: **end if**
- 19: **end for**
- 20: **end for**
- 21: **for each** code outcome μ^{c_i} **do**
- 22: Identify the error channel involving code qubit c_i .
- 23: Compute the soft flip probability $p_s(\mu^{c_i})$ using Eq. (8).
- 24: Compute the total error probability of that channel using Eq. (C1).
- 25: Calculate the edge weight $w = -\ln(p_{\text{tot}}/(1 - p_{\text{tot}}))$.
- 26: Reweight the edge corresponding to the identified error channel.
- 27: **end for**

information from classification. Moreover, this approach is also largely independent of the noise model, establishing it as a general approach for enhancing the decoding of graphlike structures with soft information.

F. Soft decoding in the presence of leakage

Leakage to higher-energy states creates persistent, correlated errors that degrade decoding performance [14]. Unlike previous soft decoding experiments that relied on postselection for small code distances [12,13], our experiments with up to $T = 100$ rounds require active leakage handling. We address this by identifying leakage in the IQ plane, where it typically appears as a separate distribution between the computational states [12].

Following established techniques for converting leakage into erasures [22], we integrate leakage handling into our soft decoding framework by treating leaked ancilla measurements as maximally ambiguous. When a measurement is identified as leaked, its outcome provides no reliable information about the stabilizer. We therefore assign a soft flip probability of $p_s = 0.5$, effectively setting the weight of the corresponding distance-two time edge to zero [22]. Since the binary assignment of a leaked state is random (0 or 1) rather than a definite error, we do not zero out the distance-one time edge. Crucially, in our experiments, we do not reset qubits upon detecting leakage. Consequently, a leaked qubit remains in a state outside the computational subspace, potentially spreading errors to other qubits via CNOT gates in subsequent rounds. While our current decoding framework does

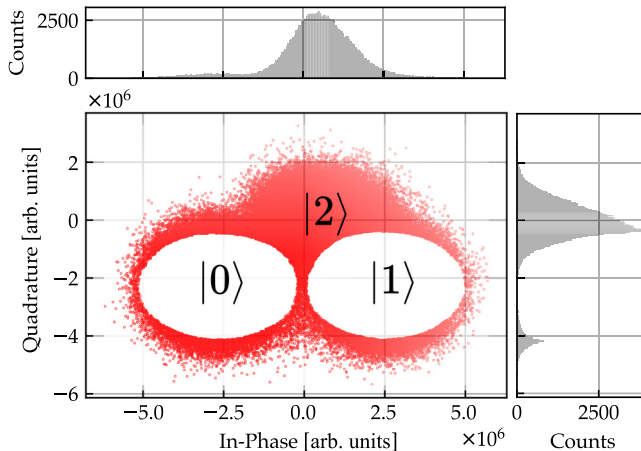


FIG. 4. Visualization of outliers in the IQ plane according to our filtering method for qubit 3 of IBM Sherbrooke. Red points indicate outliers, which are classified as leaked states. They are determined based on their sampling probabilities from the distribution $f^{(z)}(\mu)$, with points falling below the 1% threshold considered as potential leaked states.

not explicitly model these spreading mechanisms, they represent a significant source of correlated errors that could be incorporated into more advanced soft decoding models in future research.

To detect leakage, we repurpose standard calibration data. We classify an IQ point μ as an outlier, and thus potentially leaked, if its sampling probability from the computational state distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$ falls below a 1% threshold. We selected this threshold heuristically by visually inspecting the IQ data for all qubits to achieve a reasonable separation between leaked and computational states. While this approach allows us to identify leakage without dedicated calibration sequences (as visualized in Fig. 4), it is not perfect and requires further validation. Ideally, this limitation could be mitigated by performing a three-state discrimination using dedicated calibration data for the leaked state.

This outlier-based detection has limitations. It relies on a circular decision boundary rather than the ideal linear boundary between leaked and computational states [12], potentially underestimating leakage. Furthermore, a conservative threshold is necessary to avoid misclassifying genuine computational state outliers as leakage. Despite these imperfections and the lack of a robust theoretical basis, this simple heuristic significantly enhances decoding performance, as demonstrated in subsequent sections.

III. RESULTS

A. Simulations

To explore the advantages of soft decoding repetition codes on superconducting hardware, we initially performed simulations that mimic the behavior of realistic quantum hardware to assess the effectiveness of soft decoding. After these preliminary simulations, we conducted experiments on IBM Quantum’s deployed device IBM Sherbrooke, a 127-qubit Eagle device not specifically tuned for quantum error correction tasks like repetition codes. Rather than mitigating leakage and other measurement errors through fine-tuning of the measurement process, as would be done in a full demonstration of an error correcting code, we instead simply use the measurements as provided “out of the box” through Qiskit [23]. This allows us to investigate how much the leakage can be mitigated purely through the decoding. Using this general-purpose approach, we did not aim to maximize QEC performance but to evaluate how soft decoding behaves under practical conditions comparing the outcomes with those of the simulations.

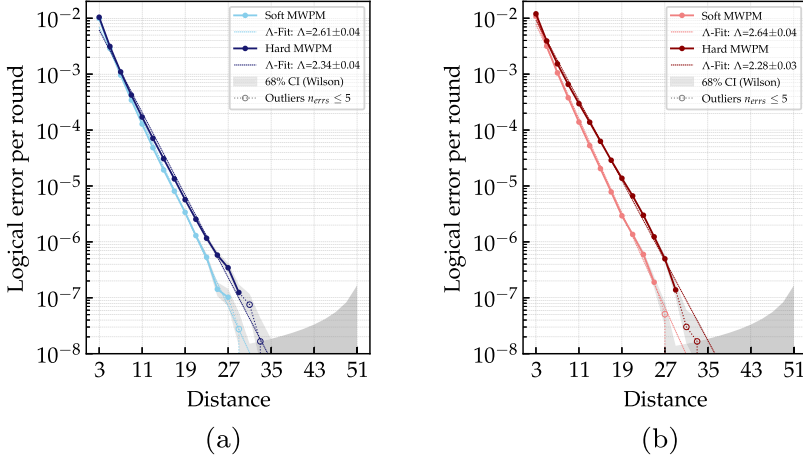


FIG. 5. Simulation results of the IBM Sherbrooke device, depicting the logical errors per round as a function of code distance for the soft and hard MPWM decoders for $T = 50$ rounds of stabilizer measurements. The lambda factors Λ are fitted from the curves' slopes in the logarithmic scale. For $d \geq 35$, no logical errors were observed; the shaded regions indicate the upper bound of the 68% Wilson confidence interval given the finite sample size. (a) Prepared logical state $|\Psi\rangle_L = | +z \rangle_L$ and (b) prepared logical state $|\Psi\rangle_L = | +x \rangle_L$.

We investigate the logical error rate over $T = 50$ rounds of stabilizer measurements, simulating a repetition code of distance $d = 51$. This is the maximal feasible odd distance that we found while ensuring that the used line of qubits did not include any cnot connection with fidelity $< 90\%$ and ancilla with readout fidelity $< 80\%$. Additionally, we derive results for all odd distances ranging from $d = 3$ to $d = 51$, employing the subsampling method utilized in our hardware experiments, as described in Appendix F. This approach allows us to closely mimic the hardware conditions and evaluate the performance of soft decoding across various distances. We compare two decoding strategies: *dynamic soft decoding* and *calibrated hard decoding*, where the latter estimates each error probability from calibration data, as described in Appendix D, and weights the decoding graph once.

While this setup with $T = 50$ does not approach the condition of $T \gg d$, it aligns with the conventional experimental practice where d is set equal to T , ensuring that the “time distance” equals the code distance. Reference [18] showed that time boundary effects in repetition codes do not affect the logical error rate per round for $T \geq 10$, supporting our decision to set $T = 50$.

We construct our noise model using averaged calibration data from the device (see Appendix D). Gate errors are modeled as depolarizing channels based on single- and two-qubit gate fidelities, while idling errors are modeled as Pauli channels with bit-flip and dephasing rates derived from T_1 and T_2 decoherence times. Since Pauli channels act symmetrically on computational basis states, we simulate only one logical state per basis. However, because bit-flip and dephasing rates differ, we present results for both the X and Z bases.

Simulation results for soft decoding of repetition codes are illustrated in Fig. 5 for the Z and X bases. These plots show the *logical error per round* ϵ_L and fit the lambda factor Λ from the curve's slope in logarithmic scale as a function of varying code distances from $d = 3$ to $d = 51$ for $T = 50$ rounds of stabilizer measurements. They compare the soft and hard decoders and include the 68% confidence intervals calculated using the Wilson score interval [24]. The confidence intervals reflect the varying number of shots for different distances, with higher distances having fewer shots due to subsampling, as indicated by the intervals' width. Points with fewer than *five* logical error events appear as dashed lines, suggesting insufficient statistical data to represent the logical error rate.

Importantly, we lack data points for distances $d \geq 35$, as no logical error events were observed in our simulations. On a logarithmic scale, a zero error rate corresponds to minus infinity and is there-

fore omitted. However, we retain the shaded 68% Wilson confidence intervals in this region to reflect statistical uncertainty. Even with zero observed errors, the finite sample size implies a nonzero upper bound on the true error rate. The widening of this region at larger distances results from a reduced number of shots: Since the computational cost of simulating each shot scales with the code distance, we collected fewer samples at higher distances, leading to broader confidence intervals.

Both error rates demonstrate a linear decay with increasing code distance on a logarithmic scale, indicating exponential suppression of errors, as detailed in Sec. II B. The curve for the soft decoder consistently lies below that of the hard decoder across all distances, demonstrating the higher performance of the soft decoder. Notably, the error rates for the maximal distances resolved for the Z and X bases are 3 and 15 times lower, respectively.

Figure 5 demonstrates that the soft decoder’s curve has a steeper slope than that of the hard decoder, indicating a higher threshold and an exponentially superior performance for the soft decoder. As detailed in Sec. II B, the lambda factor quantifies how many times lower the physical error rates are compared to the decoder’s threshold. Consequently, the ratio of the lambda factors between the soft and hard decoders directly corresponds to the ratio of their thresholds. By comparing the two Λ factors, we determine that the threshold of the soft decoder surpasses that of the hard decoder by 11.5% for the Z basis and 15.7% for the X basis, respectively. These improvements exceed the bounds of statistical noise, confirming the significant exponential enhancement provided by soft decoding for repetition codes.

B. Decoding data from quantum hardware

While the simulations offer valuable insights, they do not fully capture complexities such as correlated errors, including qubit crosstalk and leakage, which can affect hardware performance. Although recent research has proposed more accurate metrics for closer-to-hardware simulations using calibrated device error rates [25], we did not pursue this approach in our current study. Therefore, to better understand the practical implications of soft decoding, we must proceed with experimental implementation on the actual hardware.

Figure 6 depicts the logical error per round against code distance for $T = 50$ measurement rounds for all four prepared logical states. It also displays the fitted lambda factor Λ for both hard and soft decoding results for every odd distance from $d = 3$ to $d = 51$, along 68% confidence intervals using the Wilson score interval method. Similar to the simulation results, points representing fewer than five logical error events appear as dashed lines, indicating insufficient data to statistically represent the logical error rate.

The overall trend aligns with our simulations, but several crucial differences emerge. The logical error rates observed are significantly higher than those in the simulations. Comparatively, the observed logical error rates imply that the physical noise is about 1.5 times greater than the calibration data suggests. However, as previously discussed, we attribute this to correlated errors such as crosstalk and leakage. Consequently, drawing a direct connection is challenging since our noise model does not incorporate these correlated errors.

Despite these higher error rates, the data exhibit a clear trend of exponential suppression with increasing code size, indicating that the system is operating below the threshold. We note, however, that the curves are not perfectly linear on a logarithmic scale. This deviation from strict linearity is expected because the assumption of a uniform error parameter p does not hold in our experiment; instead, error rates vary spatially across the device. As we subsample different code distances, the inclusion of specific lower-fidelity qubits or couplers introduces variability in the effective noise strength, causing the observed fluctuations. Consequently, the Λ factor derived here should be interpreted as an *effective* error suppression rate averaged over the device’s inhomogeneities. While this effective Λ does not map to a single physical error rate via Eq. (4) in the strict theoretical sense, the robustness of the global exponential decay allows us to use it as a reliable empirical metric to compare the relative performance of the decoders. Furthermore, the noise profile varies depending on the prepared logical state, primarily because the physical dephasing and bit-flip error

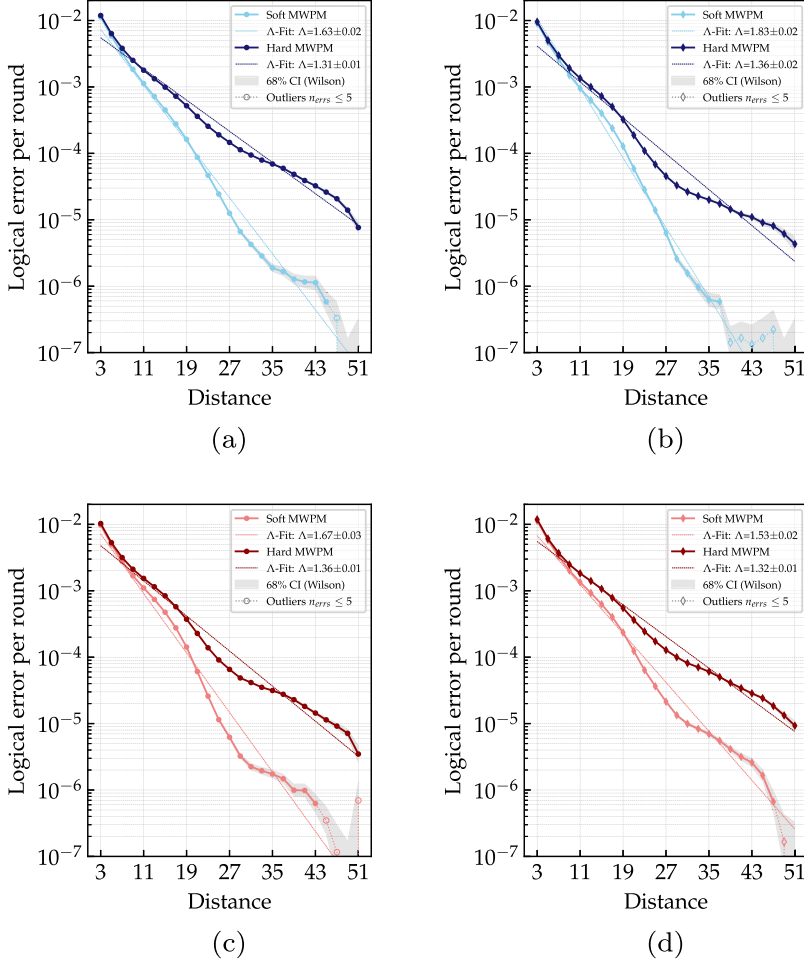


FIG. 6. IBM *Sherbrooke* device data depicting the logical errors per round against code distances for the hard and soft MPWM decoders for $T = 50$ rounds of stabilizer measurements. The lambda factors Λ are fitted from the curves' slopes in logarithmic space. (a) Prepared logical state $|\Psi\rangle_L = |+\rangle_L$, (b) prepared logical state $|\Psi\rangle_L = |-\rangle_L$, (c) prepared logical state $|\Psi\rangle_L = |+\rangle_L$, and (d) prepared logical state $|\Psi\rangle_L = |-\rangle_L$.

rates are not uniform. This discrepancy directly affects the shape of the logical error rate curves. To fully capture this behavior, we prepared all four logical states and present the corresponding plots.

Importantly, the soft decoder consistently outperforms the hard decoder, with this advantage being even more pronounced than in simulations, as will be further addressed in the next section. The soft decoder achieves logical error rates up to 30 times lower than those of the hard decoder, with a magnitude of improvement observed as early as $d = 35$ and $d = 27$ for prepared states in the X and Z bases, respectively.

This advantage stems from the steeper slopes of the soft decoder compared to the hard decoder, with this difference also being more substantial than in simulations. We present a summary of the lambda factors in Table I. By taking the ratio of the two lambda factors, we can determine the ratio of the thresholds of the two decoders. This calculation reveals that the threshold of the soft decoder is 24.4% higher than that of the hard decoder when averaged over the four prepared logical states. This enhancement significantly surpasses statistical noise, mirroring the simulations and demonstrating a clear *exponential improvement* of soft decoding for repetition codes implemented on hardware.

TABLE I. Comparison of Λ factors for various logical states using hard and soft decoding methods across $T = 50$ rounds of stabilizer measurements. The soft decoder shows an average threshold improvement of 24.4%. The Λ factors are determined from the error rate per round depicted in Fig. 6.

Logical state	Hard Λ factor	Soft Λ factor	Increase
$ +x\rangle_L$	1.36 ± 0.01	1.67 ± 0.03	+22.8%
$ +x\rangle_L$	1.32 ± 0.01	1.53 ± 0.02	+15.9%
$ +z\rangle_L$	1.31 ± 0.01	1.63 ± 0.02	+24.4%
$ +z\rangle_L$	1.36 ± 0.02	1.83 ± 0.02	+34.6%

C. Comparison of dynamic and static reweighting

Additionally, to verify that the advantages of soft decoding stem from its dynamic reweighting, we decoded the data using the *data-informed hard decoding* method. Similar to calibrated hard decoding, this method employs a nondynamic approach. Unlike the former, it does not use pre-experiment calibration data for p_s ; instead, it calculates the mean soft flip probability from all soft outcomes collected throughout the experiment. This approach necessitates statistical data collection before decoding, which leads to complications in a real-time error correction scenario. However, it serves a crucial experimental role by verifying that the value added by dynamically weighting effectively addresses the variance in the ambiguity of measurements on a shot-by-shot basis. Doing so confirms that shot-per-shot adjustments to the weights genuinely enhance the decoding process. Figure 7 reveals that the data-informed hard decoder outperforms the calibrated hard decoder yet falls short of the soft decoder's effectiveness. This outcome underscores the critical role that dynamic reweighting plays in enhancing decoding with analog information.

D. Decoding with a single byte of soft information

Soft information decoding offers a promising approach to improve error correction on quantum hardware but introduces two significant challenges for practical implementation. First, it requires transporting a much larger volume of data from the measurement electronics to the decoding hardware compared to hard decoding. Second, it imposes a computational overhead due to the

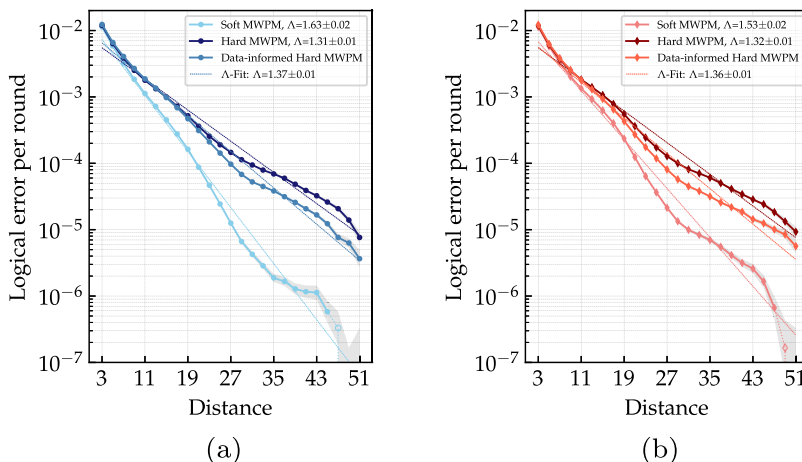


FIG. 7. Comparison of logical errors per round for data-informed hard decoding, dynamic soft decoding, and calibrated hard decoding, based on experimental data from IBM Sherbrooke for $T = 50$ rounds of measurements. (a) Prepared logical state $|\Psi\rangle_L = |+z\rangle_L$ and (b) prepared logical state $|\Psi\rangle_L = |-x\rangle_L$.

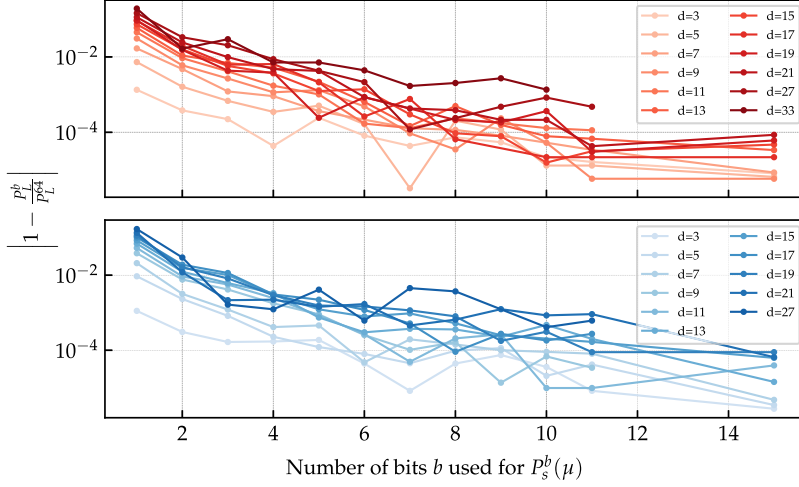


FIG. 8. Simulated relative performance gap $|1 - P_L^b / P_L^{64}|$ for multiple code distances as a function of the number of bits b used to truncate the soft flip probabilities for the states $|+x\rangle$ in red and $|+z\rangle$ in blue. The simulations are performed at twice the physical error rates of IBM *Sherbrooke*, focusing on code distances $d = 33$ for the X basis and $d = 27$ for the Z basis to maintain sufficiently high error rates. Note that for higher code distances and high bit precisions, data points are omitted where the logical error rate P_L^b identically matches P_L^{64} (i.e., the gap is zero), as the logarithm is undefined.

need to dynamically reweight the decoding graph for every shot. For practical quantum error correction, decoding must occur in real time, likely on field-programmable gate arrays (FPGAs). In this scenario, the volume of information plays a dual role: It dictates the bandwidth required for data transfer and directly impacts the computation speed, as higher bit precision increases the complexity of the reweighting logic. Rather than reporting timing measurements from our offline batched and parallelized C++ implementation, which would not reflect the performance of a dedicated FPGA design, we focus on the bit precision as the fundamental metric. These constraints motivate us to explore the trade-off between the volume of information (number of bits) required from measurements and the resulting decoding performance.

To assess the impact of reducing the precision of the soft flip probabilities on soft decoding, we repeat the simulations from Sec. III A with $T = 50$ and incrementally increase the accuracy of the soft flip probability from $b = 1$ to $b = 15$ bits as described in Appendix H.

We calculate the ratio of the logical error rate of the decoder with truncated accuracy to that of the decoder with total accuracy. We then plot the relative performance gap $|1 - P_L^b / P_L^{64}|$ against the number of bits b used for the truncation. A gap close to 0 suggests that the decoder with truncated accuracy performs on par with the full-accuracy decoder. This metric allows us to “normalize” the results and, thus, compare multiple distances within the same plot.

However, this method requires a substantial number of logical error events to stabilize the logical error rate. Consequently, we use the simulations at twice the device’s physical error rates. Additionally, we restrict our analyses to $d = 33$ for the X basis and $d = 27$ for the Z basis to maintain sufficiently high error rates. We set these limits to align with the benefits observed from soft decoding on hardware, demonstrating improvements exceeding an order of magnitude beyond these thresholds, resulting in highly variable ratios in the device data thereafter. The results of these simulations are presented in Fig. 8.

Most notably, the data reveal an approximately linear decrease of the deviation on a logarithmic scale, indicating an exponential convergence to the full-precision performance as the bit precision increases. We observe that $b = 6$ bits are sufficient to reduce the deviation to below 1% of the full-accuracy performance across all distances.

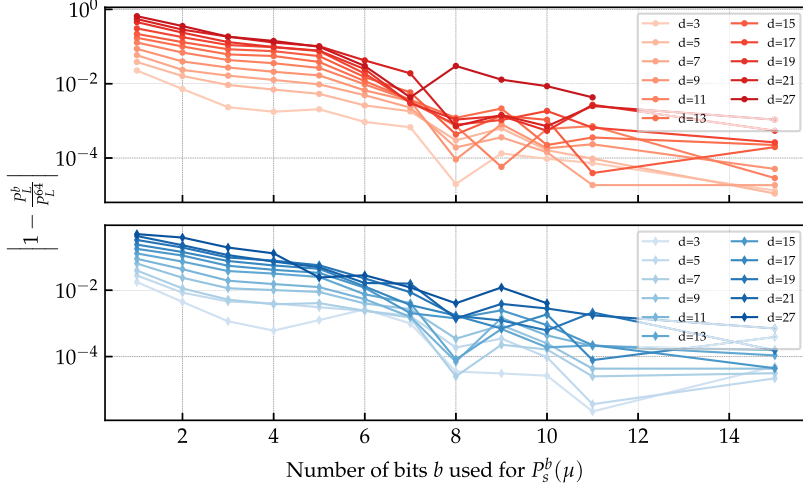


FIG. 9. IBM Sherbrooke device data depicting the relative performance gap $|1 - P_L^b / P_L^{64}|$ for varying code distances as a function of the number of bits b used to truncate the soft flip probabilities. The states $|+x\rangle$ and $|-z\rangle$ are displayed in red and blue, respectively. Distances are limited to $d = 33$ for the X basis and $d = 27$ for the Z basis to maintain a sufficiently stable probability ratio. Points where the gap is zero are omitted.

To validate the simulation results, we implemented the same methodology on hardware, revisiting experimental data from the previous section with $T = 50$ using reduced accuracy soft flip probabilities. The findings, illustrated in Fig. 9, showcase the logical error rate gap for varying code distances and different bit resolutions b . We only show the results for the $|+z\rangle_L$ and $|-x\rangle_L$ states for compactness.

Additionally, we limited our evaluation to code distances up to $d = 27$ for the X and Z bases, maintaining high error rates necessary for stable calculations. We imposed these constraints in line with the observed advantages of soft decoding on hardware in the previous section, where error rates beyond these thresholds were crucially lower, implying significantly variable ratios.

The hardware results mirror the simulation outcomes, with the relative performance gap decreasing exponentially across all distances and states. Again, larger distances demonstrate increased sensitivity to the precision of the soft flip probabilities. However, a crucial difference lies in the convergence point, which occurs later at $b = 8$ bits.

We hypothesize that the earlier convergence in our simulations results from the diminished role of soft flip probabilities in the overall error budget, given that we simulate at twice the noise rates of the hardware. Exploring the convergence behavior across varying noise rates and devices could be an avenue for future research.

Nevertheless, the crucial behavior—the exponential convergence across all distances—remains consistent between the simulations and the hardware results. This consistency is promising, as it pinpoints a particular number of bits that effectively minimizes the information volume required for soft decoding without compromising performance. In our experiments, a 1% gap is achieved at one byte, suggesting an *eightfold reduction* in the data volume needed for soft decoding. This reduction underscores the practicality of soft decoding on hardware, including in time-sensitive contexts such as real-time decoding.

More sophisticated truncation strategies could potentially reduce the amount of information even further. However, the presented truncation method is a simple and effective approach that significantly reduces the data footprint while preserving the advantages of soft decoding. Thus, unless specific hardware constraints demand it, implementing more complex strategies for reducing the amount of analog information required for soft decoding may not be necessary.

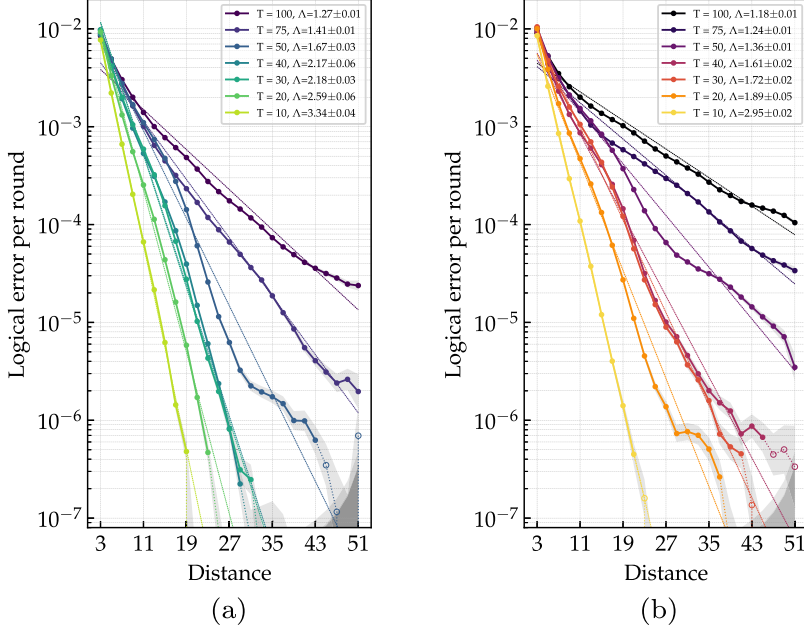


FIG. 10. IBM Sherbrooke device data depicting the logical error per round plotted against distance using both (a) soft MWPM decoder and (b) hard MWPM decoder for varying total round counts T . Results are shown for the $|+\chi\rangle$ state, with consistent behavior across all prepared states.

These conclusions are further strengthened by the known robustness of the MWPM algorithm to small fluctuations in edge weights. Consequently, a slight discretization error in the soft flip probabilities does not severely impact the decoder's ability to identify the most likely error (MLE) path. The consistency in the observed convergence behavior is intriguing and requires further investigation. We suspect a theoretical explanation for this phenomenon, potentially related to the general influence of precision in the weights of decoding graph edges on the performance of MWPM or other graph-based decoders in the presence of noise. This topic presents a promising direction for future research.

E. Effect of leakage on the decoding performance

To demonstrate the effect of leakage in our experiments, we ran repetition codes similar to those presented in Sec. III B but varied the number of rounds from $T = 10$ to $T = 100$. Increasing the number of rounds extends the experiment duration, allowing leakage to accumulate.

We plotted the logical error per round ϵ_L against distance for various total round counts T using both hard and soft decoders in Fig. 10. Additionally, we depicted the fitted lambda factors for each round. For compactness, we only show the results for the $|+\chi\rangle$ state as the behavior is consistent across all prepared states.

The logical error rate per round shows an apparent increase as the number of rounds grows for hard and soft decoders. The growth rate is less pronounced for the soft decoder than for the hard decoder, but it remains considerable. This trend suggests the presence of accumulating errors, such as leakage, crosstalk, or coherent errors, which degrade the performance of both decoding strategies over time. While our soft decoding approach mitigates the impact of leakage by treating identified leaked outcomes as maximally ambiguous, the continued error growth indicates that either residual leakage or other time-dependent error mechanisms remain significant factors. Moreover, the

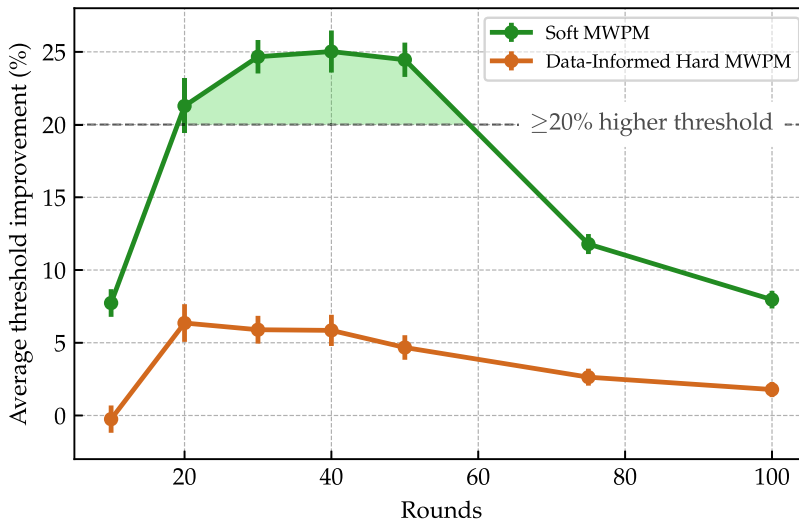


FIG. 11. Average threshold improvement of soft and data-informed decoding over hard decoding as a function of the number of rounds. The results are based on experimental data from IBM Sherbrooke.

observed increase in logical error rate is consistent with the findings from Ref. [20], who noted a decrease in the Λ factor from round to round when implementing repetition codes on hardware.

Observing Fig. 10 and comparing the soft and hard decoders across different numbers of rounds reveals an interesting feature—a variable advantage for soft decoding. To further demonstrate this variable advantage, we calculated the average threshold improvement as a function of rounds in Fig. 11. Here, we see that the benefit of soft decoding peaks in the midrange of rounds and diminishes for both lower and higher rounds.

Figure 11 reveals that the advantage of soft decoding is influenced by the number of rounds and the corresponding amount of leakage. These results indicate that while this leakage management technique does not eliminate the issues associated with leakage, it plays a vital role in the effectiveness of the soft decoding strategy, significantly enhancing its performance. It suggests that our leakage management is, in fact, a primary reason for the substantial improvement of the soft decoder over the hard decoder. This conclusion is supported by the observation that simulations, which lack leakage, show only a 13.6% improvement in the threshold, whereas hardware results demonstrate a 24.4% improvement between soft and hard decoding.

We can attribute the exact pattern observed in Fig. 11 to several causes: With fewer rounds, two contributing factors emerge—Fewer temporal edges are available for targeted adjustments, and the leakage population is lower, diminishing the soft decoder’s advantage in handling leaked states. Conversely, excessive rounds increase leakage, resulting in significant errors affecting the qubits and gates that interact with leaked qubits. This increase in errors dilutes the benefits of managing measurement ambiguities, as they become overshadowed by more prevalent errors elsewhere in the system. Furthermore, given the suboptimal nature of our leakage detection process, the accuracy of the soft decoder diminishes as leakage intensifies, hence further contributing to the diminishing benefits of soft decoding at higher rounds.

Similarly, the data-informed hard decoder shows near-zero benefits for both high and low rounds but maximizes in midrounds. In low rounds, leakage is minimal, so the mean ambiguity of the IQ points remains close to that obtained from calibration data. In midrounds, more leakage occurs, and informing the decoder about the generally higher uncertainty of our measurement points aids in handling this leakage—though not as effectively as individually indicating the ambiguity for each measurement outcome. However, for many rounds, extensive leakage affects surrounding qubits

and gates, thus diminishing the benefits of informing the decoder about the mean ambiguity of measurement edges compared to midrounds.

IV. DISCUSSIONS

Our results show that soft decoding offers a clear advantage over hard decoding. However, a significant portion of the observed advantage on hardware arises from our approach to managing leakage as maximally ambiguous rather than handling the ambiguity of $|0\rangle - |1\rangle$ state discrimination.

Our simulations of the repetition code experiments emphasize the role of leakage in the benefit of soft decoding. In retrospect, the simulations can be seen as a tool to investigate the theoretical leakage-free behavior of soft decoding, as we did not include leakage in our noise model. They suggest that the advantage of using soft information should be at an average threshold increase of about 13%. However, our hardware experiments show that the advantage of soft decoding is significantly greater with a 24.4% higher threshold. This discrepancy underlines the role of treating the leaked states as maximally ambiguous.

The impact of leakage on the soft decoding's advantage is further underlined by the threshold improvements when running experiments with varying numbers of rounds and, thus, leakage populations. We observe that the benefit of soft decoding peaks in the midrange of rounds with more than a 20% higher threshold and decreases for both lower and higher rounds to around a 7% improvement. This pattern results from the reduced leakage with a lower number of rounds, diminishing the soft decoder's advantage in handling leaked states. Conversely, with a higher number of rounds, increased leakage leads to widespread errors throughout the system, undermining the advantage of managing measurement ambiguities and thus lessening the soft decoder's benefit.

The importance of our proposed leakage treatment for the benefit of soft decoding raises the question of whether a more refined approach to leakage handling could yield even more significant benefits.

Nevertheless, we believe that our technique is a straightforward and efficient way to improve the performance of soft decoding in scenarios involving leakage. However, it is not a universal remedy, particularly in cases of severe leakage. Therefore, addressing significant leakage through additional mitigation strategies is advisable. Once large-scale leakage is controlled, soft decoding can be effectively applied to manage any residual leakage. This strategy ensures that our adapted soft decoding is employed where it is most advantageous, enhancing its benefits compared to hard decoding.

V. CONCLUSIONS

Our hardware experiments demonstrate a substantial benefit of soft decoding, resulting in up to 30 times lower logical error rates due to a 24.4% higher threshold of the soft decoder compared to the hard decoder. This advantage primarily stems from dynamically reweighting the decoding graph based on the analog measurement outcomes since static decoding with data-informed weights does not provide comparable benefits.

While soft decoding requires handling more data than hard decoding, our findings suggest that we can reduce the precision of the soft flip probabilities to just one byte without compromising the decoder's performance. This data-efficient decoder facilitates the application of soft information decoding in real-time decoding scenarios. If needed by hardware requirements, we can further reduce the precision with tailored reduction schemes.

Finally, we observe that the benefit of soft over hard decoding in our device data is largely due to the presence of leakage and our method of treating the detected leakage as maximally ambiguous measurement outcomes. This method is particularly effective in scenarios with moderate leakage, specifically for up to approximately $T \approx 50$ rounds of stabilizer measurements. In this regime, our leakage handling technique could complement hardware-based leakage reduction protocols, reducing the frequency with which leakage reduction is applied. Such a combined approach would

decrease the error budget stemming from leakage reduction while maintaining appropriate error suppression.

ACKNOWLEDGMENTS

The authors thank I. Hesner, M. Beverland, E. Chen, A. Cross, E. Pritchett, M. Carroll, and M. Li for their valuable discussions and insights. This research was sponsored by the Army Research Office and was accomplished under Grant No. W911NF-21-1-0002. B.H. acknowledges support from the NCCR SPIN, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (Grant No. 51NF40-180604).

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

DATA AVAILABILITY

The code used for the experiments and analysis in this paper is available in Ref. [26]. The data supporting the findings of this study will be made available upon reasonable request.

APPENDIX A: SOFT INFORMATION IN OTHER QUANTUM COMPUTING PLATFORMS

Other quantum computing technologies face similar challenges in achieving accurate qubit state discrimination. For instance, the state measurement in trapped ions involves detecting fluorescence from ions when illuminated with a laser. Ideally, ions in the excited state fluoresce, while those in the ground state do not. However, due to background photon count, laser intensity fluctuations, and photodetector efficiency, the resulting signal also forms a distribution of counts. The overlap between distributions for *dark state* (ground state) and *bright state* (excited state) similarly leads to a finite misassignment probability [27].

In semiconductor quantum dot systems, qubit state readout is often performed through charge sensing, where the tunneling of electrons to and from the quantum dot affects the conductance of a nearby sensor. The conductance signal, influenced by the qubit's charge state, generates distributions that must be distinguished to infer the qubit state accurately. As with superconducting and ion trap platforms, variations in tunnel rates, sensor sensitivity, and environmental noise contribute to the overlap between these distributions, posing a challenge for reliable qubit state discrimination [28].

These examples underscore a common theme across quantum computing platforms: the necessity of managing and interpreting probabilistic signals derived from physical phenomena unique to each technology. The goal remains to optimize signal-to-noise ratios and separation between state-indicative distributions, minimizing the likelihood of assignment errors.

APPENDIX B: SOFT FLIP PROBABILITY FORMULA

We will use the following model to describe these assignments in a device-independent manner. We assume that during the measurement the qubit state ρ gets projected onto one of the *true states* $|\bar{z}\rangle \in \{|0\rangle, |1\rangle\}$ with probability $\mathbb{P}[\bar{z}] = \text{Tr}[\rho |\bar{z}\rangle \langle \bar{z}|]$. We then observe a *soft outcome* μ according to the probability distribution $f^{\bar{z}}(\mu) = \mathbb{P}[\mu | \bar{z}]$, where $\mathbb{P}[\mu | \bar{z}]$ is the conditional probability of measuring μ provided that the state was projected onto $\bar{z}$. To classify the qubit state and obtain the *estimated state* $\hat{z}$, we compare the probabilities of observing the soft outcome for each state and use a *maximum likelihood assignment*:

$$\hat{z} = \begin{cases} 0, & \text{if } f^{(0)}(\mu) \geq f^{(1)}(\mu), \\ 1, & \text{otherwise.} \end{cases} \quad (\text{B1})$$

We summarize the estimated stabilizer outcomes at round t in an *outcome vector* $\hat{\mathbf{z}}_t$, which corresponds to either $\mathbf{m}_t$ or $\mathbf{m}_t^{-r}$, depending on whether we reset the ancilla qubits or not.

With this model, a *soft flip* occurs if the estimated state is not equal to the true state, $\hat{z} \neq \bar{z}$, meaning that we wrongfully estimated the state of the qubit from the soft outcome μ . On the other hand, we call a *hard flip* an error that changes the true state $\bar{z}$ of the qubit before or during the measurement.

The probability of a *soft flip* given an observed soft outcome μ can be written as

$$\mathbb{P}[\text{soft} \mid \mu] = \frac{\mathbb{P}[\text{soft}, \mu]}{\mathbb{P}[\mu]}, \quad (\text{B2})$$

where

$$\mathbb{P}[\text{soft}, \mu] = \begin{cases} f^{(1)}(\mu) \times \mathbb{P}[\bar{z} = 1], & \text{if } f^{(0)}(\mu) \geq f^{(1)}(\mu), \\ f^{(0)}(\mu) \times \mathbb{P}[\bar{z} = 0], & \text{otherwise.} \end{cases} \quad (\text{B3})$$

Note that using the maximum likelihood assignment in Eq. (B1), this can be rewritten as $\mathbb{P}[\text{soft}, \mu] = \mathbb{P}[\bar{z} = \hat{z} \oplus 1] f^{(\hat{z} \oplus 1)}(\mu)$. Furthermore, the probability of observing the soft outcome μ is

$$\mathbb{P}[\mu] = f^{(0)}(\mu) \times \mathbb{P}[\bar{z} = 0] + f^{(1)}(\mu) \cdot \mathbb{P}[\bar{z} = 1]. \quad (\text{B4})$$

Consequently, if we observe a soft outcome μ and estimate the state to be $\hat{z}$, the probability of a soft flip is

$$\mathbb{P}[\text{soft} \mid \mu] = \left[1 + \frac{\mathbb{P}[\bar{z} = \hat{z}]}{\mathbb{P}[\bar{z} = \hat{z} \oplus 1]} \frac{f^{(\hat{z})}(\mu)}{f^{(\hat{z} \oplus 1)}(\mu)} \right]^{-1}, \quad (\text{B5})$$

where $\oplus$ denotes the addition modulo 2. This model allows us to describe the soft flip device independently, only requiring the distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$ to be known. In our calculations, the prior true-state probability $\mathbb{P}[\bar{z}]$ is assumed to be 0.5. Since the ancilla qubits are not reset after the measurements, we expect each ancilla to be in the $|0\rangle$ or the $|1\rangle$ state with 50% probability after several stabilizer measurement rounds.

APPENDIX C: DYNAMIC WEIGHTING ALGORITHM

The following algorithm dynamically weights the decoding graph based on the soft outcomes of stabilizer measurements. It calculates the final error probability of code qubits with

$$\mathbb{P}_E[\mathbf{E}_c^{\text{tot}}] = p_e(1 - p_s)(1 - p_h) + (1 - p_e)p_s(1 - p_h) + (1 - p_e)(1 - p_s)p_h + p_ep_sp_h, \quad (\text{C1})$$

where p_s (p_h) are soft (hard) flip probabilities and $p_e = \mathbb{P}_E[\mathbf{E}_c]$ is the probability of an error $\mathbf{E}_c$ happening on a code qubit c due to gate, idling, or propagation errors.

APPENDIX D: NOISE MODEL ESTIMATION

We estimate our noise model using per-qubit calibration data from IBM Quantum devices, which include single- and two-qubit gate fidelities and the T_1 and T_2 decoherence times. These times are used in

$$p_X = p_Y = \frac{1 - e^{-\tau/T_1}}{4}, \quad (\text{D1})$$

$$p_Z = \frac{1 - e^{-\tau/T_2}}{2} - \frac{1 - e^{-\tau/T_1}}{4}, \quad (\text{D2})$$

to estimate the idling error probabilities. Specifically, we set the idling time τ in each round to correspond to the average measurement time of the ancilla qubits, as the code qubits are only idle while the ancilla qubits get read out. These calibrations occur roughly once per day.

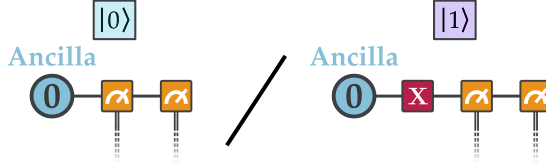


FIG. 12. Double measurement calibration circuit for estimating the probabilities of soft and hard flips in the first measurement.

We run a set of calibration circuits before each experiment to estimate the probabilities for the measurement-related processes. These circuits consist of preparing all the device qubits in the $|0\rangle$ or $|1\rangle$ state and measuring them twice in the Z basis, as depicted in Fig. 12. The second measurement allows us to differentiate between soft and hard flips by comparing the two measurement outcomes. For example, preparing the $|1\rangle$ state, we summarize the possible estimated outcomes $\hat{z}$ and contributing error channels in Table II. We assume that the reset error rate and the single X gate error rate are negligible compared to the measurement error rates.

Ignoring higher-order errors $\mathcal{O}(p^2)$, a *soft flip* occurs on the first measurement when the outcome is 0 and the second outcome is 1 again. We can interpret these outcomes as having had a misalignment on the first measurement, but the state was unchanged, leading to a correct outcome on the second measurement. On the other hand, a *hard flip* occurs when the first outcome is 0 and the second outcome is 0 again. This outcome means that the qubit decayed to the $|0\rangle$ state during the first measurement and was then measured correctly.

Using the calibration data, we can estimate the probabilities of soft and hard flips in the first measurement for the $|1\rangle$ state for each qubit as

$$p_s = \frac{N_{01}}{N_{\text{tot}}}, \quad p_h = \frac{N_{00}}{N_{\text{tot}}}, \quad (\text{D3})$$

with N_{ij} the number of times the outcomes ij were observed and $N_{\text{tot}} = N_{01} + N_{00} + N_{10} + N_{11}$ the total number of measurements. Accordingly, the probabilities for the $|0\rangle$ state are

$$p_s = \frac{N_{10}}{N_{\text{tot}}}, \quad p_h = \frac{N_{11}}{N_{\text{tot}}}. \quad (\text{D4})$$

These probabilities represent the *mean probabilities* of soft and hard flips and are used for the *calibrated hard decoding* discussed in Sec. III B.

Our experiments will average the probabilities p_s and p_h and the gate and idling errors over all the qubits involved. By averaging, we can mitigate inaccuracies due to potential noise drift.

APPENDIX E: KERNEL DENSITY ESTIMATION OF $f^{(\bar{z})}(\mu)$

To estimate the distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$, we run the calibration circuits, retrieve the soft outcomes μ , and then fit the distributions of these points using *kernel density estimation*. Kernel

TABLE II. Summary of the possible outcomes and contributing error channels when preparing a $|1\rangle$ state and measuring it twice. $\emptyset^m$, E_h^m , and E_s^m denote the absence of an error, a hard flip, and a soft flip on measurement m , respectively.

Outcomes	Smallest order	Second smallest order
11	$\emptyset^1 \emptyset^2 = \mathcal{O}(p^0)$	$E_s^1 E_s^2 = \mathcal{O}(p^2)$
00	$E_h^1 \emptyset^2 = \mathcal{O}(p^1)$	$E_s^1 E_h^2 + E_h^1 E_s^2 = \mathcal{O}(p^2)$
10	$\emptyset^1 E_h^2 + \emptyset^1 E_s^2 = \mathcal{O}(p^1)$	$E_h^1 E_s^2 + E_h^1 E_h^2 = \mathcal{O}(p^2)$
01	$E_s^1 \emptyset^2 = \mathcal{O}(p^1)$	$E_h^1 E_h^2 + E_h^1 E_s^2 = \mathcal{O}(p^2)$

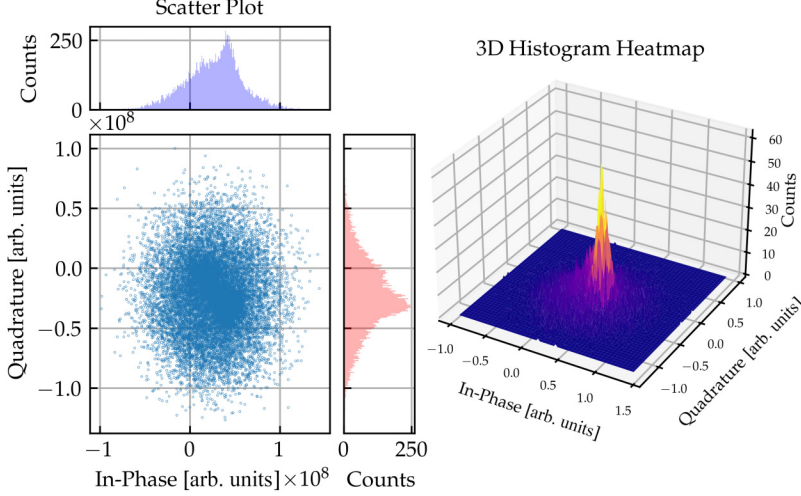


FIG. 13. IQ data for a particularly bad-mannered qubit (106) of the IBM Sherbrooke device prepared in the $|1\rangle$ state. The data show a complex behavior that is not easily modeled by a simple Gaussian distribution.

density estimation is a nonparametric method for estimating the probability density function of a random variable. We prefer this method over parametric methods like a *Gaussian mixture model* as it does not require assumptions about the underlying distribution. Moreover, as seen in Fig. 13, the distributions of the soft outcomes comprise complex shapes that could lead to inaccuracies when using Gaussian mixture models.

Kernel density estimation (KDE) is based on the convolution of a kernel function with the data points. We use the *Epanechnikov kernel* function as it has a few advantages. Statistically, it is the optimal kernel because it provides the best trade-off between bias and variance. Moreover, it has a compact support, meaning that its value is zero beyond a specific range ($|u| > 1$). This property reduces the computational complexity compared to other kernels like the Gaussian kernel since it only accounts for points within its support [29]. The only free parameter is the kernel bandwidth, which determines the smoothness of the estimated distribution, with a smaller bandwidth leading to a more jagged distribution and a larger bandwidth leading to a smoother distribution. We set the bandwidth for each qubit using *cross-validation*, splitting the data into a training and a validation set, and choosing the bandwidth that minimizes the validation error.

To estimate the distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$, we run calibration circuits as shown in Fig. 12. We use the second measurement outcome to filter out data points where a hard flip occurred, as described in Table II. This step ensures that we fit the distributions of the qubit’s “true” initial state rather than its state after the measurement.

However, a complication arises due to the state changing during the readout circuit, meaning the outcome μ would not correspond to the expected prepared state. To circumvent this issue, we use the second measurement of our calibration circuits depicted in Fig. 12 to *filter out* these hard flips. Consequently, we run calibrations for the $\bar{z} = |0\rangle$ and $\bar{z} = |1\rangle$ states and remove every shot where the second measurement outcome disagrees with the prepared state. We then retrieve all soft outcomes from the corresponding state $\mu^{\bar{z}}$ and fit the distribution $f^{(\bar{z})}(\mu)$ to these points using kernel density estimation.

APPENDIX F: EXPERIMENTAL SETUP

We implement various configurations of the repetition code, encoding in both the X and Z bases and varying the prepared logical state between $|+z\rangle_L$ and $| -z\rangle_L$. We encode the logical state,

full dataset	subsampling 1	subsampling 2	subsampling 3
Input 000000000	Input 00000	Input 00000	Input 00000
1 0 0 0 0	1 0 0	1 0 0	1 0 0
2 1 0 0 1	2 1 0	2 0 0	2 0 1
3 0 1 0 0	3 0 1	3 1 0	3 0 0
4 1 0 0 0	4 1 0	4 0 0	4 0 0
5 0 0 0 1	5 0 0	5 0 0	5 0 1
Out 0 1 0 1 0	Out 0 1 0	Out 1 0 1	Out 0 1 0

1 └───┘
2 └───┘
3 └───┘

FIG. 14. Example of subsampling for a repetition code of distance $d_s = 3$ from a $d = 5$ code. The subsampling method allows us to evaluate the performance of repetition codes across multiple distances without the need to individually execute each configuration on the hardware. Figure taken from Ref. [18].

measure T rounds of stabilizer measurements, and conclude with a final readout of all code qubits in the same basis as initially encoded. We inform our noise model with calibration data, and we utilize Pauli tracing via the Stim software to weigh the edges of the decoding graph [21].

We decode the repetition codes using the MWPM algorithm via the PyMatching software [16]. This algorithm determines the MLE by identifying the minimum-weight perfect matching within the decoding graph. To evaluate the logical error rate, we select the last qubit in the chain—though this selection is arbitrary—to carry the logical information. If the MLE affects this logical qubit, we classify it as a logical error. The results from this process allow us to compare the predicted logical error against the actual logical error observed during the final measurement of the code qubit.

We use a technique known as *subsampling* [18,20,30] to run repetition codes sample efficiently on hardware. This approach involves deploying the largest possible repetition code that the hardware can accommodate and decoding various subsets to gather data of smaller distances. This strategy generates $n = d - d_s + 1$ distinct datasets, each corresponding to a repetition code of distance d_s . An example of subsampling for a repetition code of distance $d_s = 3$ from a $d = 5$ code is shown in Fig. 14. By subsampling, we can analyze the performance of repetition codes across multiple distances without the need to individually execute each configuration on the hardware. This method also ensures consistent noise conditions across different experiments, eliminating noise drifts.

Since repetition codes are linear, they are inherently more susceptible to the impact of individual defective qubits than two-dimensional codes. A malfunctioning ancilla qubit, for instance, could disrupt the integrity of the code, effectively breaking the chain and potentially halving the code’s distance. Subsampling smaller distances and averaging over them allows us to systematically evaluate every part of the qubit chain, ensuring that even smaller distances encounter any problematic qubits. However, reducing the distance has an exponential impact on the logical error rate; thus, averaging over subsets does not fully counteract the detrimental effects of bad qubits. Furthermore, as we conclude each experiment with a final code readout, we cannot subsample the number of rounds as simply as we can for the distance. Consequently, we will run different numbers of rounds and use these variations to estimate the logical error rate per round individually.

APPENDIX G: SIMULATION SETUP

To simulate soft decoding of repetition codes as if we run it on the hardware, we first generate outcomes for stabilizer and final code measurements using the circuit-level Pauli noise model. We configure the circuit-level noise model with the calibration data obtained from the hardware for

two-qubit gates, single-qubit gates, and decoherence times. We use these values alongside the explicit code circuit to estimate the noise model. The Stim software [21] facilitates the outcome sampling process by providing tools for Pauli tracing, which we also use to weight the graph.

To then simulate the measurement process and get the soft outcomes μ , we use measurement calibrations run on the hardware to estimate and model the density distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$ using kernel density estimation. For each outcome z , we sample an IQ point according to the fitted distribution $f^{(z)}(\mu)$ and assign this IQ point to the measurement outcome. We use calibration data from the IBM Sherbrooke device for both the noise model and the IQ distributions. We replicate this procedure for all measurement outcomes to assemble the IQ points for the stabilizer and final code measurements. To ensure that the characteristics of the IQ distributions are the only source of soft flips, we set the probability of a soft flip in our noise model used for the generation of hard outcomes to $p_s = 0$.

With these IQ points μ , we reestimate the measurement outcomes $\hat{z}(\mu)$ based on Eq. (B1) and calculate the corresponding soft flip probabilities $p_s(\mu)$ according to Eq. (8). This reestimation incorporates soft flips as they would occur on the device. We use the soft flip probabilities to adjust the weights of the decoding graph according to Algorithm 1. Subsequently, we decode the outcomes and the resulting syndromes using PyMatching.

The software that implements the estimation of outcomes and soft flip probabilities, as well as the software for reweighting the decoding graph, will be made available soon. To ensure rapid execution times, we wrote the performance-critical components in C++. We achieve the kernel density estimation using the MLPack library [31].

Importantly, as discussed in Sec. IIF, the calibration data do not reflect leakage. Consequently, the simulation result fails to capture the full extent of the measurement process that would occur for a more significant number of rounds. However, the simulation results provide valuable insight into the potential benefits of soft decoding repetition codes on hardware.

APPENDIX H: HOW TO REDUCE THE AMOUNT OF MEASUREMENT INFORMATION

Several possibilities exist to reduce the data footprint in the soft decoding process. However, we expect that reducing the amount of analog information leads to similar performance reductions across different methods. Consequently, we focus on a simple, hardware-inspired approach to minimize data usage. This approach involves estimating the outcome $\hat{z}(\mu)$ and the probability of a misassignment $p_s(\mu)$ from the IQ point μ directly on the control card. After estimations, the card truncates $p_s(\mu)$ to a specified precision, reducing the number of bits required for its representation. Following this, the control card transmits both the outcome and the truncated probability of a soft flip to the random access memory (RAM) for subsequent decoding. This method effectively reduces the data retrieved from the measurements by processing them at the control card level before only passing the outcome and the truncated probability of a soft flip to the RAM.

To facilitate fast probability estimation, $f^{(0)}(\cdot)$ and $f^{(1)}(\cdot)$ are pre-estimated during device calibration and stored in a lookup grid, allowing for the computation of the needed $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$ for each IQ point μ in $\mathcal{O}(1)$ time. Additionally, with multiple control cards, each managing a set of qubits, this estimation process can be parallelized, significantly reducing the computational load on the decoder unit by distributing the workload across multiple control cards.

To assess the impact of accuracy loss due to the truncation of p_s , we analyze the decoder's performance using the full accuracy soft flip probability, represented as a 64-bit double (p_s^{64}), and compare it to the performance using the truncated soft flip probability (p_s^{trunc}). We achieve this truncation by rounding the double to b bits. An illustration of the effects of truncating $p_s(\mu)$ is presented in Figs. 15 and 16.

Truncating the soft flip probability p_s uniformly across its entire range is straightforward but may not be the most efficient. It is possible to consider adaptive methods that provide higher precision for ranges where soft flip probabilities are more commonly observed and lower precision elsewhere. As illustrated in Fig. 16, a significant number of points cluster within a specific interval, suggesting

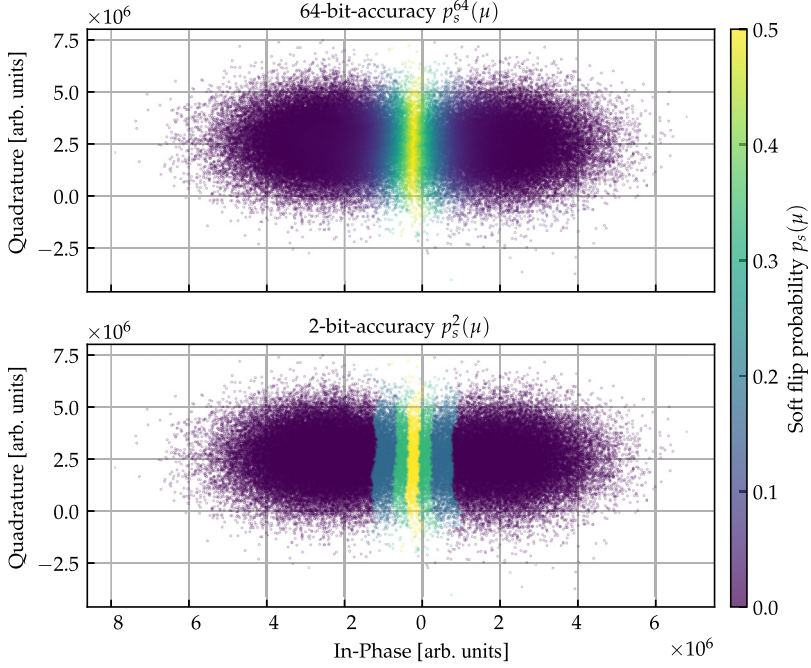


FIG. 15. Colormap representation of measurement outcomes μ for qubit 72 of IBM **Sherbrooke**. The top row shows the full accuracy soft flip probability p_s^{64} , while the bottom row illustrates the effects of truncating the soft flip probability to 2 bits.

that increased accuracy in these areas could enhance efficiency. Additionally, identifying where the decoder is most sensitive to the precision of soft flip probabilities and prioritizing these ranges could further optimize performance. However, as discussed in Sec. III D, this primary truncation method

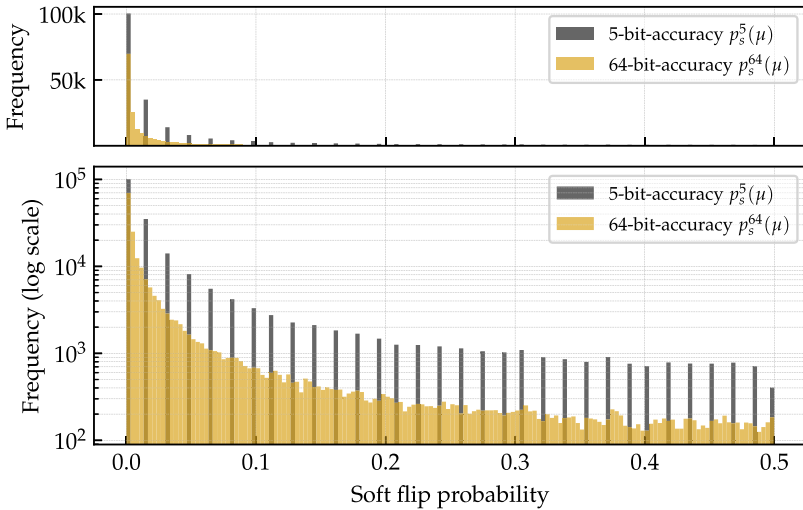


FIG. 16. Histogram of soft flip probabilities for qubit 72 in IBM **Sherbrooke**. The plot illustrates the distribution of soft flip probabilities and the effects of truncating the probabilities to 5 bits. Plotting the frequency without a logarithmic scale reveals that a significant portion of points cluster within 0% and 5%.

significantly reduces data needs while preserving the advantages of soft decoding. Therefore, unless specific hardware constraints demand it, implementing more complex strategies for reducing the amount of analog information required for soft decoding may not be necessary.

-
- [1] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff, Realizing repeated quantum error correction in a distance-three surface code, *Nature (London)* **605**, 669 (2022).
 - [2] Y. Zhao *et al.*, Realization of an error-correcting surface code with superconducting qubits, *Phys. Rev. Lett.* **129**, 030501 (2022).
 - [3] Google Quantum AI, Suppressing quantum errors by scaling a surface code logical qubit, *Nature (London)* **614**, 676 (2023).
 - [4] D. Bluvstein *et al.*, Logical quantum processor based on reconfigurable atom arrays, *Nature (London)* **626**, 58 (2024).
 - [5] M. P. da Silva *et al.*, Demonstration of logical qubits and repeated error correction with better-than-physical error rates, [arXiv:2404.02280](https://arxiv.org/abs/2404.02280).
 - [6] R. S. Gupta, N. Sundaresan, T. Alexander, C. J. Wood, S. T. Merkel, M. B. Healy, M. Hillenbrand, T. Jochym-O'Connor, J. R. Wootton, T. J. Yoder, A. W. Cross, M. Takita, and B. J. Brown, Encoding a magic state with beyond break-even fidelity, *Nature (London)* **625**, 259 (2024).
 - [7] R. Acharya *et al.*, Quantum error correction below the surface code threshold, *Nature (London)* **638**, 920 (2025).
 - [8] C. A. Pattison, M. E. Beverland, M. P. da Silva, and N. Delfosse, Improved quantum error correction using soft information, [arXiv:2107.13589](https://arxiv.org/abs/2107.13589).
 - [9] B. M. Varbanov, M. Serra-Peralta, D. Byfield, and B. M. Terhal, Neural network decoder for near-term surface-code experiments, *Phys. Rev. Res.* **7**, 013029 (2025).
 - [10] J. Bausch, A. W. Senior, F. J. H. Heras, T. Edlich, A. Davies, M. Newman, C. Jones, K. Satzinger, M. Y. Niu, S. Blackwell, G. Holland, D. Kafri, J. Atalaya, C. Gidney, D. Hassabis, S. Boixo, H. Neven, and P. Kohli, Learning to decode the surface code with a recurrent, transformer-based neural network, *Nature (London)* **635**, 834 (2024).
 - [11] X. Xue, B. D'Anjou, T. F. Watson, D. R. Ward, D. E. Savage, M. G. Lagally, M. Friesen, S. N. Coppersmith, M. A. Eriksson, W. A. Coish, and L. M. K. Vandersypen, Repetitive quantum nondemolition measurement and soft decoding of a silicon spin qubit, *Phys. Rev. X* **10**, 021006 (2020).
 - [12] N. Sundaresan, T. J. Yoder, Y. Kim, M. Li, E. H. Chen, G. Harper, T. Thorbeck, A. W. Cross, A. D. Córcoles, and M. Takita, Demonstrating multi-round subsystem quantum error correction using matching and maximum likelihood decoders, *Nat. Commun.* **14**, 2852 (2023).
 - [13] H. Ali, J. Marques, O. Crawford, J. Majaniemi, M. Serra-Peralta, D. Byfield, B. Varbanov, B. M. Terhal, L. DiCarlo, and E. T. Campbell, Reducing the error rate of a superconducting logical qubit using analog readout information, *Phys. Rev. Appl.* **22**, 044031 (2024).
 - [14] K. C. Miao *et al.*, Overcoming leakage in scalable quantum error correction, *Nat. Phys.* **19**, 1780 (2023).
 - [15] G. P. Gehér, M. Jastrzebski, E. T. Campbell, and O. Crawford, To reset, or not to reset—that is the question, *npj Quantum Inf.* **11**, 39 (2025).
 - [16] O. Higgott, PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching, [arXiv:2105.13082](https://arxiv.org/abs/2105.13082).
 - [17] O. Higgott and C. Gidney, Sparse blossom: Correcting a million errors per core second with minimum-weight matching, *Quantum* **9**, 1600 (2025).
 - [18] Google Quantum AI, Exponential suppression of bit or phase errors with cyclic error correction, *Nature (London)* **595**, 383 (2021).
 - [19] B. Hetényi and J. R. Wootton, Tailoring quantum error correction to spin qubits, *Phys. Rev. A* **109**, 032433 (2024).

- [20] J. Kelly *et al.*, State preservation by repetitive error detection in a superconducting quantum circuit, [Nature \(London\) **519**, 66 \(2015\)](#).
- [21] C. Gidney, Stim: A fast stabilizer circuit simulator, [Quantum **5**, 497 \(2021\)](#).
- [22] M. Kang, W. C. Campbell, and K. R. Brown, Quantum error correction with metastable states of trapped ions using erasure conversion, [PRX Quantum **4**, 020358 \(2023\)](#).
- [23] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, Quantum computing with Qiskit, [arXiv:2405.08810](#).
- [24] E. B. Wilson, Probable inference, the law of succession, and statistical inference, [J. Am. Stat. Assoc. **22**, 209 \(1927\)](#).
- [25] I. Hesner, B. Hetényi, and J. R. Wootton, Using detector likelihood for benchmarking quantum error correction, [Phys. Rev. A **111**, 052452 \(2025\)](#).
- [26] https://github.com/quantumjim/soft_information_decoding.
- [27] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, Trapped-ion quantum computing: Progress and challenges, [Appl. Phys. Rev. **6**, 021314 \(2019\)](#).
- [28] G. Burkard, T. D. Ladd, J. M. Nichol, A. Pan, and J. R. Petta, Semiconductor spin qubits, [Rev. Mod. Phys. **95**, 025003 \(2023\)](#).
- [29] V. A. Epanechnikov, Non-parametric estimation of a multivariate probability density, [Theory Probab. Appl. **14**, 153 \(1969\)](#).
- [30] Y. Naveh, E. Kashefi, J. R. Wootton, and K. Bertels, Theoretical and practical aspects of verification of quantum computers, in *Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (IEEE, Dresden, Germany, 2018), pp. 721–730.
- [31] R. R. Curtin *et al.*, mlpack 4: A fast, header-only C++ machine learning library, [J. Open Source Softw. **8**, 5026 \(2023\)](#).