ETH zürich

IBM Research

Master Thesis

Soft Information Quantum Error Correction

Quantum Computational Science Group
IBM Quantum, IBM Research Europe - Zürich

Department of Physics (D-PHYS)
ETH Zürich

Maurice D. Hanisch

April 2024

Supervision
Dr. James Wootton
Dr. Joseph Renes
Prof. Dr. Renato Renner

Abstract

Quantum error correction promises a viable path to realizing scalable, fault-tolerant quantum devices. In recent years, many groundbreaking experiments have been conducted showcasing the progress of experimental quantum error correction. The success of these experiments depends, among other things, on the classical decoding method used. However, these classical algorithms traditionally only interpret binary data, ignoring valuable information contained in the complete analog measurement data. To address this issue, Pattison *et al.* proposed a method that fully utilizes this analog data and improves decoding performance [1]. This thesis presents experimental results from superconducting hardware that incorporate this richer "soft information" into the decoding process. We further advance the method proposed by Pattison *et al.* by a heuristic extension tailored to manage leakage errors. Our empirical results, utilizing repetition codes, reveal that soft decoding not only improves error rates but also elevates the threshold by up to 25%, underscoring soft information's notable potential to improve quantum error correction exponentially. Furthermore, we address the trade-off between the volume of information acquired from the measurements and the performance of the decoding operation. Our findings reveal that the decoding performance saturates as information increases. Notably, this plateau is reached across all code distances for the same information quantity. In our experiments, this convergence is reached at one byte of information, suggesting that the amount of information retrieved from measurements can be significantly reduced without compromising performance. This reduction evidences the practicality of soft information decoding in time-critical scenarios such as real-time decoding. Finally, we explore the effect of leakage on soft decoding and demonstrate that our heuristic extension to handle leakage primarily contributes to the observed benefits of soft decoding. However, its effectiveness decreases at high leakage rates, suggesting it is optimally used at intermediate levels.

Acknowledgements

I am grateful to Prof. Renato Renner and Dr. Stefan Woerner for making this collaboration possible within their groups, which allowed me to work on this exceptional project.

I extend my deepest gratitude to my supervisor, Dr. James Wootton, whose intuitive approach to problem-solving and infectious humor not only made this project formative but also really enjoyable.

I want to thank Dr. Joseph Renes for being my academic supervisor and also thank him and Dr. Christophe Piveteau for their brilliant QEC lecture, which sparked my interest in this incredible field.

I am indebted to Dr. Bence Hetényi for his extensive support and enriching discussions, which were crucial to the success of my thesis. My appreciation also goes to Ian Hesner and Conrad Haupt for patiently answering my endless questions.

Lastly, I am thankful to the Quantum Computational Science group—Samuele Piccinelli, Antony Gandon, Sabina Dragoi, Alexander Jürgens, Anastasia Agathangelou, Laura Mismetti, and everyone else. They were a major reason I looked forward to coming to the lab every day.

Contents

Chapter 1

Introduction

Quantum computers can provide asymptotically faster computation times than their classical counterparts in specific applications [2, 3], e.g., cryptography, material sciences, or chemistry [4, 5, 6, 7, 8]. However, outperforming CMOS-based machines requires the availability of large-scale quantum systems. From a hardware perspective, the fabrication of such devices remains a tremendous challenge. This challenge is further compounded by the need to maintain proper coherence while processing quantum information, a critical requirement for broad-scale devices. Present methods for storing and manipulating quantum data are highly susceptible to disturbances, and even minor environmental interactions can significantly impair the information being processed and compromise the system's fault tolerance.

Quantum error correction is fundamentally recognized as a central strategy for deploying fault-tolerant devices at scale. It promises exponential error suppression, closing the divide 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.

These advancements are particularly evident in implementing the surface code, a leading candidate for fault-tolerant quantum computation. Progress has been marked by an upward path from implementing a distance-three ($d = 3$) surface code [9, 10], advancing to a distance-five ($d = 5$) surface code [11], and culminating in a distance-seven ($d = 7$) surface code that has also demonstrated improvements in the fidelities of logical entangling gates with increasing code distance [12]. Furthermore, recent experiments have achieved break-even error correction performance, matching or surpassing their physical counterparts. Key milestones include achieving break-even fidelity in the initialization of logical qubits within a color code [12], preparing logical Bell pairs with error rates 800 times lower than their physical level [13], and successfully distilling magic states with fidelity beyond the break-even point [14]. These developments 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 mid-circuit measurements, and the performance of the classical decoding algorithm used to process measurement outcomes and determine corrective operations. Usually, outcomes from mid-circuit 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 degrade decoding accuracy.

Pattison *et al.* developed a method to circumvent this loss by incorporating the full range of analog data, also known as soft information, into the decoding process [1]. Their simulations indicated that using soft information in surface code decoding can raise the code's threshold by 25% compared to decoders that utilize only binary, or "hard," information. This result reveals a substantial improvement in the exponential scaling of error suppression, going 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.

1.1 Previous works

In addition to the work by Pattison *et al.*, the benefit of employing soft information has been validated through simulations using neural network decoders [15, 16]. On the hardware front, soft decoding has been implemented in several settings: for improving the readout fidelity on a Si/SiGe two-qubit system[1] [17], and for superconducting qubits using both a distance-three ($d = 3$) heavy-hexagon code and, very recently, a distance-three ($d = 3$) two-dimensional repetition code [18]. However, these hardware experiments were limited to at most $d = 3$ codes, which restricted their ability to evaluate and compare thresholds between soft and hard decoding—a significant and anticipated aspect of this research.

Furthermore, these experiments, involving $T = 15$, 10, and 16 rounds of stabilizer measurements, respectively, managed to post-select for leakage errors—transitions to higher energy states detrimental to quantum error correction [19]. This post-selection was feasible because this number of rounds helped restrict the accumulation of leakage, which typically increases with the duration of the experiment. Consequently, the issue of leakage and its impact on soft decoding, also not addressed in the initial studies by Pattison *et al.* or subsequent simulation efforts, remains a critical area for further investigation.

This thesis aims to experimentally analyze the threshold benefit of soft decod-

[1] With one qubit serving as the readout ancilla for the other qubit.

ing compared to hard decoding by employing various distances, from $d = 3$ to $d = 51$, of the repetition code and analyzing the exponential scaling behavior of both decoders with code distance. Additionally, given that soft information error correction involves processing and transmitting a greater volume of information, we will evaluate how the performance of soft decoding is influenced when we retrieve 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 will 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.

1.2 Outline of thesis

We organize this thesis as follows: Chapter 2 introduces quantum error correction, addressing the decoding aspect and the intricacies associated with implementing quantum codes on hardware. Chapter 3 explores the principle of soft decoding, covering the measurement process, the errors incurred due to information loss during classification, and the practical implementation of soft information decoding. Chapter 4 covers the main results of this thesis. It begins with a brief introduction to repetition codes. Then, it presents results on the threshold comparison between soft and hard decoding, the information-performance tradeoff, and the impact of leakage on soft decoding.

1.3 Overview of contributions

The main novel additions to the field of quantum error correction in this thesis are:

- The acquisition of empirical data for large-distance soft decoding experiments from Chapter 4.
- The heuristic method to treat leakage in soft decoding outlined in Section 3.3.5.
- The empirical evidence of a threshold difference between soft and hard decoding, showing up to a 25% improvement for repetition codes on superconducting hardware in Section 4.4.
- The analysis of the information-performance tradeoff in soft decoding discussed in 4.3.

Chapter 2

Quantum error correction preliminaries

2.1 Quantum error correction framework

Quantum error correction faced early skepticism primarily owing to the no-cloning theorem, preventing simple replication for error correction, and the analog nature of quantum information, seeming to demand arbitrary precise error correction. *Shor's code* overcame these challenges by distributing the information of a logical qubit to several physical qubits through entanglement, thus evading the no-cloning theorem. It then employs the measurement of *stabilizers*—observables that preserve the logical subspace—to enable detection and correction of specific error groups. The measurement process effectively *digitizes* errors through the wavefunction collapse, addressing the second issue and avoiding the need for arbitrary precise control to correct analog errors.

To understand and describe how quantum error correction codes like the Shor code work, we will first introduce the Pauli group and its symplectic structure. Then, we will address the stabilizer formalism and discuss the decoding aspect of quantum error correction codes.

2.1.1 Symplectic structure of the Pauli group

We will first introduce the *Pauli group* and its *symplectic structure* to arrive at a general framework to describe quantum error correcting codes. The Pauli group P_n consists of tensor products of Pauli operators acting on individual qubits:

$$P_n = \{i^c P_1 \otimes \cdots \otimes P_n \mid c \in \mathbb{Z}_4,\ P_i \in \{I, X, Y, Z\}\}, \tag{2.1}$$

where I is the identity operator, and X, Y, and Z are the Pauli operators acting on a single qubit. We can further simplify this expression by considering that $Y = iXZ$,

which allows us to express the Pauli group as:

$$P_n = \left\{ i^c (-1)^t X^{\mathbf{u}} Z^{\mathbf{v}} \mid c, t \in \mathbb{F}_2,\ \mathbf{u}, \mathbf{v} \in \mathbb{F}_2^n \right\}. \tag{2.2}$$

Consequently, each $g \in P_n$ is characterized by a vector $\mathbf{w} = (\mathbf{u}, \mathbf{v}) \in \mathbb{F}_2^{2n}$ and a phase $(t, c) \in \mathbb{F}_2^2$. Because the Pauli's are Hermitian and unitary, $X^{\mathbf{u}} X^{\mathbf{u}'} = X^{\mathbf{u}+\mathbf{u}'}$ and $Z^{\mathbf{v}} Z^{\mathbf{v}'} = Z^{\mathbf{v}+\mathbf{v}'}$. Also, X anticommutes with Z, consequently $X^{\mathbf{u}} Z^{\mathbf{v}} = (-1)^{\mathbf{u}\cdot\mathbf{v}} Z^{\mathbf{v}} X^{\mathbf{u}}$. This enables us to write the product of two Pauli operators as:

$$\left(i^c (-1)^t X^{\mathbf{u}} Z^{\mathbf{v}}\right) \left(i^{c'} (-1)^{t'} X^{\mathbf{u}'} Z^{\mathbf{v}'}\right) = i^{c+c'} (-1)^{t+t'} (-1)^{\mathbf{u}'\cdot\mathbf{v}} X^{\mathbf{u}+\mathbf{u}'} Z^{\mathbf{v}+\mathbf{v}'}. \tag{2.3}$$

The commutation relations between the Pauli operators give rise to a *symplectic structure* on the Pauli group. According to Equation 2.3, two Pauli operators g and g' commute if and only if $\mathbf{u} \cdot \mathbf{v} + \mathbf{u}' \cdot \mathbf{v}' = 0$. By defining a *bilinear symplectic form* on the vector space $\mathbb{F}_2^{2n}$ as:

$$\left\langle \mathbf{w}, \mathbf{w}' \right\rangle := \mathbf{w}^T J \mathbf{w}' \quad \text{with} \quad J = \begin{pmatrix} 0 & I_n \\ I_n & 0 \end{pmatrix}, \tag{2.4}$$

we can express the commutation relations as $\langle \mathbf{w}, \mathbf{w}' \rangle = 0$. Consequently, the Pauli group P_n forms a *symplectic vector space* with respect to the symplectic form $\langle \cdot, \cdot \rangle$.

This symplectic structure results in a symplectic basis of $\mathbb{F}_2^{2n}$ with elements $e_1, \ldots, e_n$ and $f_1, \ldots, f_n$ such that $\langle e_i, e_j \rangle = \langle f_i, f_j \rangle = 0$ and $\langle e_i, f_j \rangle = \delta_{ij}$. Interestingly, any basis of $\mathbb{F}_2^{2n}$ corresponds directly to a set of generators of the Pauli group:

$$P_n = \langle g_1, \ldots, g_n, \tilde{g}_1, \ldots, \tilde{g}_n \rangle. \tag{2.5}$$

For instance, the cartesian basis of $\mathbb{F}_2^{2n}$ corresponds to X and Z operators acting on individual qubits. These generators can be grouped into pairs of anticommuting operators $(g_j, \tilde{g}_j)$ such that:

$$\begin{cases} \{g_j, \tilde{g}_j\} = 0 & \text{for } j = k, \\ [g_j, g_k] = [\tilde{g}_j, \tilde{g}_k] = [g_j, \tilde{g}_k] = 0 & \text{for } j \neq k. \end{cases} \tag{2.6}$$

I.e., the generators g_j and $\tilde{g}_j$ anticommute with each other and commute with all other generators.

2.1.2 Stabilizer formalism

In this section, we will briefly introduce the *stabilizer formalism*, which we will use throughout this thesis. For a more detailed introduction, we refer to [2, 20, 21].

Consider a quantum code with n physical qubits encoding k logical qubits in the

subspace $\mathcal{C} \subset \mathcal{H}^{\otimes n}$. The Pauli group acting on the n qubits can be decomposed into three abelian subgroups: the *stabilizer subgroup* S, the *destabilizer subgroup* D, and the *logical subgroup* L.

To describe this decomposition, we will first look at the representation on $\mathbb{F}_2^{2n}$ of the Pauli group. For any codespace $\mathcal{C}$, we can define the subspace $\mathcal{S} \subset \mathbb{F}_2^{2n}$ that leaves the codespace invariant up to a sign, i.e., for any $\mathbf{w} = (\mathbf{u}, \mathbf{v}) \in \mathcal{S}$, $(t, c) \in \mathbb{F}_2$ and any $|\psi\rangle_L \in \mathcal{C}$:

$$\left(i^c (-1)^t X^{\mathbf{u}} Z^{\mathbf{v}}\right) \cdot |\psi\rangle_L = \pm |\psi\rangle_L. \tag{2.7}$$

Pairing up the basis elements of $\mathcal{S} \subset \mathbb{F}_2^{2n}$ into pairs $(g_i, \tilde{g}_i)$ as discussed in the previous section, we can define the stabilizer subgroup S as the group generated by the g_i and the destabilizer subgroup D as the group generated by the $\tilde{g}_i$. I. e., for all $g_i \in S$ and all $|\psi\rangle_L \in \mathcal{C}$:

$$g_i \cdot |\psi\rangle_L = (+1) \cdot |\psi\rangle_L, \tag{2.8}$$

and for all $\tilde{g}_i \in D$:

$$\tilde{g}_i \cdot |\psi\rangle_L = (-1) \cdot |\psi\rangle_L. \tag{2.9}$$

Consequently, the stabilizers uniquely define the logical subspace $\mathcal{C}$ as their $+1$ eigenspace, and we can characterize a quantum code directly by its stabilizers.

We will then define the orthogonal complement $\mathcal{S}^\perp \subset \mathbb{F}_2^{2n}$ of $\mathcal{S}$ with respect to the symplectic form, i.e., $\langle \mathbf{w}, \mathbf{w}' \rangle = 0$ for all $\mathbf{w} \in \mathcal{S}$ and $\mathbf{w}' \in \mathcal{S}^\perp$. The *logical subgroup* L is then defined as the group generated by the basis elements of $\mathcal{S}^\perp$. Consequently, the logical subgroup L is the group of operators that commute with all the stabilizers and destabilizers but act non-trivially on the logical subspace $\mathcal{C}$. Pairing up the basis elements of $\mathcal{S}^\perp$ into pairs $(l_i, \tilde{l}_i)$, we can define the logical Z_L and X_L operators on each logical qubit as the l_i and $\tilde{l}_i$ respectively. We will call the *distance* the weight of the smallest possible weight of a logical generator. The distance corresponds to the minimum weight an operator must carry to logically affect the codespace. In addition, using the rank-zero theorem, we can show that the dimension of the logical subspace is $k = n - m$, where $m = \text{rank}(\mathcal{S})$ is the number of independent stabilizer generators. This dimension corresponds to the k X_L and Z_L operators. In summary, for a quantum code on n qubits with m stabilizer operators generating S, we have $k = n - m$ logical qubits. Each logical qubit has a Z_L and X_L operator, generating the logical subgroup L. Specifying the stabilizers is sufficient to fully characterize the quantum code as the logical and destabilizer operators can be derived from the symplectic structure of the Pauli group.

Let $\mathtt{E}$ be an error acting on the qubits. If we measure the m stabilizer generators $g_i \in S$ sequentially, the state will collapse to either the $+1$ or -1 eigenstate. Overloading the notation, we will name the outcome of the measurements

$\mathbf{m}[\mathtt{E}] = (m_1, \ldots, m_m) \in \mathbb{F}_2^m$. These outcomes depend on the commutation relations between the error and the stabilizers and are given by:

$$m_i = \begin{cases} +1 & \text{if } [\mathtt{E}, g_i] = 0, \\ -1 & \text{if } \{\mathtt{E}, g_i\} = 0. \end{cases} \tag{2.10}$$

Consequently, the stabilizer measurements only detect the error if it anti-commutes with at least one of the generators. More generally, the error $\mathtt{E}$ can be decomposed into a stabilizer, destabilizer, and logical part, i.e.,

$$\mathtt{E} = \mathtt{E}_\mathtt{S}\mathtt{E}_\mathtt{D}\mathtt{E}_\mathtt{L}, \tag{2.11}$$

such that $\mathtt{E}_\mathtt{S} \in S$, $\mathtt{E}_\mathtt{D} \in D$, and $\mathtt{E}_\mathtt{L} \in L$. The measurement outcome vector will then be the sum of the outcome vectors of the stabilizer, destabilizer, and logical parts, i.e.,

$$\mathbf{m}[\mathtt{E}] = \mathbf{m}[\mathtt{E}_\mathtt{S}] + \mathbf{m}[\mathtt{E}_\mathtt{D}] + \mathbf{m}[\mathtt{E}_\mathtt{L}] = 0 + \mathbf{m}[\mathtt{E}_\mathtt{D}] + 0. \tag{2.12}$$

However, only the destabilizer part will have a non-trivial outcome. If the error $\mathtt{E}$ has a non-trivial logical part, it will affect the codespace $\mathcal{C}$, but the measurements will not detect it. This naturally leads us to the next section, where we will discuss the decoding of quantum error correction codes, i.e., how to find the most likely error given the stabilizer measurements.

2.2 Decoding of quantum codes

The decoding aspect of quantum codes is crucial for their effectiveness in a practical setting. The goal of decoding is to find the *most likely logical error* (MLL) given the stabilizer measurements according to a noise model $\mathbb{P}_E[\mathtt{E}] \in [0, 1]$. A simple approach would just be to look for the *most likely error* $\mathrm{MLE} \in P_n$ that matches the measurement vector $\mathbf{m}$, i.e.,

$$\mathrm{MLE}(\mathbf{m}) = \underset{\mathtt{E}_\mathtt{S} \in S, \mathtt{E}_\mathtt{L} \in L}{\arg\max}\, \mathbb{P}_E[\mathtt{E}_\mathtt{S} \cdot \mathtt{E}_\mathtt{D}[\mathbf{m}] \cdot \mathtt{E}_\mathtt{L}]. \tag{2.13}$$

But the stabilizers have a trivial effect on the codespace $\mathcal{C}$, allowing us to marginalize over them to get the most likely logical error $\mathrm{MLL} \in L$:

$$\mathrm{MLL}(\mathbf{m}) = \underset{\mathtt{E}_\mathtt{L} \in L}{\arg\max} \sum_{\mathtt{E}_\mathtt{S} \in S} \mathbb{P}_E[\mathtt{E}_\mathtt{S} \cdot \mathtt{E}_\mathtt{D}[\mathbf{m}] \cdot \mathtt{E}_\mathtt{L}]. \tag{2.14}$$

For a given code, finding the MLL is *optimal* in the sense that it minimizes the probability of logical errors after correcting the MLL on the physical qubits.

Computing the MLL is more challenging than computing the MLE. However,

finding the MLE is already a non-trivial task as it is known to be NP-complete [22], for a code of n qubits with m stabilizers, 2^{2n-m} possible errors match the measurement outcomes. Consequently, the number of potential faults grows exponentially with qubits, making an exhaustive search infeasible for large codes.

For practical quantum error correction, we need efficient decoding algorithms. This means that exploiting the structure of the code is necessary to find the MLE or an *approximation* of it. In general, multiple decoding algorithms exist, but there is often a tradeoff between the complexity and the accuracy of its approximation to the MLE.

2.2.1 Cycle codes and decoding graphs

In the following, we will only consider exploiting one specific type of code, namely *almost cyclic CSS codes*. *Cycle codes* are codes where each qubit is involved in exactly two stabilizers. *Almost cycle codes*, on the other hand, are codes that have qubits involved in *one and two* stabilizers. Finally, simplifying the definition, CSS codes are characterized by stabilizer generators that consist exclusively of either X or Z-type operators. Almost cycle CSS codes can be made cyclic by adding a stabilizer of the corresponding type, acting on all qubits involved in only one stabilizer. We assign the parity of all original stabilizer measurements to the result of this additional stabilizer. The codes presented in this thesis are all almost cycle CSS codes.

The cycle structure of stabilizer codes ensures that exactly two stabilizers detect any single-qubit error. For multi-qubit errors (error strings), the non-trivial stabilizer outcomes are found at the string's ends, allowing efficient error finding using a *decoding graph*[1]. Since all qubits are involved in precisely two stabilizers, this graph represents qubits as edges linking two stabilizers as nodes. We then mark the non-trivial stabilizer outcomes as non-trivial nodes. The edges in the graph are weighted according to the noise model $\mathbb{P}_E[\cdot]$, where the weight of an edge w_{ij} between two stabilizers i and j is given by:

$$w_{ij} = -\log(\mathbb{P}_E[\mathtt{E_{ij}}]/(1 - \mathbb{P}_E[\mathtt{E_{ij}}])), \tag{2.15}$$

where $\mathbb{P}_E[\mathtt{E_{ij}}]$ is the probability of an error happening on the qubit involved in stabilizers i and j, which both stabilizers would detect. Hence, a high error probability would result in a low weight.

To explain the choice of Equation 2.15, consider the error string $\mathbf{e}$ such that $\mathbf{e}[i] = 1$ if the qubit i is faulty and $\mathbf{e}[i] = 0$ otherwise. Let $p_i = \mathbb{P}_E[i]$, then the probability

[1] We will skip the definition of *tanner graphs* and directly define the *decoding graph* as we will only consider cyclic codes.

of the error string $\mathbf{e}$ is given by:

$$\mathbb{P}_E[\mathbf{e}] = \prod_i p_i^{\mathbf{e}[i]}(1-p_i)^{1-\mathbf{e}[i]} = \prod_i (1-p_i) \prod_i \left(\frac{p_i}{1-p_i}\right)^{\mathbf{e}[i]}. \tag{2.16}$$

Taking the logarithm of this expression, we get:

$$-\log(\mathbb{P}_E[\mathbf{e}]) = -\sum_i \log(1-p_i) + \sum_i \mathbf{e}[i] w_i. \tag{2.17}$$

Consequently, the probability of the error $\mathbf{e}$ amounts to the sum of the weights of the qubits involved in the error string plus a constant term. As a result, finding the MLE corresponds to finding the shortest weighted path pairing of the non-trivial nodes. The edges in the pairing path then correspond to the qubits involved in the MLE error. This problem is known as the *minimum weight perfect matching* (MWPM) problem, which can be solved using the *blossom algorithm* [23]. Its runtime goes as $\mathcal{O}(V^3)$, where V is the number of vertices in the decoding graph. As we have at most $2n$ vertices in the decoding graph for cyclic codes, the runtime is polynomial in the number of qubits. More decoding methods exist on graphs, such as the *Union-Find* algorithm [24], which is almost linear in time but will have a higher approximation error. In this work, we will only use the MWPM algorithm for decoding. However, the proposed methods are based on the decoding graph and thus can be utilized for other decoding algorithms.

To illustrate the principle of a decoding graph, we utilize the simplest example of an almost cyclic code: a linear arrangement of n qubits interleaved by $n-1$ stabilizers that link them. This setup effectively represents the repetition code, as we will see in Chapter 4. Consider, for example, a decoding graph for 4 qubits and 3 stabilizers, as depicted in Figure 2.1. To render this graph cyclic, we introduce an additional stabilizer node that links the qubits involved in only one stabilizer. Since this node's value reflects the parity of the other nodes, its constant depiction is unnecessary. By cutting the chain, we transform the graph into a one-dimensional line, leaving a dangling edge. It is crucial, however, to remember that we can always match this dangling edge to the boundary.

2.2.2 Faulty stabilizer measurements

Complications arise if the stabilizer measurements themselves are faulty. To remedy this, we need to *repeat the measurements* enough times. If the measurement error probabilities $\mathbb{P}_E[\mathtt{E_m}]$ are comparable to the qubit error probabilities $\mathbb{P}_E[\mathtt{E_q}]$, repeating the measurements $T \gg d$ times, where d is the distance of the code, is equivalent to having optimal *infinite memory* up to a negligible amount[2] [25].

[2]This assumes that error correlations decay exponentially in time.

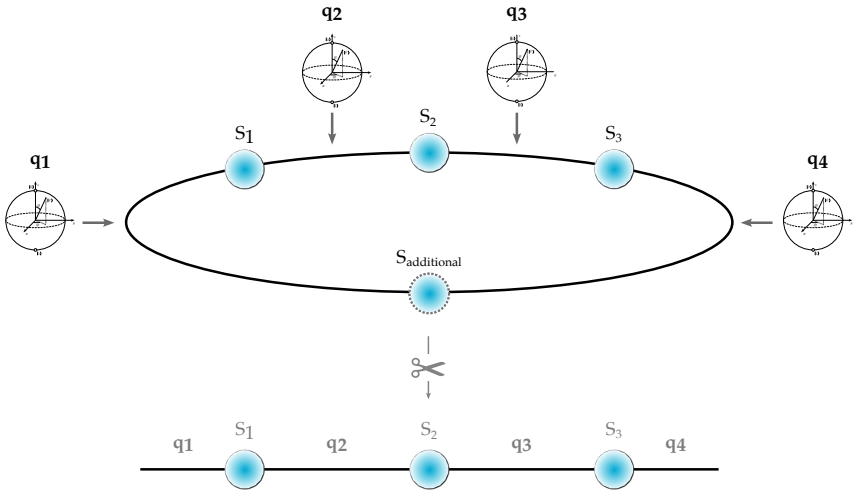


Figure 2.1: Decoding graph for a linear arrangement of 4 qubits (q_i), represented by Bloch spheres, interconnected by 3 stabilizers (S_i), shown as blue nodes. An extra stabilizer node with a dashed boundary is introduced to make the graph cyclic. This additional node carries the parity of the other nodes and can thus be discarded, resulting in a line graph.

To decode this additional information, we extend the decoding graph with a time dimension, repeating and appending the previous graph T times. We call this decoding graph G_T. We then compute the non-trivial nodes of the decoding graph as the changes in outcomes between the repeating nodes in time direction. These non-trivial nodes represent the *syndrome vector* $\mathbf{s}$. This vector is obtained by computing changes over time between consecutive measurement vectors using XOR operations:

$$\mathbf{s}_t = \mathbf{m}_t \oplus \mathbf{m}_{t-1}, \tag{2.18}$$

where t denotes the time index. Consequently, a single measurement error would manifest in two successive non-trivial syndrome nodes. Hence, we can represent the measurement error channels as additional "vertical" edges between the repeating nodes. The principle is illustrated in Figure 2.2. We then weigh the additional edges according to the probability of a measurement error happening $\mathbb{P}_E[\mathtt{E}_\mathtt{m}]$.

More generally, any error channel $\mathcal{E}$ that "flips" pairs of stabilizers can be represented as an edge connecting the respective stabilizers with its weight derived according to the noise model $\mathbb{P}_E[\mathcal{E}]$.

Following this procedure, qubit and measurement errors manifest similarly in the decoding graph, with error strings reflecting non-trivial nodes at their ends. Consequently, finding the ML, including measurement errors, is equivalent to finding the shortest path pairing as before.

More broadly, any group of error channels $\{\mathcal{E}_i\}$ that induce the same pairs of non-trivial syndromes can be considered equivalent from a decoding perspective. By

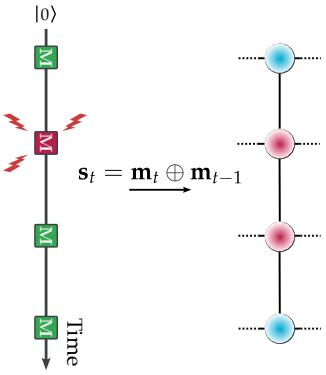


Figure 2.2: Single stabilizer column of the decoding graph G_T with its corresponding measurement timeline. A measurement error occurs at time $t = 2$, resulting in non-trivial syndrome nodes at $t = 2$ and $t = 3$.

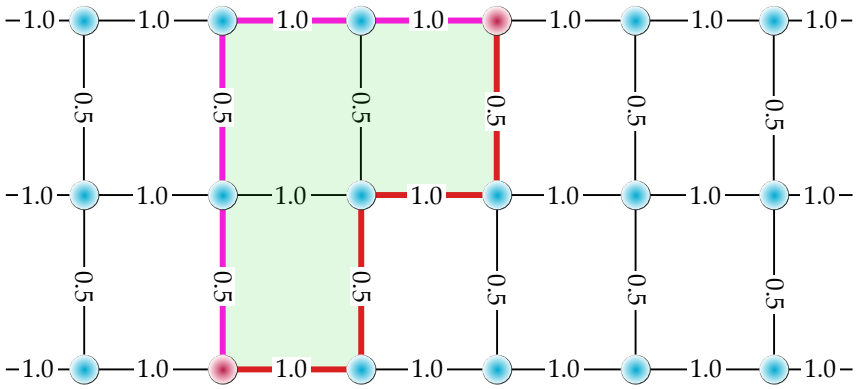


Figure 2.3: Decoding graph for a linear arrangement of 7 qubits and $T = 2$ rounds of measurements. The additional edges between the repeating nodes represent the measurement errors. The weights on the edges are symbolic and illustrate a noise model with higher measurement error than qubit error probabilities. Shown are two non-trivial syndrome nodes and two potential matching paths of equal weight, the combination of which creates a closed loop.

adding the corresponding edges, weighted according to the noise model $\mathbb{P}_E[\{\mathcal{E}_i\}]$, we can collectively decode these channels using MWPM.

We depict the modified decoding graph of Figure 2.1 with multiple rounds of measurements in Figure 2.3 with symbolic weights. The graph displays two non-trivial syndromes and two potential matching paths of equal weight. These paths combine to form a closed loop, indicating that the detected errors are equivalent modulo a product of stabilizers. This loop feature is typical of topological codes, although we will not explore its implications in this thesis.

Consequently, to decode an experiment with faulty stabilizer measurements, we must run multiple cycles of stabilizer measurements and decode the results on a graph with an additional time dimension. As the reader might have noticed, this adds a lot of structural complexity to the decoding. This extra complexity implies

that inaccuracies in the noise model used to weight the graph are more likely to compound. Hence, the decoding performance is more sensitive to the exactness of our noise modeling. This thesis focuses on utilizing more information from the measurements to address this issue by improving the accuracy of our noise model.

2.3 Hardware considerations for error correction

In subsequent chapters, we will discuss experimental results obtained from superconducting quantum devices. However, implementing quantum codes on physical hardware introduces specific challenges that merit detailed exploration.

2.3.1 Implementing quantum codes on hardware

Despite the objective of performing error correction during logical computations, the necessitated real-time error correction is challenging to achieve and has not yet been fully realized on hardware. Similarly to previous experiments, we will focus on *post-experiment decoding* experiments, which are equivalent to *quantum memory* experiments. For this type of experiment, we prepare a logical state, perform several rounds of stabilizer measurements, and, finally, do a logical readout of the code qubits, aligning with the initially prepared logical state.

The purpose of the final readout is two-fold. First, it allows us to estimate the fidelity of the decoder by comparing the predicted logical error with the actual logical error inferred from the logical readout. We say the decoder failed if the actual logical error differs from the predicted error. Second, we artificially compute a last round of stabilizer measurements from the code qubit measurements. This last round of artificial stabilizer outcomes is effectively perfect because any fault in the measurements themselves is equivalent to a code qubit error, and we will treat it as such. Having a last round of "perfect" stabilizer measurements ensures that the *time boundary* does not absorb measurement errors, making them undetectable. In other words, it ensures that measurement errors always "flip" two syndromes, which is crucial to not mistake a measurement error for a code qubit error [26].

Reading stabilizers requires measuring a unitary operator whose eigenvalues are ± 1. Since most quantum computing platforms only natively support computational basis readouts, stabilizer measurement involves applying the controlled unitary to an ancilla qubit, flanked by two Hadamard gates, followed by its readout, as seen in Figure 2.4. However, this process creates more errors as the gates to implement the controlled unitary are faulty. Moreover, the controlled gate enables the *propagation* of faults between the ancilla and the data qubits[3]. Hence, faulty stabi-

[3]This is generally critical for the fault tolerance of the code, but we will only discuss this aspect for repetition codes in later chapters.

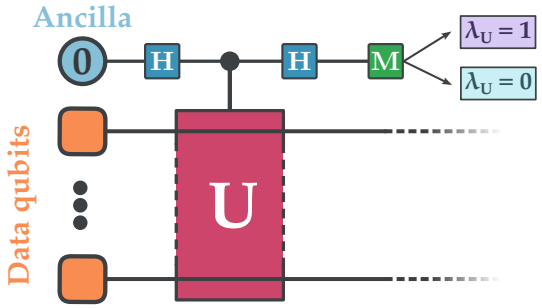


Figure 2.4: Circuit to implement the readout of a unitary U on multiple qubits using an ancilla qubit. The ancilla qubit is prepared in the $|+x\rangle$ state, entangled with the data qubits using the controlled-U gate, and read out using a Hadamard gate followed by a computational basis measurement. The outcome of the measurement corresponds to the eigenvalue of the unitary U.

lizer measurements can arise from various error sources associated with the gate operations used in the readout process.

After measuring the stabilizer, one might consider resetting the ancilla to the $|0\rangle$ state to prevent additional error propagation. However, resetting the qubit is a non-trivial task, requiring either waiting for the qubit to decay to the ground state or applying an *active reset* operation. The former would mean that the code qubits in the system must be idle for as long as the ancillas decay, causing them to decay themselves. The latter would require applying an X gate conditioned on the measurement outcome. However, this would require *dynamic operations*, which also cause idling times and specific errors. To circumvent these issues, we do not reset the ancilla qubit at all. We then compute the changes in outcomes in the measurement to simulate a scenario as if resets had been performed. Let $\mathbf{m}_t^{\neg r}$ be the measurement outcomes at time t without resetting the ancilla qubits. We can then write the *corrected* measurement outcomes at time t as

$$\mathbf{m}_t = \mathbf{m}_t^{\neg r} \oplus \mathbf{m}_{t-1}^{\neg r}. \tag{2.19}$$

We then use the corrected outcomes $\mathbf{m}_t$ to compute the syndrome vector $\mathbf{s}_t$ as before. We illustrate this process in Figure 2.5.

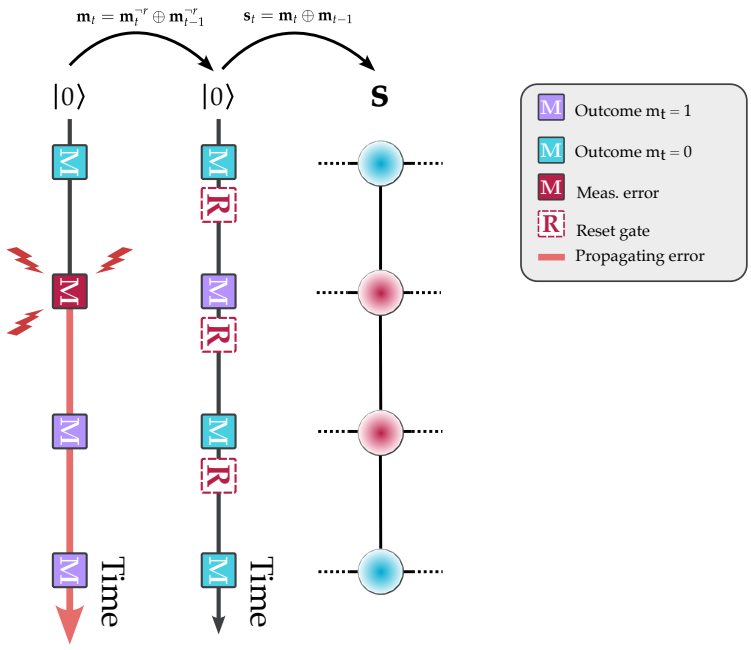


Figure 2.5: Timeline of measurement outcomes without resets, the inferred corrected outcomes, and its corresponding single stabilizer column of the decoding graph G_T. A measurement error occurs at time $t = 2$. We correct the outcomes to mimic a scenario with resets. Following these corrections, the syndrome is computed from the simulated outcomes, revealing two non-trivial syndrome nodes at times $t = 2$ and $t = 3$ as in Figure 2.2. Meas., measurement.

2.3.2 Hardware inspired noise model

As outlined in Section 2.2.1, employing decoding graphs for quantum error correction necessitates a noise model. We are keen to experiment with real hardware via a quantum circuit, so we build upon the *circuit-level noise model*. This model directly accounts for the explicit circuit to implement the code and its stabilizer measurements. It presumes that *depolarizing channel* acts on qubits following each gate operation, affecting a single or two qubits based on the gate type, with distinct probabilities. This model describes the noise acting on the code qubits and the errors emerging from the stabilizer measurement implementation.

This noise model allows us to describe another error channel that arises from error propagation between the ancilla and the code qubits. This error channel is specific to the entangling gate ordering in the circuit and, hence, particular to the code. We exemplify it with our linear code and set the stabilizer measurements as ZZ parity readouts[4], as depicted in Figure 2.6. The displayed error between the CNOTs will cause two *diagonal* stabilizer measurements—separated in space and time—to be non-trivial. In general, correctly ordering the CNOT gates ensures that this error channel always flips pairs of syndrome nodes. Thus, as discussed

[4]We will further explain this choice of stabilizer measurement in Chapter 4.

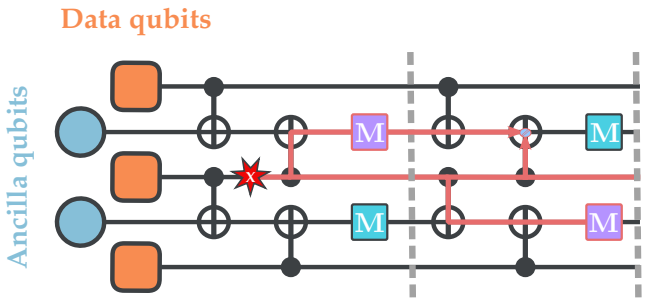


Figure 2.6: Circuit to implement the readout of a ZZ stabilizer measurement for $T = 2$ rounds of measurements and $n = 3$ qubits. The error that occurs between two CNOTs on the code qubit causes two non-trivial measurements separated in space and time.

in Section 2.2.2, if we add the corresponding edge connecting the two syndrome nodes, we can decode errors on the decoding graph by matching as before.

To make the noise model more realistic, we extend it to include *idling errors* and *measurement errors* separately. Idling errors occur on the code qubits at the end of each stabilizer measurement round. These errors appear due to the code qubits idling while we perform operations on the ancilla qubits. We only consider two types of idling errors, *relaxation* and *dephasing*, related to the T_1 and T_2 times respectively [27]. Relaxation is related to the qubit's decay from the excited state to the ground state. We model it by the *amplitude damping channel* with the Kraus operators

$$K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix}, \quad K_1 = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix}. \tag{2.20}$$

Dephasing is related to the qubit's loss of coherence, modeled by the *phase damping channel* with the Kraus operators

$$K_0' = \sqrt{1-p'} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad K_1' = \sqrt{p'} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.21}$$

For compatibility with our simulation and decoding tools, we approximate these channels as Pauli channels using *Pauli twirling* [28, 29]. For an idling time τ, both channels can be approximated together as a single Pauli channel

$$\rho \to p_I I\rho I + p_X X\rho X + p_Y Y\rho Y + p_Z Z\rho Z, \tag{2.22}$$

with $p_I = (1 - p_X - p_Y - p_Z)$ and Pauli error probabilities

$$p_X = p_Y = \frac{1 - e^{-\tau/T_1}}{4}, \tag{2.23}$$

$$p_Z = \frac{1 - e^{-\tau/T_2}}{2} - \frac{1 - e^{-\tau/T_1}}{4}. \tag{2.24}$$

It is important to note that this approximation neglects any off-diagonal Pauli terms, making the channel symmetric, which is not the case for the amplitude-damping channel. However, this simplification is necessary to make the noisy quantum circuit an efficiently (classically) simulable Clifford circuit. This is crucial to be able to simulate our quantum experiments later on.

Regarding measurement errors, the circuit-level noise model accounts for faults on the ancilla qubits due to the controlled unitary circuitry. These errors cause faulty stabilizer measurements by altering the ancilla qubit state directly. We will call these errors causing faulty outcomes that agree with the ancilla state *hard flips*. On the other hand, *soft flips* are errors that only affect the measurement outcome without altering the ancilla state. As the circuit-level noise model primarily accounts for hard flips induced by gate errors during stabilizer measurements, we expand this model to encompass both hard and soft flips emerging directly from the measurement process. Hard flips, altering the ancilla state and measurement outcome, originate in an ancilla qubit relaxation or excitation during readout. We model these errors by a probabilistic X gate appearing with probability p_h pre-measurement. Soft flips, altering just the measurement outcome, are included in the noise model as a probabilistic flip of the measurement result that does not affect the ancilla with probability p_s. We will discuss the physical origin of these errors in the next chapter.

Finally, as described in Section 2.3.1, any measurement errors, soft or hard, in the final logical readout will be treated as a code qubit error. Consequently, if $p_e = \mathbb{P}_E[\mathtt{E}_c]$ is the probability of an error $\mathtt{E}_c$ happening on a code qubit c due to gate, idling, or propagation errors, we model the total probability of an error on that code qubit due to the additional measurement errors as

$$\begin{aligned}\mathbb{P}_E[\mathtt{E}_c^{\mathtt{tot}}] = \; & p_e(1-p_s)(1-p_h) \\ & + (1-p_e)p_s(1-p_h) \\ & + (1-p_e)(1-p_s)p_h \\ & + p_e p_s p_h.\end{aligned} \tag{2.25}$$

More generally, employing a Pauli noise model enables us to construct a decoding graph, or better, a hypergraph, for any quantum code, including non-cyclic ones. To do so, we represent each error channel as a hyperedge connecting the corresponding stabilizers. We then calculate its probability using Gottesman-Knill simulations [30] or similar Pauli tracing methods. In our experiments, we will use the Pauli tracing software Stim [31] to simulate the noise model and construct the decoding graph efficiently.

2.3.3 Error suppression factor Λ

We typically evaluate the merit 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. This value originates in the *threshold theorem*, which posits that we can execute a quantum circuit with G gates with an arbitrary final accuracy ϵ through the concatenation of error-correcting codes, provided the failure probability p of each hardware component is below p_{th}. The required gate counts for such an error-corrected circuit is given by:

$$\mathcal{O}(\text{poly}(\log G/\epsilon)G). \tag{2.26}$$

This number represents only a polylogarithmic increase over the number of gates in the original, uncorrected circuit [2]. The theorem implies that if the device's fault probability remains below the threshold, the logical error rate will exponentially decrease as the code size increases.

More precisely, assuming a constant probability of a fault p across all potential errors, the logical error rate ϵ_L decreases exponentially with respect to the *order of fault-tolerance* f, which is the maximum number of faults that the code can correct [28]:

$$\epsilon_L \propto \left(\frac{p}{p_{\text{th}}}\right)^{f+1}. \tag{2.27}$$

For stabilizer codes with distance d, the order of fault-tolerance is $f = \lfloor d/2 \rfloor$ as the code can detect up to $d-1$ errors but only unambiguously correct up to $\lfloor d/2 \rfloor$ errors. Consequently, the logical error rate ϵ_L exponentially decreases with the distance of the code as

$$\epsilon_L \propto \left(\frac{p}{p_{\text{th}}}\right)^{\lfloor \frac{d}{2} \rfloor + 1}. \tag{2.28}$$

The code in question does not solely determine the threshold value, as it is also significantly influenced by the decoding method [32]. Typically, researchers simulate the performance of a code and its decoder across varying error rates to estimate the threshold. However, as Kelly *et al.* point out, "the threshold error rate is not, however, a terribly meaningful experimental quantity" [26]. Determining this threshold in experiments would require precise manipulation of the noise model and, hence, gates with accurately controllable error rates—a daunting task.

Instead, we employ the *error suppression factor* Λ as a practical metric for comparing the performance of various codes and decoders [26]. We define Λ through the scaling of the logical error rate ϵ_L as

$$\epsilon_L \propto \left(\frac{1}{\Lambda}\right)^{\lfloor \frac{d}{2} \rfloor + 1}. \tag{2.29}$$

Consequently, Λ effectively measures *how far below the threshold the error rate of a system is*, without needing to determine the threshold itself.

Comparing Equation 2.29 with Equation 2.28, 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

$$\frac{p_{th,1}}{p_{th,2}} = \left(\frac{\Lambda_1}{\Lambda_2}\right). \tag{2.30}$$

When decoding multiple rounds of stabilizers, 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 [32]:

$$P_L = \frac{1}{2}\left(1 - (1 - 2\epsilon_L)^T\right). \tag{2.31}$$

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

$$P_L \approx T\epsilon_L. \tag{2.32}$$

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 apply Equation 2.29 to determine the Λ-factor. Utilizing this methodology, we will calculate the lambda factors to compare the performance of various decoders in the subsequent chapters.

Chapter 3

Quantum error correction with soft information

This chapter will discuss the theoretical and experimental aspects of quantum error correction with soft information. We will first discuss how the measurement process leads to so-called *soft flips* due to the analog nature of measurement outcomes in Section 3.1. We will then introduce a method to improve decoding by incorporating extra analog information from the measurements in Section 3.2. Finally, we will outline the experimental procedure to investigate the benefits of soft information in quantum error correction in Section 3.3.

3.1 Measurements on hardware induce soft flips

Measurements on quantum devices are inherently probabilistic processes that lead to errors in the measurement outcomes. To better understand these errors, we will first discuss the peculiarities of the measurement process and then propose a model to describe the faults that arise from it.

3.1.1 Measurement process on quantum devices

When considering the explicit circuit to implement stabilizer measurements, the used gates introduce errors that can alter stabilizer measurement outcomes, primarily through modifications to the ancilla qubit's state. Additionally, errors may arise from the measurement process, encompassing another category of errors that alters the outcomes without affecting the ancilla qubit. We first discuss the aspects of the measurement process that lead to these errors.

On most quantum devices, the measurement process culminates in a final continuous signal. We then discretize this analog signal into a binary value indicating the outcome of the measurement. This discretization process leads to *loss of infor-*

mation and *missclassification errors*. To describe these errors, we discuss readout on superconducting hardware, which is the focus of our experiments. However, the principles are broadly applicable across various quantum computing platforms, as discussed later.

In superconducting hardware, we perform qubit readouts by coupling a resonator to the qubit. The qubit's state then induces a *dispersive shift* in the resonator's frequency. To detect this shift, we apply a microwave tone to the resonator and measure the reflected signal, which has an altered amplitude and phase according to the qubit's state. We integrate this reflected signal to extract two key metrics: the *in-phase* (I) and *quadrature* (Q) components and plot them on the *IQ plane*. This plane has I and Q on its axes and visually represents the qubit's state through the distribution of I and Q points. Statistically, the I and Q points cluster into two distinct Gaussian distributions on the IQ plane, each representing one of the qubit's possible states. Figure 3.1 shows statistical IQ data obtained from a superconducting qubit. To ascertain the qubit's state from the reflected signal, we assess the likelihood of a data point belonging to each Gaussian and convert it into a binary outcome. The dispersive shift determines the separation between the two Gaussians, whereas the width of the Gaussian curves is dependent, among other things, on the duration of the integration window. Despite efforts to separate these distributions, a non-zero overlap exists, creating a situation where an IQ data point might fall into an area more likely associated with the incorrect state. Such occurrences lead to a *misclassification error*, commonly called a *soft flip*.

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 misclassification probability [33].

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 [34].

These examples underscore a common theme across quantum computing platforms: the necessity of managing and interpreting probabilistic signals derived

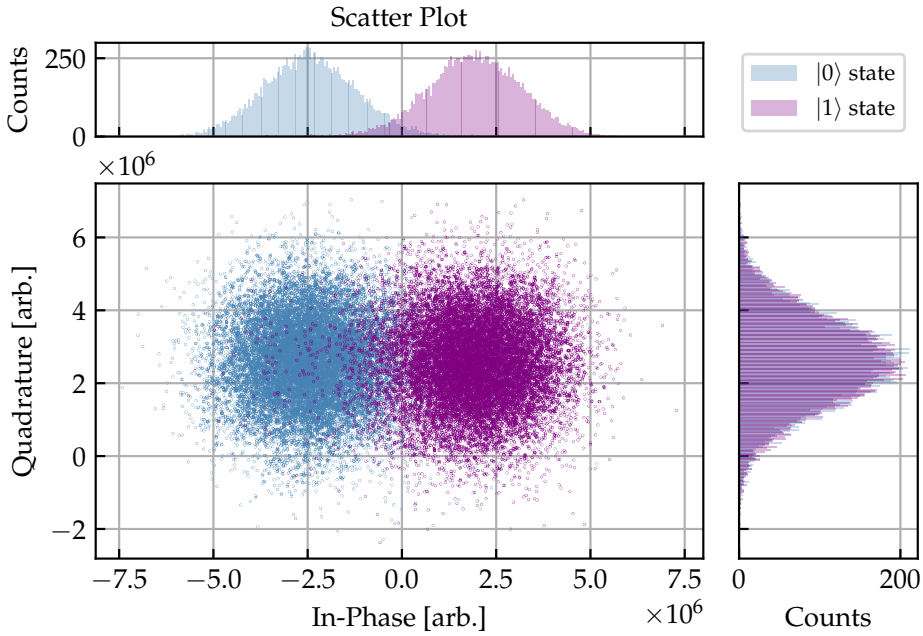


Figure 3.1: IQ data for qubit 72 of the `IBM Sherbrooke` device prepared in the $|0\rangle$ (blue) and $|1\rangle$ (violet) state. The data shows two distinct Gaussian distributions corresponding to the qubit's computational states.

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 misclassification errors, commonly referred to as soft flips.

3.1.2 Measurement process model

We will use the following model to describe these misclassifications in a *device-independent manner*. When measuring the qubit state, it first gets projected onto the eigenspace of the *true state* $\bar{z} \in \{0, 1\}$. We then observe a *soft outcome* μ according to the probability distribution $f^{\bar{z}}(\mu) = \mathbb{P}[\mu \mid \bar{z}]$. 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*:

$$\hat{z} = \begin{cases} 0 & \text{if } f^{(0)}(\mu) \geq f^{(1)}(\mu) \\ 1 & \text{otherwise} \end{cases}. \tag{3.1}$$

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

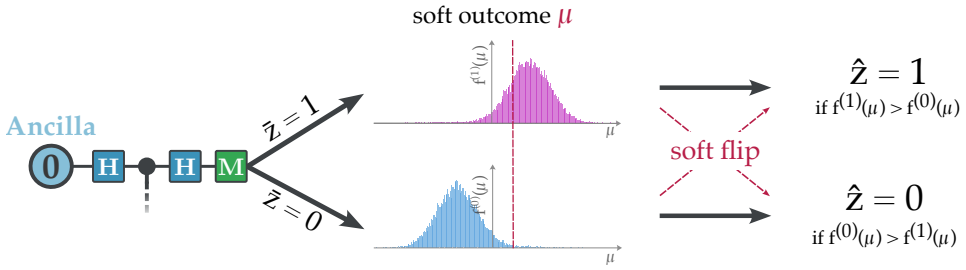


Figure 3.2: Illustration of the measurement process model. The qubit state gets projected onto the eigenspace of the true state $\bar{z}$, leading to a soft outcome μ according to the probability distribution $f^{\bar{z}}(\mu)$. We then obtain the estimated state $\hat{z}$ through a maximum likelihood assignment.

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

$$\mathbb{P}[\text{soft} \mid \mu] = \frac{\mathbb{P}[\text{soft}, \mu]}{\mathbb{P}[\mu]}, \tag{3.2}$$

where

$$\mathbb{P}[\text{soft}, \mu] = \begin{cases} f^{(1)}(\mu) \cdot \mathbb{P}[\bar{z} = 1] & \text{if } f^{(0)}(\mu) \geq f^{(1)}(\mu) \\ f^{(0)}(\mu) \cdot \mathbb{P}[\bar{z} = 0] & \text{otherwise,} \end{cases} \tag{3.3}$$

and

$$\mathbb{P}[\mu] = f^{(0)}(\mu) \cdot \mathbb{P}[\bar{z} = 0] + f^{(1)}(\mu) \cdot \mathbb{P}[\bar{z} = 1]. \tag{3.4}$$

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

$$\mathbb{P}[\text{soft} \mid \mu] = [1 + \frac{\mathbb{P}[\bar{z} = \hat{z}]}{\mathbb{P}[\bar{z} = \hat{z} \oplus 1]} \frac{f^{(\hat{z})}(\mu)}{f^{(\hat{z} \oplus 1)}(\mu)}]^{-1}. \tag{3.5}$$

This model allows us to describe the soft flips device independently, only requiring the distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$ to be known. An illustration of the measurement process model is shown in Figure 3.2.

In the subsequent discussions, we will assume that the priors are equal, $\mathbb{P}[\bar{z} = 0] = \mathbb{P}[\bar{z} = 1] = 0.5$, hence leading to

$$\mathbb{P}[\text{soft} \mid \mu] = [1 + \frac{f^{(\hat{z})}(\mu)}{f^{(\hat{z} \oplus 1)}(\mu)}]^{-1}. \tag{3.6}$$

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, as discussed in Section 2.3.2, could improve the model's accuracy and, con-

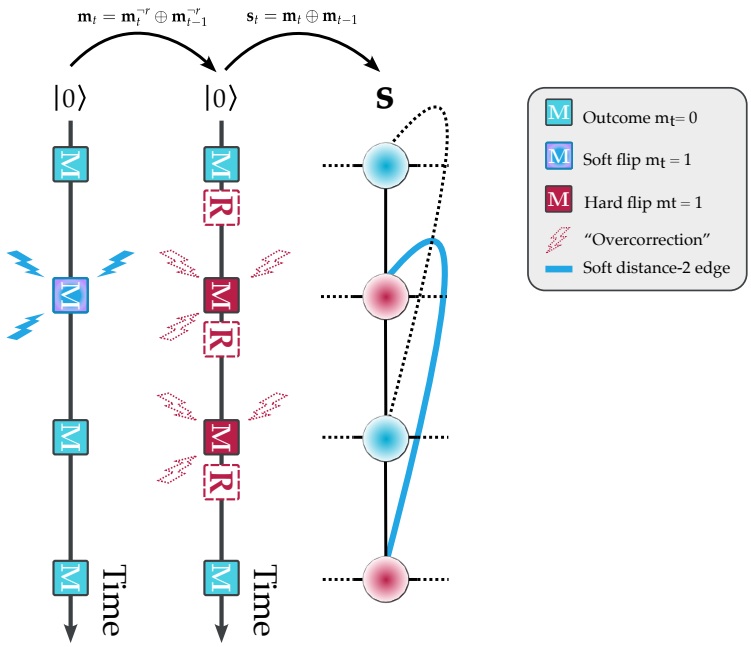


Figure 3.3: Timeline of measurement outcomes without resets, the inferred corrected outcomes, and its corresponding single stabilizer column of the decoding graph G_T. A soft flip occurs at time t=2. Due to overcorrection, the column contains a pair of flipped syndromes separated by two time steps.

sequently, the efficacy of the methods discussed in this chapter.

3.2 Decoding with soft information

Having modeled the process that gives rise to soft flips, we now discuss incorporating this information into the decoding process. We will first discuss the implications of soft flips on the decoding graph and then introduce a method to dynamically adjust the weights in the decoding graph based on the soft outcomes.

3.2.1 The soft flip error channel

As discussed in Section 3.1.1, hard flips alter the ancilla qubit's state, leading to a faulty measurement outcome. In contrast, soft flips only change the measurement outcome; therefore, the outcomes disagree with the state of the ancilla. This difference is crucial when we *do not reset* the ancilla qubits. When correcting the measurement outcomes to synthesize "reset outcomes" $\mathbf{m}_t = \mathbf{m}_t^{\neg r} \oplus \mathbf{m}_{t-1}^{\neg r}$, we assume the ancilla qubits are in the state we measured them in. This assumption is valid for hard flips but not soft flips. Consequently, if an outcome gets altered due to a misassignment, we will "overcorrect," leading to a pair of syndromes separated by *two* time steps, as illustrated in 3.3.

Consequently, a *single error channel*—a soft flip—can connect two syndrome nodes separated by two time steps. We represent this error channel in the decoding graph by adding a *distance-two edge*. We will call this distance-two edge the *soft edge*. Moreover, as a soft flip will only flip pairs of syndromes, we can still decode by matching, as discussed in Section 2.2.2. It is important to note that this *weight-two-time error* has implications on the approximation $T \gg d$ to an optimal *infinite memory*. Nonetheless, we see negligible effects on the decoding performance when aligned with the device's noise model.

These weight-two-time errors do not occur for all rounds; a discrepancy arises when considering the last round of stabilizers. As discussed in Section 2.3.1, we artificially compute the final round of stabilizer outcomes from the code measurements. This last round is effectively perfect. Consequently, a soft flip in the previous round of stabilizer measurements would only cause one faulty measurement outcome, not two consecutive ones. Hence, it produces a pair of flipped syndromes separated by a single time step. A hard flip has the same effect. Consequently, soft and hard flips are two independent, indistinguishable error channels for the last round. Their probabilities combine to give the total probability of a measurement error in the last round as

$$\mathbb{P}_E[\mathtt{E_m}] = p_h(1-p_s) + (1-p_h)p_s, \tag{3.7}$$

with $p_h = \mathbb{P}_E[\mathtt{E_h}]$ and $p_s = \mathbb{P}_E[\mathtt{E_s}]$ the probabilities of a hard and soft flip, respectively.

On the other hand, when we *do reset* the ancilla qubits, these weight-two-time errors should not appear in theory. However, an active reset with an X gate conditioned on the stabilizer measurement outcome will be faulty when a soft flip occurs. This faulty reset would then have the same effect as a faulty artificial correction $\mathbf{m}_t = \mathbf{m}_t^{\neg r} \oplus \mathbf{m}_{t-1}^{\neg r}$ from the previous case. One could circumvent this issue using a *second independent measurement* for the active resets. A weight-two error would only appear when the *two* measurements have a soft flip, a higher-order error process we neglect. Furthermore, a soft flip in the first measurement (the stabilizer measurement) would be corrected by the second measurement, thus only giving a single faulty outcome. Hence, having an independent second measurement for the resets would effectively remove the weight-two-time errors, and soft and hard flips would contribute to the same weight-one error channel with total probability $\mathbb{P}_E[\mathtt{E_m}] = p_h(1-p_s) + (1-p_h)p_s$. Nonetheless, the additional measurement would lead to added idling times and errors.

3.2.2 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. As we proceed, this data, which captures subtle variations beyond the binary outputs, will be referred to as *soft information*.

A simple approach to improve decoding with soft information consists of first using calibration data to estimate the distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$. We then utilize these distributions to estimate the mean probability of a soft flip according to Equation 3.3 as

$$\mathbb{P}[\text{soft}] = \int \mathbb{P}[\text{soft}, \mu] \, \mathrm{d}\mu \tag{3.8}$$

$$= \sum_{z=0}^{1} \int_{f^{(z)}(\mu) < f^{(z\oplus 1)}(\mu)} f^{(z)}(\mu) \, \mathrm{d}\mu. \tag{3.9}$$

We can then augment the decoding graph with soft edges weighted by $\mathbb{P}[\text{soft}]$. This method enables a better-fitted noise model, enhancing the decoding performance.

However, we can approach utilizing soft information from an alternative angle and *dynamically adjust the weights* in the decoding graph. Instead of utilizing the estimated states of the stabilizer measurements $\{\hat{\mathbf{z}}_\mathbf{t}\}_{t=1,\ldots,T}$ as our input for the decoder, we retrieve the soft outcomes $\{\boldsymbol{\mu}_t\}_{t=1,\ldots,T}$ before classification. Employing these soft outcomes alongside estimations of the probability distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$, we compute the probability of a soft flip $\mathbb{P}[\text{soft} \mid \mu]$ for each measurement outcome μ, as per Equation 3.6. This calculation enables us to dynamically adjust the weights in the decoding graph based on the measurement outcome's ambiguity, thus refining our noise model and decoding graph on a *per-shot basis*. The full 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 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 graph-like structures with soft information.

3.3 Experimental procedure for soft information decoding

The following section will discuss the general experimental procedure we used to investigate the benefits of soft information in quantum error correction. This procedure is code and platform-independent.

Algorithm 1 Dynamic Reweighting of Soft Edges in the Decoding Graph

Input:

- A weighted decoding graph G_T for T rounds of measurements, with stabilizer nodes $\{n_{a,t} | a \in \{1, \ldots, m\}, t \in \{1, \ldots, T\}\}$.
- Soft stabilizer measurement outcomes $\{\boldsymbol{\mu}_t\}_{t=1,\ldots,T}$.
- Soft code measurement outcomes $\mu^{c_1}, \ldots, \mu^{c_n}$.
- Probability distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$.

Output:

- A reweighted decoding graph $\tilde{G}_T$.

for each $t = 1$ **to** T **do**
  for each stabilizer $a = 1$ **to** m **do**
    Retrieve the soft outcome $\mu_{a,t}$ from $\boldsymbol{\mu}_t$.
    Compute the soft flip probability $p_s(\mu_{a,t})$ using Equation 3.6.
    if $t \leq T - 1$ **then**
      Calculate the edge weight $w = -\log(p_s/(1 - p_s))$.
      Reweight the distance-two edge $(n_{a,t}, n_{a,t+2})$.
    end if
    if $t = T$ **then**
      Compute the total measurement error probability using Equation 3.7.
      Calculate the edge weight $w = -\log(p_{\text{tot}}/(1 - p_{\text{tot}}))$.
      Reweight the distance-one edge $(n_{a,t}, n_{a,t+1})$.
    end if
  end for
end for
for each code outcome μ^{c_i} **do**
  Identify the error channel involving code qubit c_i.
  Compute the soft flip probability $p_s(\mu^{c_i})$ using Equation 3.6.
  Compute the total error probability of that channel using Equation 2.25.
  Calculate the edge weight $w = -\log(p_{\text{tot}}/(1 - p_{\text{tot}}))$.
  Reweight the edge corresponding to the identified error channel.
end for

3.3.1 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 Equations 2.23 and 2.24 to estimate the idling error probabilities. Specifically, we set the idling time 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.

We run a set of calibration circuits before each experiment to estimate the probabilities for the measurement-related processes. These circuits consist of preparing

Figure 3.4: Double measurement calibration circuit for estimating the probabilities of soft and hard flips in the first measurement.

all the device qubits in the $|0\rangle$ or $|1\rangle$ state and measuring them twice in the Z basis, as depicted in Figure 3.4. 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 3.1. We assume that the reset error rate and the single X gate error rate are negligible compared to the measurement error rates.

Outcomes	Smallest order error	Second smallest order error
11	$\varnothing^1\varnothing^2 = \mathcal{O}(p^0)$	$\mathrm{E}_\mathrm{s}^1\mathrm{E}_\mathrm{s}^2 = \mathcal{O}(p^2)$
00	$\mathrm{E}_\mathrm{h}^1\varnothing^2 = \mathcal{O}(p^1)$	$\mathrm{E}_\mathrm{s}^1\mathrm{E}_\mathrm{h}^2 + \mathrm{E}_\mathrm{s}^1\mathrm{E}_\mathrm{s}^2 = \mathcal{O}(p^2)$
10	$\varnothing^1\mathrm{E}_\mathrm{h}^2 + \varnothing^1\mathrm{E}_\mathrm{s}^2 = \mathcal{O}(p^1)$	$\mathrm{E}_\mathrm{h}^1\mathrm{E}_\mathrm{s}^1 + \mathrm{E}_\mathrm{h}^1\mathrm{E}_\mathrm{s}^2 = \mathcal{O}(p^2)$
01	$\mathrm{E}_\mathrm{s}^1\varnothing^2 = \mathcal{O}(p^1)$	$\mathrm{E}_\mathrm{h}^1\mathrm{E}_\mathrm{h}^2 + \mathrm{E}_\mathrm{h}^1\mathrm{E}_\mathrm{s}^2 = \mathcal{O}(p^2)$

Table 3.1: Summary of the possible outcomes and contributing error channels when preparing a $|1\rangle$ state and measuring it twice. $\varnothing^m$, E_h^m, E_s^m denotes the absence of an error, a hard flip, and a soft flip on measurement m, respectively.

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 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

$$p_s = \frac{N_{01}}{N_{\text{tot}}}, \quad p_h = \frac{N_{00}}{N_{\text{tot}}}, \tag{3.10}$$

with $N_{\mathtt{ij}}$ the number of times the outcomes $\mathtt{ij}$ 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

$$p_s = \frac{N_{10}}{N_{\text{tot}}}, \quad p_h = \frac{N_{11}}{N_{\text{tot}}}. \tag{3.11}$$

These probabilities represent the *mean probabilities* of soft and hard flips and can be used for the *naive approach* discussed in Section 3.2.

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.

3.3.2 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 density estimation is a non-parametric 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 Figure 3.5, the distributions of the soft outcomes comprise complex shapes that could lead to inaccuracies when using Gaussian mixture models. Nonetheless, we will compare the performance of kernel density estimation with a simple Gaussian model in the experiments, as discussed in Section 3.3.3.

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 [35]. 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 Figure 3.4. We use the second measurement outcome to filter out data points where a hard flip occurred, as described in Table 3.1. 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

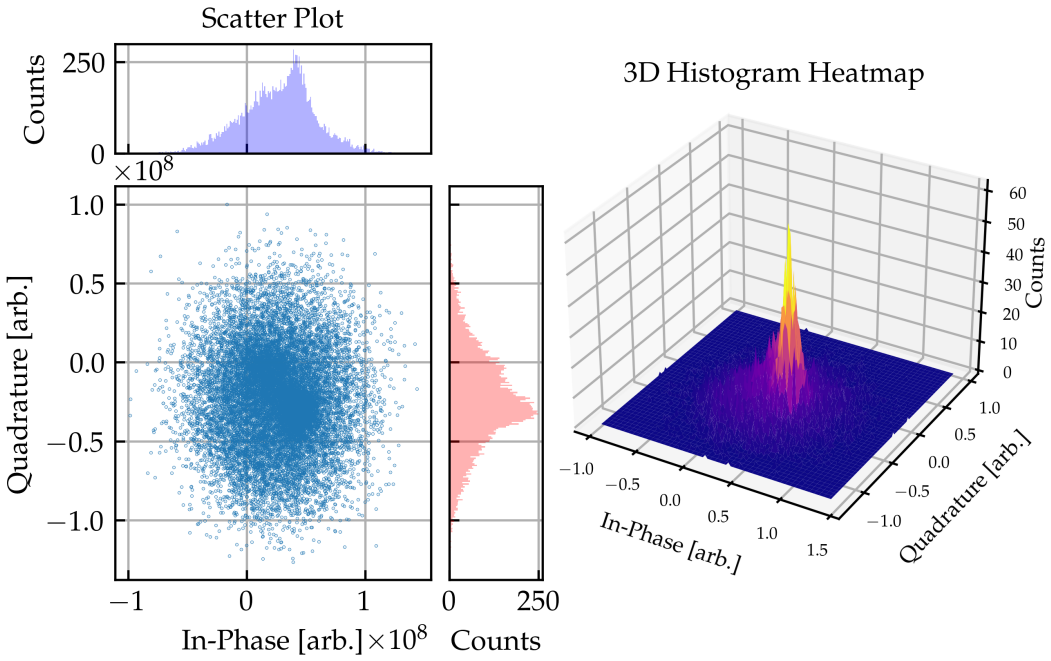


Figure 3.5: IQ data for a particularly bad-mannered qubit (106) of the `IBM Sherbrooke` device prepared in the $|1\rangle$ state. The data shows a complex behavior that is not easily modeled by a simple Gaussian distribution.

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 Figure 3.4 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 $\boldsymbol{\mu}^{\bar{z}}$ and fit the distribution $f^{(\bar{z})}(\mu)$ to these points using kernel density estimation.

3.3.3 One-dimensional Gaussian model of $f^{(\bar{z})}(\mu)$

As an alternative to the general, non-parametric kernel density estimation, we can model the distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$ using a simple Gaussian model. To do so, we make multiple assumptions. First and foremost, we assume that the distributions of each state are monomodal 2D Gaussians. Moreover, readout calibration on IBM devices ensures these distributions have identical variances and align along the I axis, which enables a linear *decision boundary* to simplify classification. We further assume that the I and Q components are independent, reflected in a diagonal covariance matrix. Consequently, we can project the problem onto the I axis and model the distributions as 1D Gaussians.

As we assume the distributions to be monomodal and we need to filter out

hard flips, we use the combined soft measurement data from both prepared states $\bar{z} = |0\rangle, |1\rangle$ to fit a bimodal Gaussian distribution with tied covariance, implying identical variance across the modes. The two distributions $f^{(0)}(\mu)$ and $f^{(1)}(\mu)$ then correspond to the two modes of the Gaussian mixture model. We use these distributions' means ν_0, ν_1, and standard deviation σ to estimate the probability of a soft flip. According to Equation 3.6, the soft edge weight is defined as

$$w(\mu) = -\log\left(\frac{p_s(\mu)}{1-p_s(\mu)}\right) = -\log\left(\frac{f^{(\hat{z})}(\mu)}{f^{(\hat{z}\oplus 1)}(\mu)}\right). \tag{3.12}$$

Using $f^{(\hat{z})}(\mu) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\mu_I - \nu_{\hat{z}})^2}{2\sigma^2}}$, with μ_I the in-phase component of the soft outcome, we can then write the weight as

$$w(\mu) = \frac{\Delta\nu^2 - 2\mu_I \Delta\nu}{2\sigma^2}. \tag{3.13}$$

with $\Delta\nu^2 = \nu_{\hat{z}}^2 - \nu_{\hat{z}\oplus 1}^2$ and $\Delta\nu = \nu_{\hat{z}} - \nu_{\hat{z}\oplus 1}$. Shifting the IQ plane to center the midpoint between the Gaussians at $I = 0$ and defining $D = |\Delta\nu|$ as the mean's separation, we simplify the weight to

$$w(\mu) = \frac{D}{\sigma^2}|\mu_I'|. \tag{3.14}$$

This formula shows that the weight, effectively the inverse of ambiguity, is directly proportional to the distance of the IQ point to the midpoint and the ratio of the separation D to the variance σ^2. This relationship emphasizes how ambiguity is linearly modulated by the measurement's proximity to the decision boundary. Moreover, it also reflects that the soft flip probability decreases with greater Gaussian separation and increases with broader distribution widths.

3.3.4 Comparisons of decoding strategies

To assess the benefits of using soft information in quantum error correction, we conduct experiments without resetting the ancilla qubits and compare *four different decoding strategies*:

- **Naive Hard Decoding:** We entirely disregard the soft flip channel and decode on a graph lacking distance-2 soft edges. We only use the readout error rate given by calibrations from IBM devices, which also covers gate and idling errors.

- **Calibrated Hard Decoding:** We incorporate the mean probabilities of soft and hard flips p_h, p_s estimated from double measurement calibrations before

each experiment. We adjust the weights of the decoding graph once before decoding commences and disregard the soft outcomes.

- **Dynamic Soft Decoding:** We also begin with double measurement calibrations to assess the mean probability of a hard flip. Using Algorithm 1, we dynamically reweight the distance-2 soft edges on a shot-per-shot basis according to the soft outcomes.

- **Data-Informed Hard Decoding:** Similar to Calibrated Hard Decoding, this method employs a non-dynamic 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 ambiguities. Doing so confirms that shot-per-shot adjustments to the weights genuinely enhance the decoding process.

The initial decoding method, Naive Hard Decoding, is fundamentally limited as it disregards the soft flip error channel. This oversight makes it an unsuitable metric for evaluating the advantages of incorporating soft information into the decoding process. Essentially, this method serves only demonstrative purposes.

A more effective approach to assess the benefits of utilizing soft information involves including the distance-two soft edges and weighting them based on the *pairwise correlation method* [32, 36, 37]. This technique adjusts edge weights according to the observed frequency of non-trivial syndrome nodes. However, this method has several drawbacks. It requires running the quantum code multiple times to gather sufficient shot statistics. This approach presupposes access to experimental data before effective decoding can commence, which might inadvertently bias the results. Additionally, this heuristic approach demands significant data to provide reliable estimates, complicating its practical implementation.

A recent study by Ali *et al.* [18] introduced using a simple 1D Gaussian model—discussed in Section 3.3.3—and pairwise correlation to improve decoding on a repetition code using soft information. This timely release closely aligns with our own efforts. Prompted by this coincidence, we will further investigate the results of applying the 1D Gaussian model and directly compare its effectiveness to kernel density estimation in the next chapter.

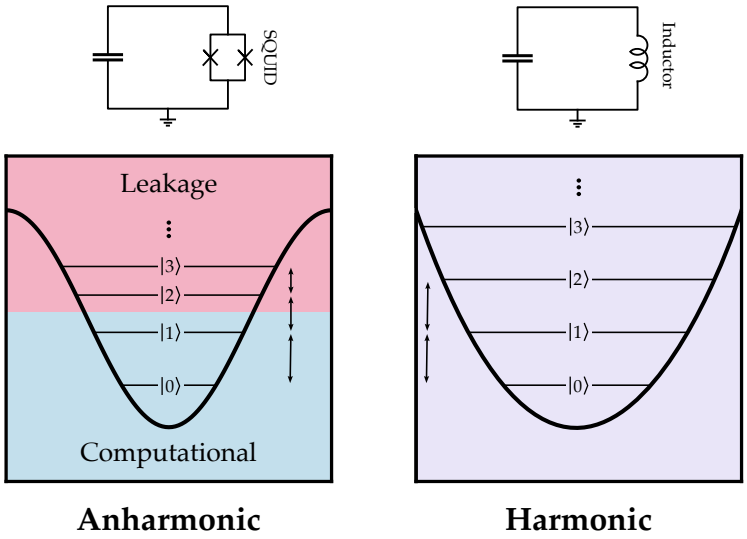


Figure 3.6: Comparative illustration of a transmon qubit circuit (left) and an LC resonator circuit (right). The transmon qubit's potential is anharmonic, leading to a computational subspace (blue) and a higher energy subspace (pink) with different energy transitions. Nonetheless, this anharmonicity is finite, resulting in transitions into higher energy states during computation, also called leakage. In comparison, the potential of the LC circuit is harmonic and has, thus, evenly spaced energy levels. The transmon energy level diagram is adapted from [19].

3.3.5 Soft decoding in the presence of leakage

Transmon qubits, featured in IBM's devices, are a leading platform for quantum computation. However, transmons have a finite anharmonicity, as illustrated in Figure 3.6, allowing transitions to higher energy states. Such transitions can excite a qubit into a higher energy state, resulting in *leakage errors*. These excitations often occur when implementing two-qubit gates [38, 39], as well as measurements [40, 41, 42, 43].

Leakage poses a significant challenge in quantum error correction. Once a qubit leaks, it does not operate as intended, leading to persistent faults over multiple stabilizer measurements. Quantum error correction typically assumes that errors are uncorrelated; however, leakage undermines this assumption, posing a significant challenge for the experimental implementation of quantum error correction protocols. Miao *et al.* demonstrated the detrimental effect of leakage on decoding performance in superconducting qubits [19], showing a much higher sensitivity of code performance to leakage than Pauli errors. Past experiments involving multiple rounds of quantum error correction have focused on identifying and post-selecting leaked qubits, as seen in the studies conducted by [32, 9].

Soft information decoding experiments have also selectively removed samples with detected leaked qubits [18, 37]. Notably, these studies involved no more than $T = 16$ rounds of measurements. In contrast, our experiments extend to $T = 100$

rounds, significantly increasing the likelihood of leakage and thus reducing the yield to nearly zero if post-selection were applied. We will expand on this issue in the next chapter, where we will quantify the amount of leakage in our experiments.

Several strategies have been developed to mitigate leakage in quantum systems, such as implementing leakage-reducing reset protocols [32, 44], applying optimal control techniques to gate operations [45], and integrating leakage reduction units directly into the circuit design [46, 47]. Despite the availability of these methods, our experimental setup lacked these advanced features. Furthermore, a study by Fowler *et al.* [47] demonstrated that a threshold still exists even under leakage, albeit significantly lower. Consequently, we opted to proceed with our experiments without directly addressing leakage at the hardware level. Instead, we focused on managing leakage through classical decoding techniques.

Unexpectedly, our experiments discovered a heuristic method that enhances decoding performance when paired with soft information decoding in the presence of leakage. We observed that treating the IQ points identified as originating from leaked states as *maximally ambiguous* drastically improved our decoding outcomes.

We hypothesize that retrieving a measurement outcome from a leaked state can still contain partial information about the state of adjacent qubits. Treating these leaked measurements as maximally ambiguous allows us to incorporate this information into the decoding graph while allowing the matching algorithm to flexibly consider or disregard these outcomes. This flexibility arises from the cost of matching the corresponding distance-two time edge to the subsequent measurement, which is effectively zero as the outcome is maximally uncertain.

We also experimented with two different decoding methods, both of which performed poorly. The first method involved designating the node corresponding to a leaked state as a boundary node, allowing it to match independently of its syndrome. The second method entailed setting the weight of both the distance-one and distance-two time edge to zero, effectively ignoring the outcome and allowing arbitrary matching. Unfortunately, both methods led to a notable decline in decoding performance, the reasons for which remain unclear. Consequently, this thesis will only discuss the results obtained by treating the outcomes as maximally ambiguous—meaning that the corresponding soft edges have weight zero—though further research is necessary to delve into the underlying mechanisms.

To determine whether an IQ point is leaked, we repurposed calibration data that was not originally intended for leakage detection. Retrieving the calibration data involves a simple circuit with just state preparation and subsequent readout. Ideally, this circuit should not exhibit leakage, although this assumption is only approximate for our experiments. For each IQ point μ, we assess it against the calibration data to determine whether it qualifies as an *outlier*. We define an outlier as a point with a sampling probability from $f^{(z)}(\mu)$ below a specified threshold for

both states $z = 0, 1$. We have empirically set this threshold at 1% to optimize detection efficacy. We consider such points leaked states and assign them a soft flip probability p_s of 0.5. We display the effects of this method on our data in Figure 3.7.

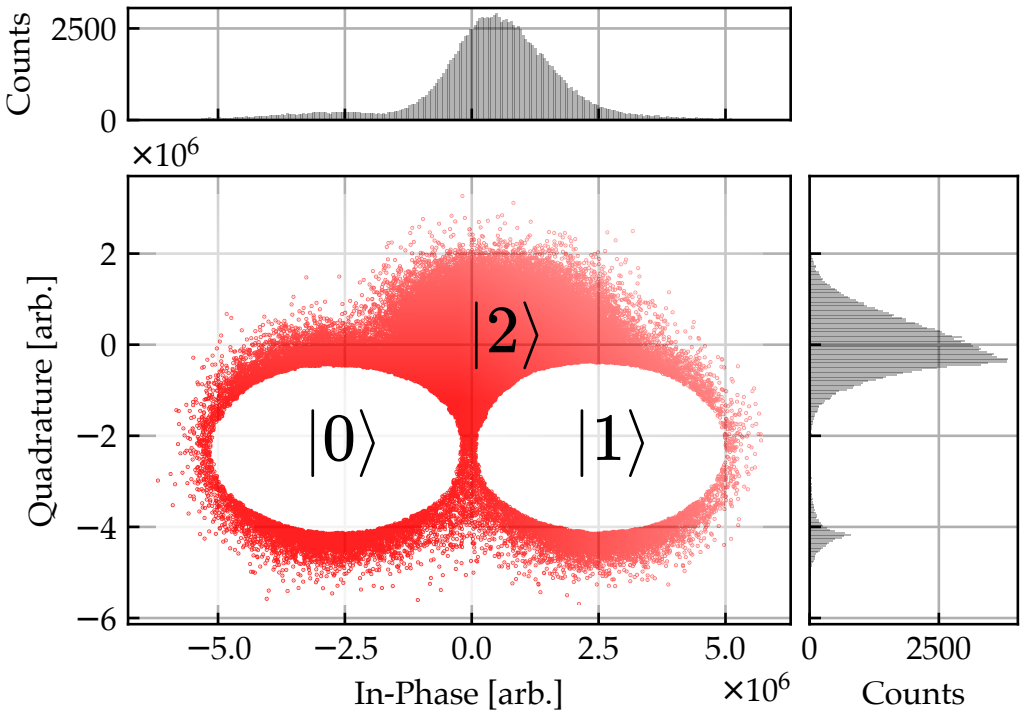


Figure 3.7: Vizualization of outliers in the IQ plane according to our filtering method for qubit 3 of `IBM Sherbrooke`. Red points indicate outliers, which are classified as leaked states. They are determined based on their sampling probabilities from the distribution $f^{(z)}(\mu)$, with points falling below the 1% threshold considered as potential leaked states.

This method encounters classification errors in identifying leakage for two primary reasons. Firstly, the calibration data, comprising only 15,000 shots[1], does not adequately cover the IQ plane. This limitation is problematic because our employed Epanechnikov kernel is local, leading to potential misclassifications. For instance, if an IQ point from a "good" computational state lies a few standard deviations away from the mean, it might be erroneously classified as an outlier and, thus, as leakage. Secondly, as described in [37] and depicted in Figure 3.8, the IQ distribution of the leaked state forms a Gaussian with a variance similar to the other two Gaussians. Thus, ideally, the decision boundary between the leaked and computational states should be linear. However, focusing solely on outliers results in a decision boundary that matches the variance of the two Gaussians, which can miss

[1]This number originates in a hardware limitation.

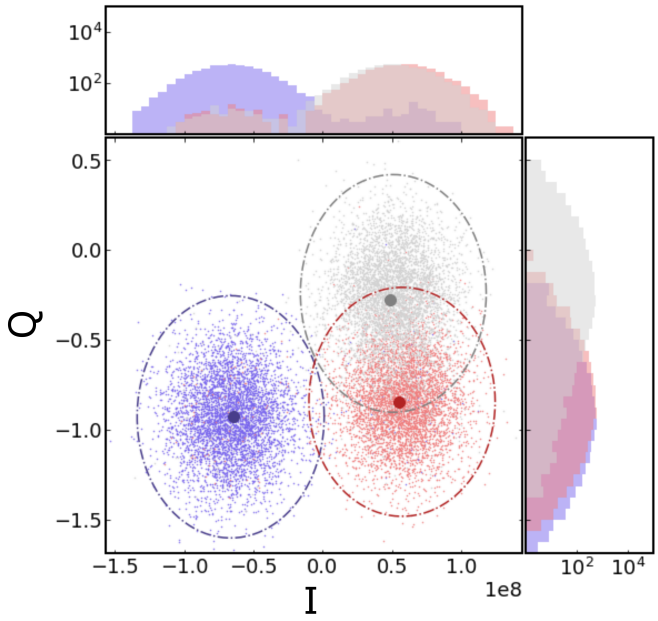


Figure 3.8: IQ data for qubit QF11 from `IBM Peekskill` when prepared in the $|0\rangle$, $|1\rangle$, and $|2\rangle$ states. The data is fitted with a Gaussian mixture model, showing that the leaked IQ points form a Gaussian with a variance similar to the other two Gaussians. Figure taken from [37].

some leaked states, as shown in Figure 3.7. As a result, our method may misclassify points that are not leaked as leaked and vice versa. We attempt to balance these errors by setting the threshold at 1%.

Despite these imperfections—stemming from the method's original unintended use for this application—it still significantly enhances decoding performance, as we will demonstrate in subsequent chapters. We believe that adopting specialized calibration specifically for leakage, as suggested in [37], and integrating it with our current method could substantially improve decoding accuracy further. However, our current approach to handling leakage lacks a robust theoretical basis, and the information indicating a state leaked might be more effectively utilized in a specialized decoder than by simply assigning a soft flip probability of 0.5.

It's also crucial to note that a 1D Gaussian model fails to classify leakage effectively because the quadratures of leakage points typically fall between the means of the two other Gaussians, as seen in Figure 3.7. Consequently, the 1D Gaussian model cannot differentiate between leaked and computational states.

Chapter 4

Soft decoding repetition codes on superconducting hardware

4.1 Essentials of repetition codes

In classical error correction, one effective method for addressing errors is replicating the data multiple times. For example, to transmit a bit string b of length n, we can repeat the string d times, yielding a new string b' of length $n \cdot d$. If we transmit b' and the receiver receives a noisy version of b', he can correct errors through majority voting.

This concept extends to quantum error correction, although quantum mechanics requires modifications due to the impossibility of directly cloning quantum states. We utilize what is known as the d-dimensional quantum repetition code, where d represents the number of repetitions. A generic quantum state $|\psi\rangle = \alpha\,|+z\rangle + \beta\,|-z\rangle$ is encoded as

$$|\psi\rangle \to |\psi\rangle_L = \alpha\,|+z\rangle^{\otimes d} + \beta\,|-z\rangle^{\otimes d} \tag{4.1}$$

$$= \alpha|00\ldots0\rangle + \beta|11\ldots1\rangle. \tag{4.2}$$

This encoding method complies with the no-cloning theorem, and we can achieve it using a circuit depicted in Figure 4.1.

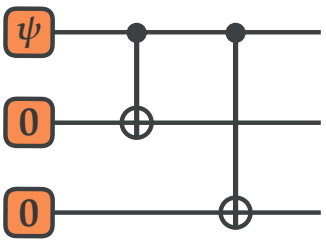


Figure 4.1: Circuit for encoding a quantum state using a repetition code.

The repetition code effectively guards against bit-flip (X) errors. However, it does not protect against all types of errors. For instance, a phase-flip (Z) error on any qubit i in the encoded state $|\psi\rangle_L$ alters the state to

$$Z_i |\psi\rangle_L = \alpha|11\ldots1\rangle - \beta|00\ldots0\rangle, \tag{4.3}$$

resulting in a decoded state of $|\psi\rangle = \alpha|+z\rangle - \beta|-z\rangle$ instead of $|\psi\rangle = \alpha|+z\rangle + \beta|-z\rangle$. Conversely, encoding the state in the X-basis

$$|\psi\rangle \to |\psi\rangle_L = \alpha|+x\rangle^{\otimes d} + \beta|-x\rangle^{\otimes d} \tag{4.4}$$

$$= \alpha|++\ldots+\rangle + \beta|--\ldots-\rangle, \tag{4.5}$$

protects against phase-flip errors but leaves the state vulnerable to bit-flip errors. In summary, repetition codes only shield against one type of error at a time, making them partial solutions that function more as a classical code adapted for quantum computing rather than a complete quantum code.

4.1.1 Stabilizer formalism

As discussed in Section 2.1.2, we can employ the stabilizer formalism to describe repetition codes. Considering the encoding in Equation 4.1, the stabilizer generators that leave the codeword invariant are Z Paulis acting on adjacent qubits

$$S = \langle Z_1Z_2, Z_2Z_3, \ldots, Z_{d-1}Z_d\rangle. \tag{4.6}$$

Consequently, the measurement of stabilizers results in evaluating the parity of neighboring qubits. If the parity is odd, an error occured on one of the qubits.

Since we encode a single logical qubit, we have two logical generators: X_L and Z_L. The logical operator X_L, which produces a logical X on the logical qubit and transitions $|+z\rangle_L = |+z\rangle^{\otimes d}$ to $|-z\rangle_L = |-z\rangle^{\otimes d}$, commutes with the stabilizers and is given by

$$X_L = X^{\otimes d}. \tag{4.7}$$

Conversely, the logical operator Z_L transitions $|+x\rangle = |+z\rangle^{\otimes d} + |-z\rangle^{\otimes d}$ to $|-x\rangle = |+z\rangle^{\otimes d} - |-z\rangle^{\otimes d}$ and is given by

$$Z_L = Z_i, \tag{4.8}$$

where i is any single qubit in the encoded state. We will discuss this flexibility in choosing qubit i to represent the logical Z information of the encoded state in Section 4.1.2.

The code distance in the repetition code is d when considering the X errors and

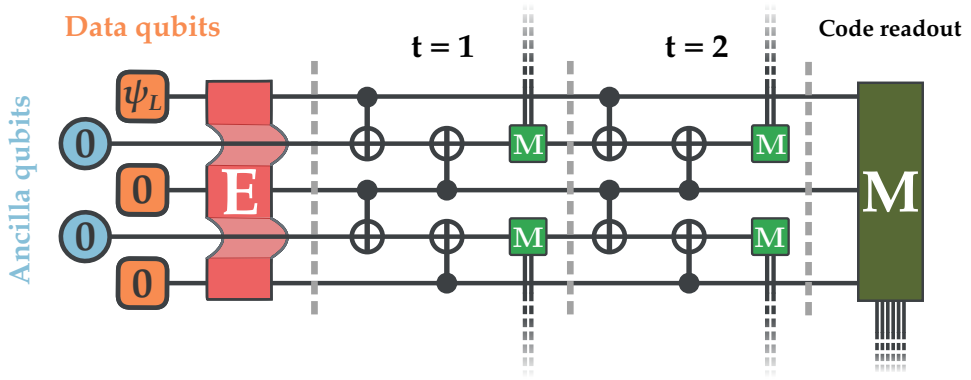


Figure 4.2: Quantum circuit to implement a repetition code with two rounds of stabilizer measurements on $n = 3$ data qubits (orange) and $n = 2$ ancilla qubits (blue). The encoding block (E) entangles the logical information in the X or Z basis. The final measurement block (dark green) measures the code qubits in the same basis as the encoding.

merely 1 with Z errors. This discrepancy reflects that the repetition code can only correct one type of error at a time. Despite the limitation of correcting only one kind of error at a time, repetition codes serve as a valuable tool for conceptual demonstrations. They offer a code distance that increases linearly with the number of qubits, making them suitable for high-distance experiments on the intermediate-scale quantum devices that we have today.

Furthermore, repetition codes are almost cyclic, enabling their decoding through the MWPM algorithm, as explored in Section 2.2.1. This decoding method is also extensively used for surface codes, widely recognized for their promising potential in fault-tolerant quantum error correction. Therefore, improved decoding performance achieved with repetition codes likely has implications for the broader application with surface codes.

4.1.2 Repetition codes on superconducting hardware

To implement repetition codes on superconducting hardware, we configure a 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_iZ_{i+1} stabilizers using the method illustrated in Figure 2.4, which results in CNOT gates between neighboring qubits for Z_iZ_{i+1} stabilizers. The quantum circuit for a repetition code for two rounds of stabilizer measurement on $n = 3$ data qubits is depicted in Figure 4.2.

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,

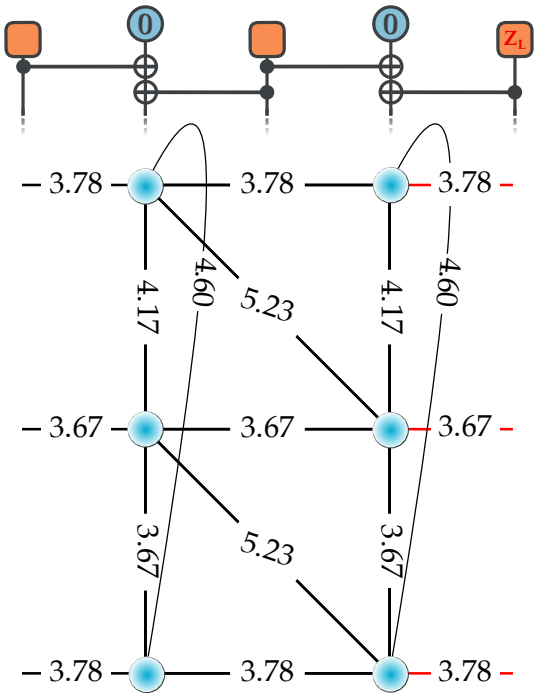


Figure 4.3: Decoding graph for a repetition code of distance $d = 3$ with $T = 2$ stabilizer round, including the final code qubit readout. The graph represents the syndrome nodes (blue circles) connected by code qubit and measurement error edges. The weights of the edges are typical for the `IBM Sherbrooke` noise model. The red edges represent the qubit that carries the logical information.

while the soft flip channel creates a distance-two-time edge. The stabilizer circuit illustrated in Figure 4.2 also aligns with the example from Section 2.3.2, and an additional error channel links the diagonal syndromes. 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 Figure 4.3.

We implement various configurations of the repetition code, encoding both in the X and Z bases and varying the prepared logical state between $|+z\rangle_L$ and $|-z\rangle_L$. We encode the logical state, 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, as described in Section 2.3.2, and we utilize Pauli tracing via the Stim software to weight the edges of the decoding graph [48].

We decode the repetition codes using the MWPM algorithm via the PyMatching software [49]. This algorithm determines the most likely error (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,

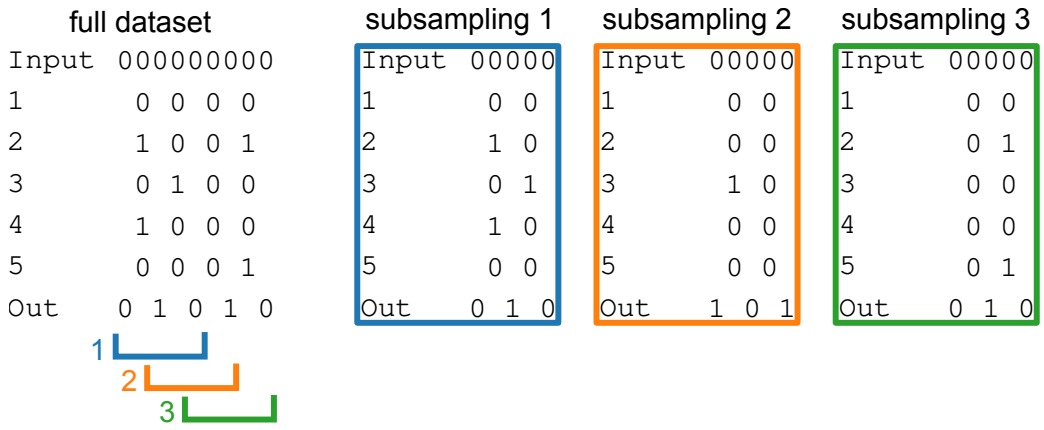


Figure 4.4: Example of subsampling for a repetition code of distance $d_s = 3$ from a $d = 5$ code. The subsampling method allows us to evaluate the performance of repetition codes across multiple distances without the need to individually execute each configuration on the hardware. Figure taken from [32].

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* [26, 50, 32] 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 $d = 5$ is shown in Figure 4.4. 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.

4.2 Benefits of soft decoding with repetition codes

To explore the advantages of soft decoding repetition codes on superconducting hardware, we initially conducted simulations that mimic the actual device settings to assess the effectiveness of soft decoding. Following these simulations, we implement soft decoding directly on the hardware and compare the outcomes with those of the simulations.

4.2.1 Simulation results

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 noise model described in Section 2.3.2. 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 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 re-estimate the measurement outcomes $\hat{z}(\mu)$ based on Equation 3.1 and calculate the corresponding soft flip probabilities $p_s(\mu)$ according to Equation 3.6. This re-estimation 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 estimation using the MLPack library [51].

Importantly, as discussed in Section 3.3.5, the calibration data does 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.

We investigate the logical error rate over $T = 50$ rounds of stabilizer measurements, simulating a repetition code of distance $d = 51$—the maximal feasible distance for our used hardware[1]. Additionally, we collect results for all odd distances ranging from $d = 3$ to $d = 51$, employing the subsampling method utilized in our hardware experiments. 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*, as outlined in Section 3.3.4. While this setup with T equal to 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. Findings from Chen *et al.* validate this choice, showing that time boundary effects in repetition codes, prevalent for $T \leq 10$ rounds, vanish for higher round numbers, supporting our decision to set $T = 50$ [32].

Furthermore, because we modeled symmetric decoherence channels, the noise model is independent of the prepared logical state. Consequently, we only simulate one of the logical states for each basis.

Vizualization convention

In this chapter, we employ specific visualization conventions to enhance the clarity and readability of our results. We distinguish logical states prepared in the X basis using red colors, and those in the Z basis with blue. Round markers indicate the $|+\rangle$ states, while diagonal markers specify the $|-\rangle$ states. We use hollow markers and dashed lines for statistical data that lacks sufficient significance—perhaps due to a low number of observed events. These exceptions and the type and threshold for the number of events are marked in the legend. Additionally, to represent statistical uncertainty, we often include 68% confidence intervals calculated using the Wilson score interval method [52]. These conventions are illustrated in Figure 4.5.

[1] This is the maximal feasible odd distance that we found while ensuring that no significant "bad" qubit was included in the used line of qubits.

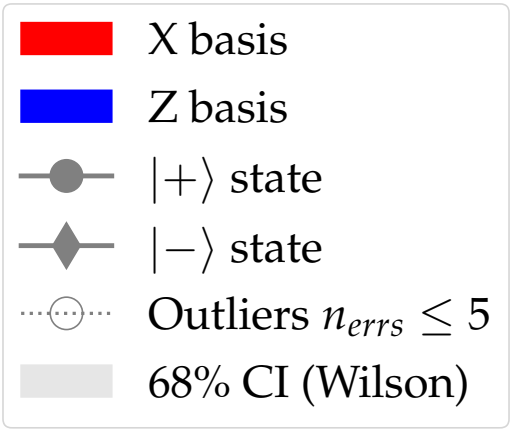


Figure 4.5: Visualization conventions used in this chapter for displaying simulation and hardware results.

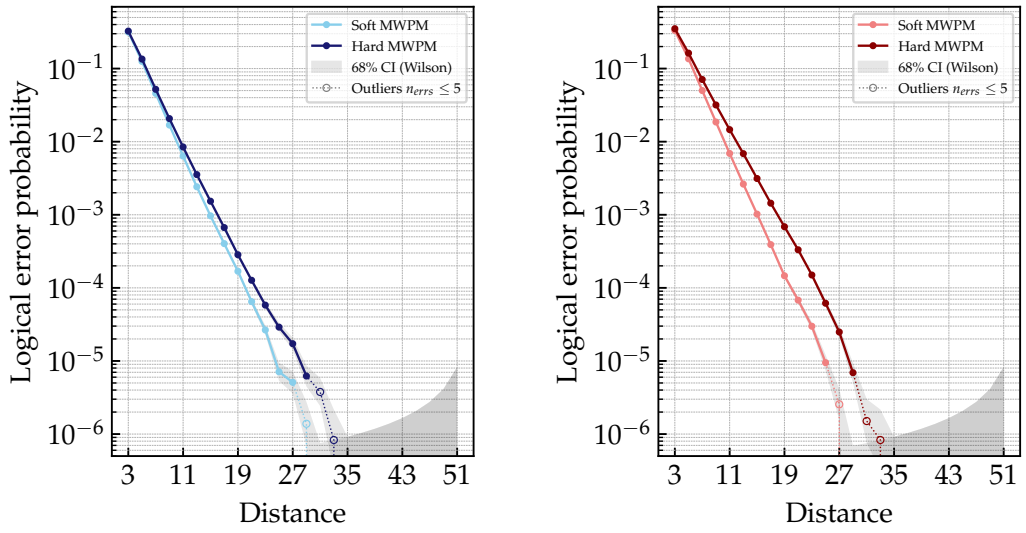
Decoding results

Simulation results for soft decoding of repetition codes are illustrated in Figure 4.6 for the Z and X bases. These plots show the logical error probability on a 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 logical error rates of the soft and hard decoders and include the 68% confidence intervals calculated using the Wilson score interval. 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 the error rates drop below 10^{-7}, necessitating more than tens of millions of shots to record a single logical error event. However, processing even one million shots poses a significant computational challenge.

Both error rates demonstrate a linear decline with increasing code distance on a logarithmic scale, indicating exponential suppression of errors, as detailed in Section 2.3.3. 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 *three* and *fifteen times lower*, respectively. This performance improvement arises from the steeper slope of the soft decoder's curve, indicating a higher threshold.

Referring to Section 2.3.3, to analyze this difference in threshold more thoroughly, we calculate the *logical error per round* ϵ_L and fit the Lambda factor Λ from the curve's slope in logarithmic space. We plot these logical error per round against



(a) Prepared logical state $|\psi\rangle_L = |+z\rangle_L$.

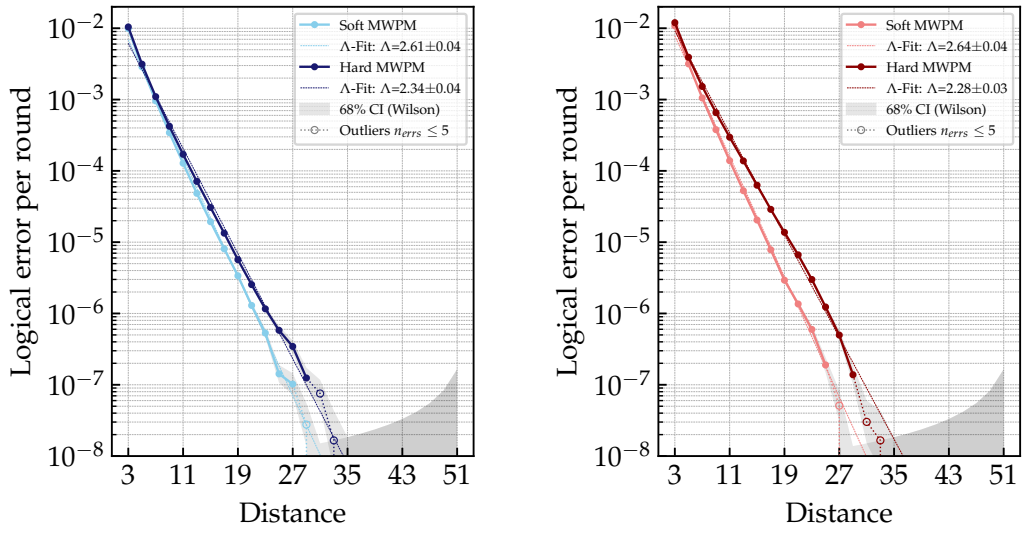
(b) Prepared logical state $|\psi\rangle_L = |+x\rangle_L$.

Figure 4.6: Simulation results of the `IBM Sherbrooke` device, depicting the logical error probabilities as a function of code distance for the soft and hard MPWM decoders for $T = 50$ rounds of stabilizer measurements.

the distance for the Z and X bases, along with the fitted lambda curve, in Figure 4.7. As discussed, due to the small logical error rate P_L, the error rate per round approximates to $\epsilon_L \approx \frac{P_L}{T}$, making the trend of ϵ_L almost identical to that of P_L. Consequently, we will present our results in terms of the logical error per round for its intuitive and straightforward interpretation.

Figure 4.7 demonstrates that the soft decoder's curve has a steeper slope than the hard decoder's, indicating a higher threshold and an exponentially superior performance for the soft decoder. As detailed in Section 2.3.3, 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.

To manage the meager error rates and resolve higher distances with a feasible number of shots, we modify the simulations by doubling the error rates derived from device calibration data. This adjustment increases the occurrence of logical errors, enabling data collection for higher distances within a reasonable shot count. Critically, we only enhance the error rates for gates, idling times, and hard measurement errors; we do not alter the IQ calibration data, so the soft flip probability



(a) Prepared logical state $|\psi\rangle_L = |+z\rangle_L$.

(b) Prepared logical state $|\psi\rangle_L = |+x\rangle_L$.

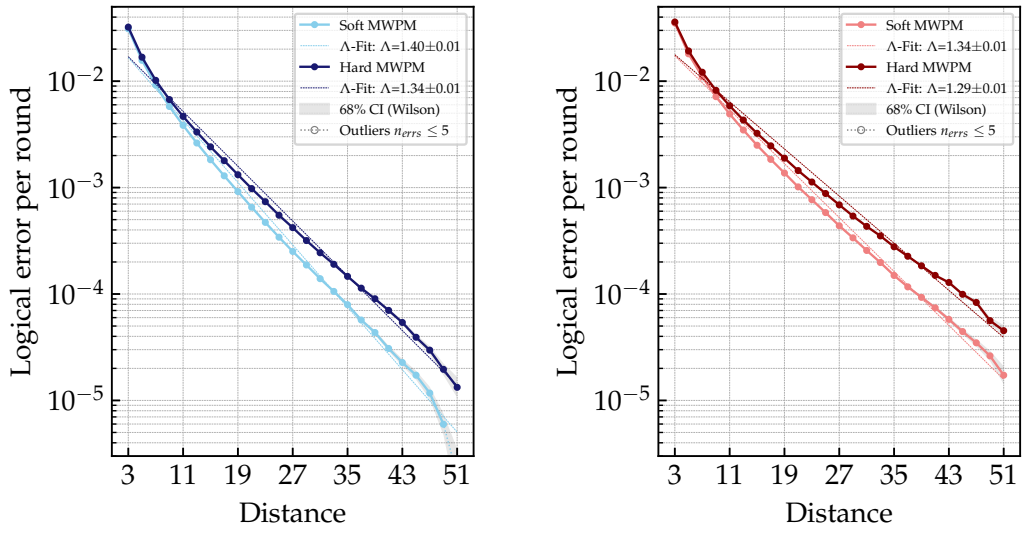
Figure 4.7: Simulation results of the `IBM Shebrooke` device, depicting the logical errors per round as a function of code distance for the soft and hard MPWM decoders for $T = 50$ rounds of stabilizer measurements. The Lambda factors Λ are fitted from the curves' slopes in logarithmic space.

remains unchanged. Figure 4.8 presents the logical error rate per round for these adjusted simulations.

The plots reveal several vital observations. First, the logical error rate per round is exponentially higher when compared with simulations using half the noise model, underscoring the exponential relationship between the logical and physical error rates. Second, there is a pronounced disparity in the error rates between the Z and X bases. This difference, though present in earlier simulations, is more pronounced here. It arises due to the inherent bias in superconducting qubits, which exhibit a higher rate of dephasing errors than bit-flip errors. Our device calibrations verify this bias and show that an idling phase flip error is *three times* more likely than an idling bit-flip error. Thirdly, the curves display slight curvature at the beginning and end, a trend also observed in the hardware data, although it is more pronounced there. We attribute this characteristic to the subsampling method, as it is absent in simulations where we compute the error rates for each distance separately[2].

Lastly, while the soft decoder still shows an exponential advantage over the hard decoder, this advantage is reduced with the threshold increase now at 4.5% for the Z basis and 3.9% for the X basis. This diminished difference can be explained by the fact that we only increased the error rates for gates, idling times, and hard mea-

[2]This observation has been preliminarily verified and requires further analysis.



(a) Prepared logical state $|\psi\rangle_L = |+z\rangle_L$.

(b) Prepared logical state $|\psi\rangle_L = |-z\rangle_L$.

Figure 4.8: Simulation results with doubled error rates for gates, idling times, and hard measurement errors of the `IBM Sherbrooke` device. Depicted are the soft and hard decoders' logical errors per round for $T = 50$ rounds of stabilizer measurements. The Lambda factors Λ are fitted from the curves' slopes in logarithmic space.

surement flips, while the error rates for soft flips remained constant. Consequently, the advantage of enhanced accuracy per shot on soft flip edges is less significant and overshadowed by the higher error rates elsewhere in the system. This finding aligns with research by Pattison *et al.*, who demonstrated that the effectiveness of soft decoding depends on the ratio of soft flip probability to hard flip probability, as shown in Figure 4.9.

4.2.2 Superconducting hardware results

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 (I. Hesner *et al.*, work in progress.), 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.

Benefit of soft decoding on hardware

Figure 4.10 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

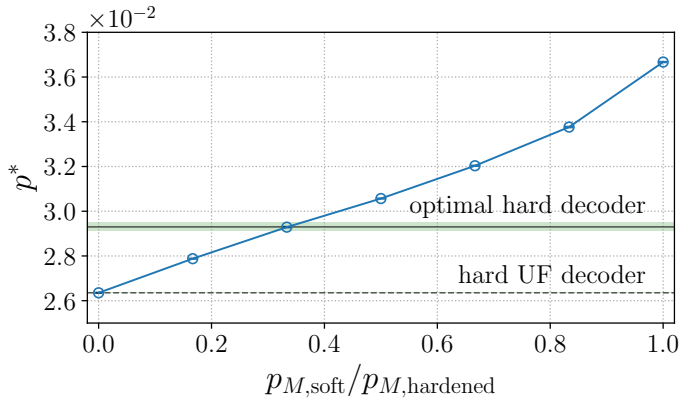
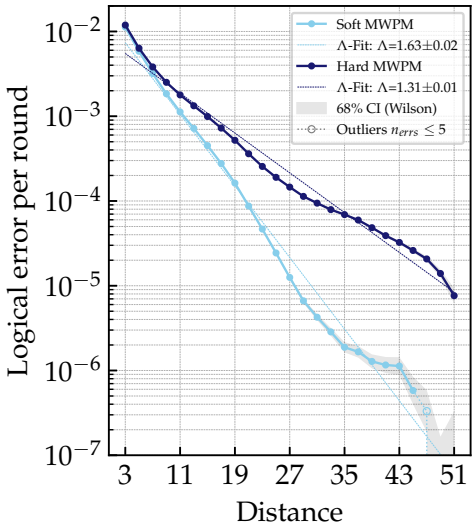


Figure 4.9: Threshold $p*$ obtained with the soft decoder as a function of $\frac{p_s}{p_s+p_h-2p_sp_h}$, where p_s and p_h are the soft and hard flip probabilities, respectively. Also depicted as lines are the thresholds of the hard Union-Find decoder and the theoretically best possible hard decoder. The plot underscores that the benefit of soft decoding is contingent on the ratio of soft flip to other error probabilities. Figure taken from [1].

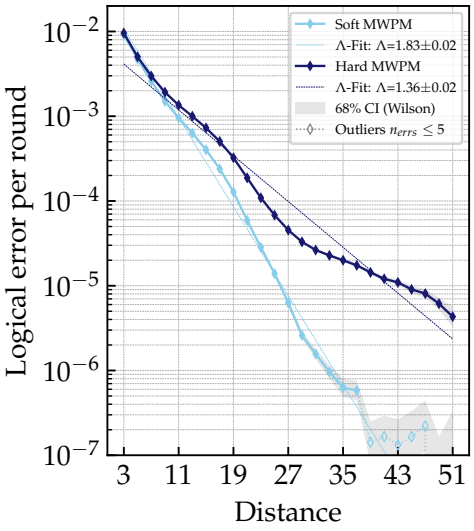
Lambda factor Λ. It includes hard and soft decoding results for every odd distance from $d = 3$ to $d = 51$, along 68% confidence intervals using the Wilson score interval method. Similar to the simulation results, points representing fewer than *five* logical error events appear as dashed lines, indicating 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 our calibration 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.

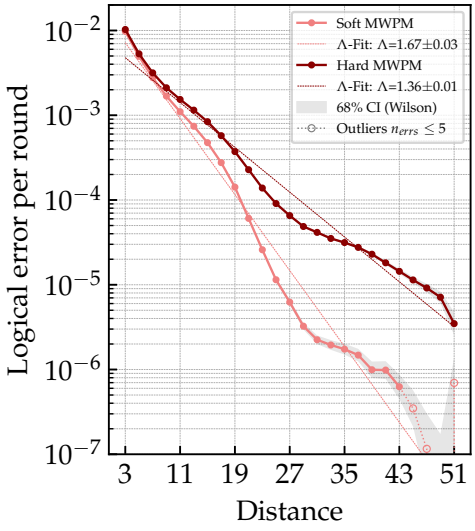
Despite these higher error rates, they maintain a linear relationship with distance on a logarithmic scale, indicative of the expected exponential suppression with increasing code size. Although some curvature was already evident in our simulations, attributed to subsampling, this effect appears to be more pronounced in the hardware data. We suspect that the anisotropy of the hardware noise, which leads to varying error rates across different qubit subsets, amplifies the subsampling curvature effects. This noise variation might also contribute to poorer error rates, particularly if specific low-quality qubits disrupt the repetition code chain, drastically reducing code performance as discussed in Section 4.1.2. Moreover, the logical error rate curves exhibit some variability depending on the prepared logical



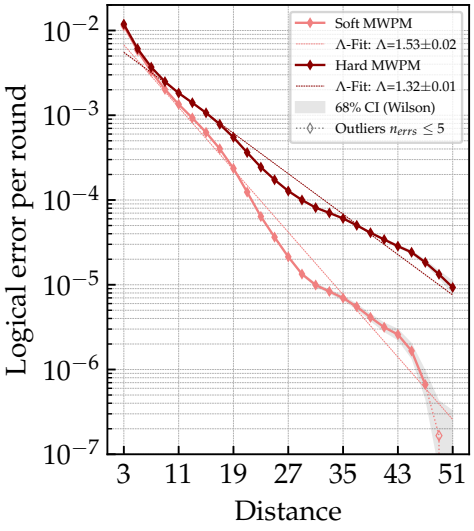
(a) Prepared logical state $|\psi\rangle_L = |+z\rangle_L$.



(b) Prepared logical state $|\psi\rangle_L = |-z\rangle_L$.



(c) Prepared logical state $|\psi\rangle_L = |+x\rangle_L$.



(d) Prepared logical state $|\psi\rangle_L = |-x\rangle_L$.

Figure 4.10: `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.

Logical state	Hard Λ-factor	Soft Λ-factor	Threshold 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%

Table 4.1: 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 Figure 4.10.

state, yet the overall trend remains consistent across all states.

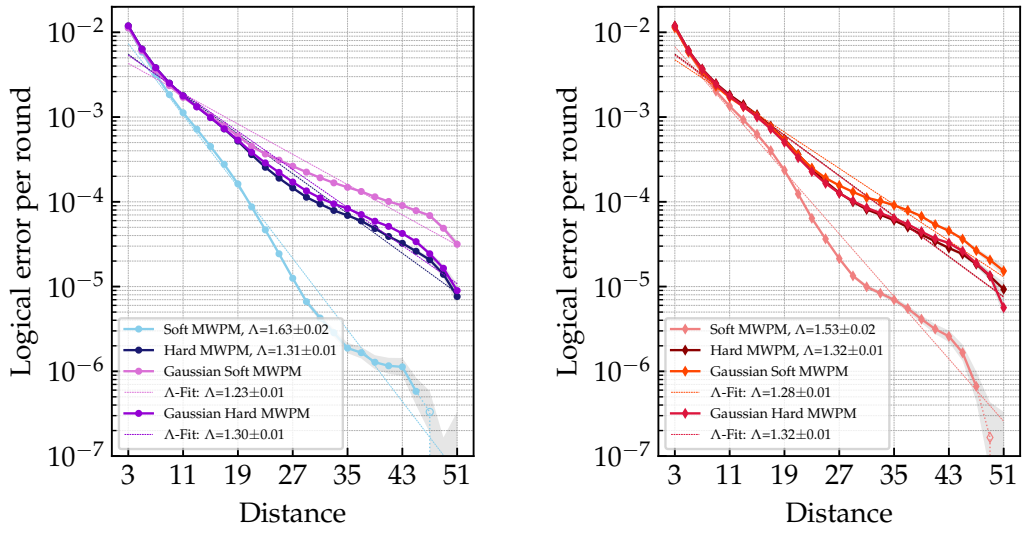
Significantly, the soft decoder consistently outperforms the hard decoder, with this advantage being even more pronounced than in simulations, as will be further addressed in Section 4.4. The soft decoder achieves logical error rates up to *30 times lower* than those of the hard decoder, with a *magnitude* 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 4.1. 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.

Other approaches to soft decoding

Until now, our analysis has employed kernel density estimation to model the IQ distributions. As outlined in Section 3.3.2, kernel density estimation is complex and computationally expensive. To ensure that it does not overparameterize the problem, we compared its performance against the simpler 1D Gaussian model, detailed in Section 3.3.3. Figure 4.11 depicts the logical error rate per round for hard and soft decoding using this Gaussian model based on the same experimental data from the previous section. To avoid redundancy, we only show one state for each basis, with the remaining data in Appendix A.1.

The comparison reveals that the Gaussian model significantly underperforms



(a) Prepared logical state $|\psi\rangle_L = |+z\rangle_L$.

(b) Prepared logical state $|\psi\rangle_L = |-x\rangle_L$.

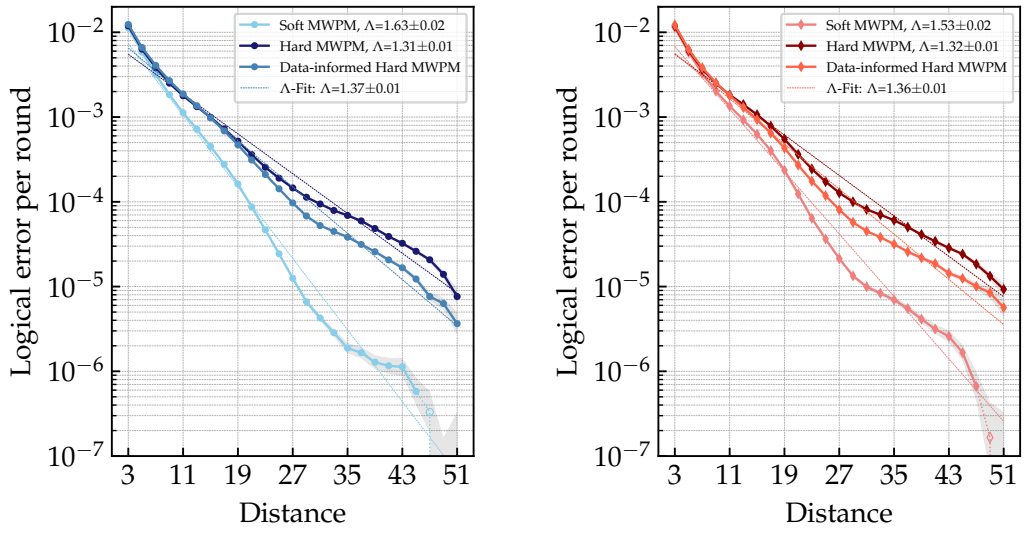
Figure 4.11: Comparison of logical errors per round for hard and soft decoding using the 1D Gaussian model and kernel density estimation, based on experimental data from `IBM Sherbrooke` for $T = 50$ rounds of measurements.

relative to the kernel density estimation technique. The logical error rates for the hard decoder remain roughly similar between the two models, suggesting comparable outcome classifications for $\hat{z}(\mu)$. However, the error rates for the Gaussian soft MWPM are even poorer than those of the hard MWPM. This disparity suggests that while the Gaussian model accurately estimates the outcome $\hat{z}(\mu)$, it fails to correctly capture the probabilities of soft flips, $p_s(\mu)$. As discussed in Section 3.3.5, the 1D Gaussian model fails to adequately account for leakage, likely contributing to the poor performance of the soft decoder.

The effectiveness of this model appears to depend on the specific IQ distributions of the device. Although it was ineffective in our case, Ali *et al.*'s research demonstrates that it can perform well under different conditions [18].

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 described in Section 3.3.4. Figure 4.12 reveals that the data-informed hard decoder outperforms the calibrated hard decoder yet falls short of the soft decoder's effectiveness[3]. This outcome underscores the critical role that dynamic reweighting plays in enhancing decoding with analog information.

[3] Additional data for the other states can be found in Appendix A.2.



(a) Prepared logical state $|\psi\rangle_L = |+z\rangle_L$.

(b) Prepared logical state $|\psi\rangle_L = |-x\rangle_L$.

Figure 4.12: 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.

4.3 Information-performance trade-off

Soft information decoding offers a promising approach to enhance error correction on quantum hardware but requires increased data collection from measurements. This increased data need complicates its practical implementation, particularly in real-time decoding scenarios. To address this issue, we explore the trade-off between the quantity of information required from measurements and the resulting performance improvement. First, we will discuss the implementation of real-time soft decoding on hardware and its challenges.

4.3.1 Hardware aspects of real-time soft decoding

As described in Section 3.1.1, the measurement process on superconducting hardware consists of probing the coupled resonator with a microwave pulse, capturing the reflected signal, and integrating this signal. The control system of the hardware oversees both the pulse control and the subsequent signal processing. Typically, the system includes multiple control cards, each responsible for a specific set of qubits. Each control card processes the data and then forwards the results to the control system's RAM for further handling.

For efficient operation, each control card is equipped with a buffer that temporarily stores processed data. However, these buffers have limited capacity. Once a buffer is full, the corresponding control card cannot accept new data until some

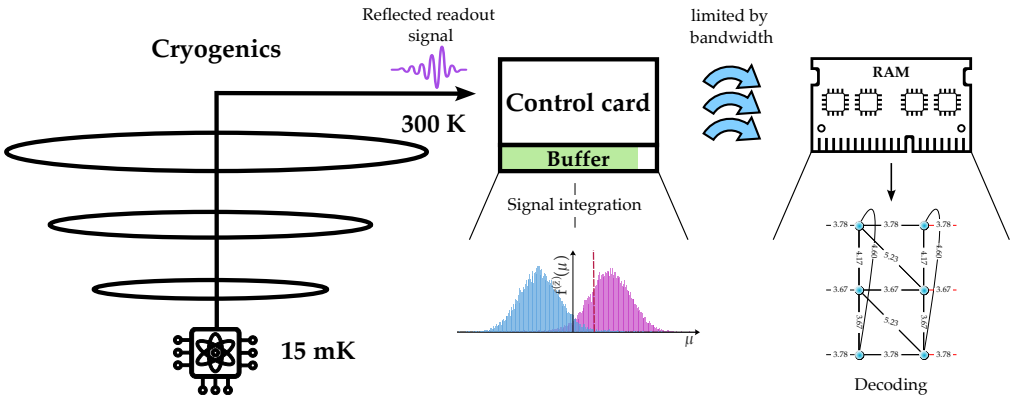


Figure 4.13: Schematic of the data flow from qubit measurements to the control system's RAM. The control card processes the reflected analog signal to get the IQ point μ and then transfers it to the RAM. A buffer, acting as short-term memory, temporarily stores this data to balance transfer speeds before the RAM receives it. The buffer fills depending on the bandwidth and data size. On the RAM, the data is accessed for subsequent decoding.

buffer contents have been transferred to the RAM. Therefore, it is crucial that the control system can process and transfer data quickly enough to prevent buffer overflow and ensure continuous data flow. The process is illustrated in Figure 4.13.

The control system's bandwidth constrains the data transfer rate from control cards to RAM. Excessive data transfer demands can overwhelm the control system, potentially causing buffer overflow, data loss, and operational disruptions. This limitation is critical in real-time decoding, where data must be processed and decoded swiftly within the duration of one or multiple stabilizer rounds.

Real-time decoding becomes particularly challenging with soft decoding, which necessitates retrieving 'full' analog data instead of mere binary outcomes. Transitioning from requiring a single bit per measurement to multiple bits significantly increases the bandwidth requirements, posing a considerable challenge from a hardware perspective.

To address this issue, we explore the trade-off between the quantity of information required from measurements and the resulting performance enhancements from soft decoding. We aim to identify the minimal amount of information necessary to preserve the advantages of soft decoding. Understanding the impact of reducing the volume of information is essential not only for optimizing data transfer but also for the decoder's operational speed, as the data size directly affects processing times. Additionally, this understanding aids in determining the required precision of analog measurement outcomes. Overall, this analysis will provide valuable insights to evaluate the practical relevance of soft decoding in time-critical scenarios, such as real-time decoding.

4.3.2 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 RAM for subsequent decoding. This method effectively reduces the data retrieved from the measurements by processing it 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 Figure 4.14 and Figure 4.15.

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 Figure 4.15, a significant number of points cluster within a specific interval, suggesting 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 the subsequent section, this primary truncation method 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.

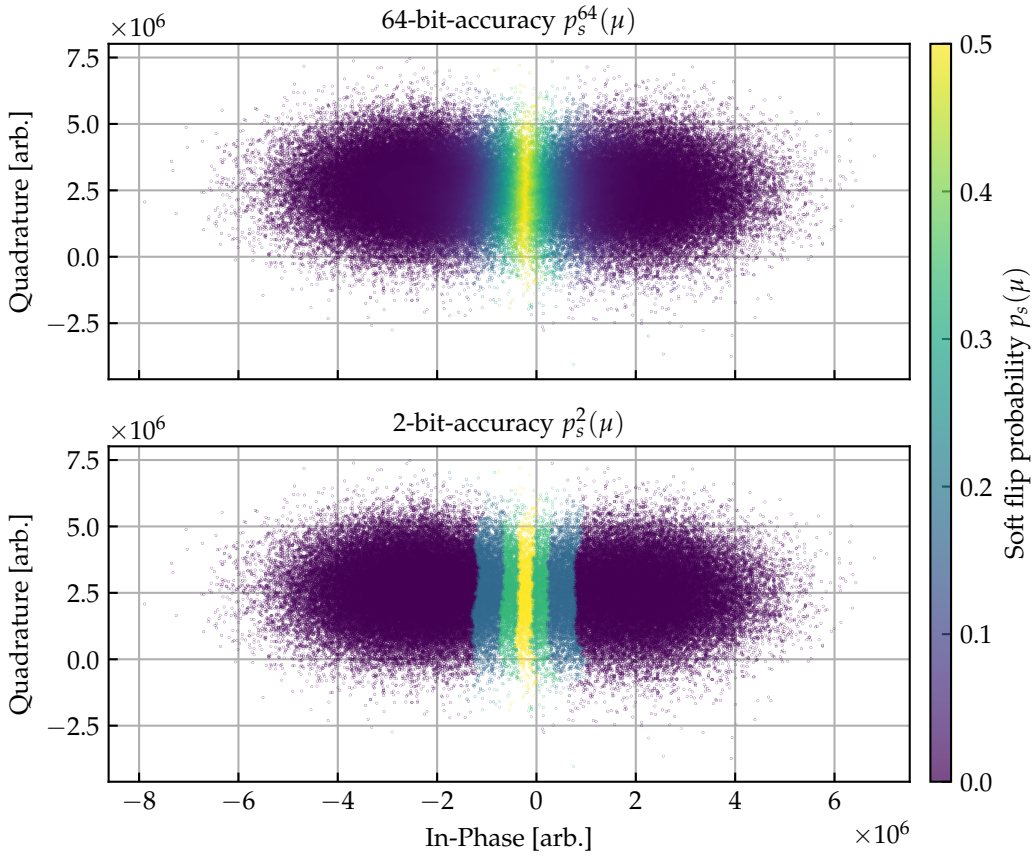


Figure 4.14: Colormap representation of measurement outcomes μ for qubit 72 of `IBM Shebrooke`. The top row shows the full accuracy soft flip probability p_s^{64}, while the bottom row illustrates the effects of truncating the soft flip probability to 2 bits.

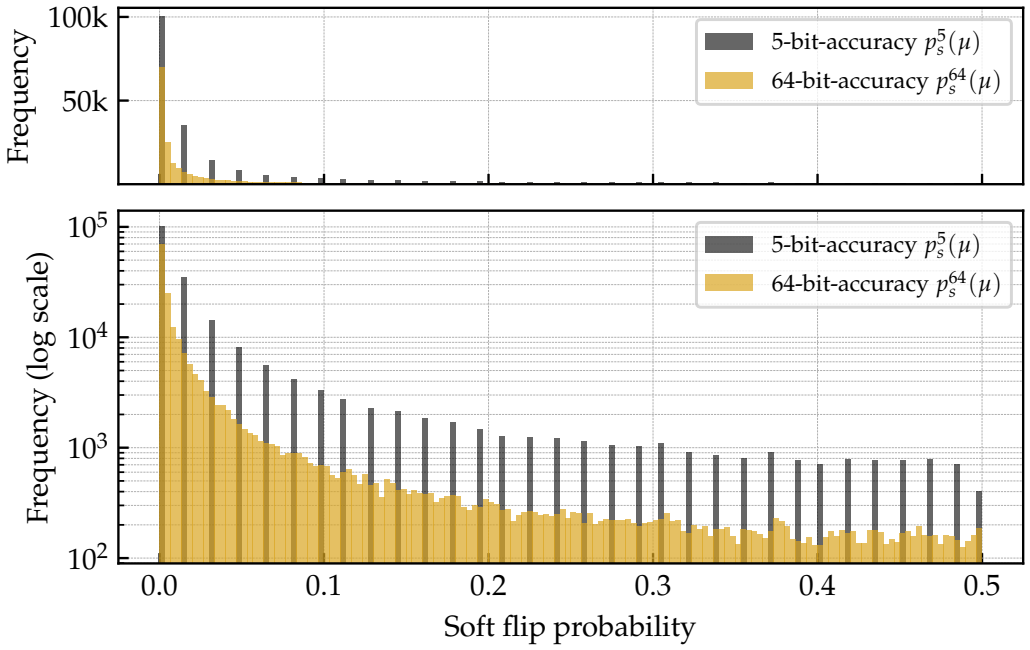


Figure 4.15: Histogram of soft flip probabilities for qubit 72 in `IBM Shebrooke`. The plot illustrates the distribution of soft flip probabilities and the effects of truncating the probabilities to 5 bits. Plotting the frequency without a logarithmic scale reveals that a significant portion of points cluster within 0% and 5%.

4.3.3 Simulation results

To assess the impact of reducing the precision of the soft flip probabilities on soft decoding, we replicate the simulations from Section 4.2.1 with $T = 50$ and incrementally increase the accuracy of the soft flip probability from $b = 1$ to $b = 15$ bits.

We calculate the ratio of the logical error rate of the decoder with truncated accuracy to that of the decoder with total accuracy. This calculation, represented as P_L^b / P_L^{64}, is plotted against the number of bits b used for the truncation. A ratio of 1 suggests that the decoder with truncated accuracy performs on par with the full-accuracy decoder. This ratio 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 ratio. Consequently, we use the simulations at twice the device's physical error rates, similar to those shown in Figure 4.8. 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

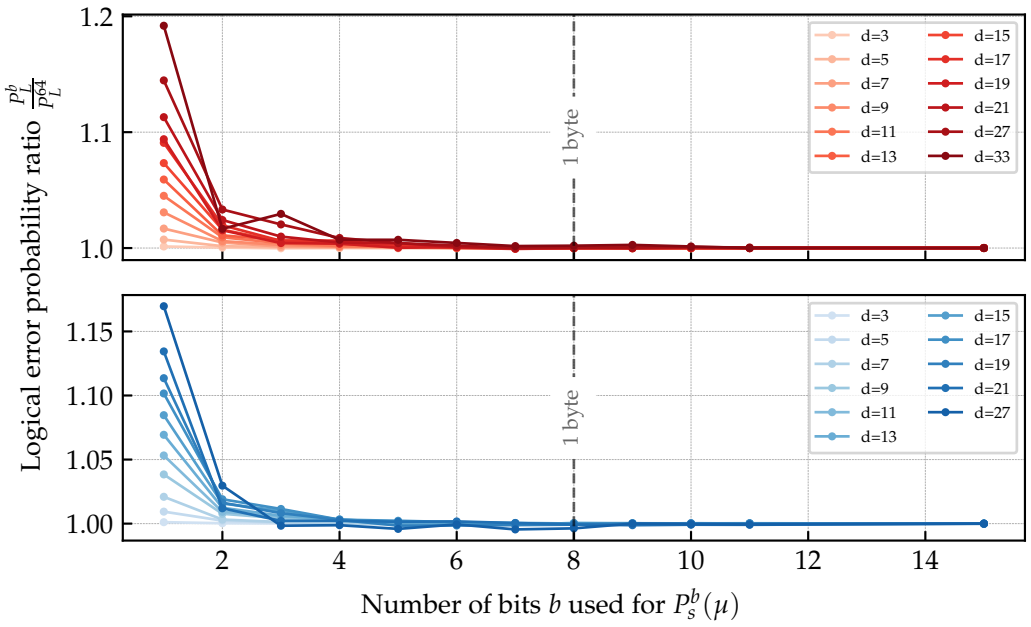


Figure 4.16: Simulated logical error probability ratio P_L^b / P_L^{64} for varying code distances as a function of the number of bits b used to truncate the soft flip probabilities for the states $|+x\rangle$ in red and $|+z\rangle$ in blue. The simulations are conducted at twice the physical error 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.

in Figure 4.16.

Most notably, the data reveals that the ratio P_L^b / P_L^{64} consistently converges to 1 across all distances and both bases, resulting in a *convergence point*. Remarkably, this convergence occurs at $b = 6$ bits, which indicates that just *one byte* of information per measurement is more than sufficient to match the performance of the full-accuracy decoder. Moreover, we note that large code distances exhibit a more rapid convergence. This trend implies that larger codes are particularly sensitive to the precision of the soft flip probabilities, likely due to their increased complexity and the ensuing accumulation of inaccuracies during the decoding process.

4.3.4 Superconducting hardware results

To validate the simulation results, we implemented the same methodology on hardware, revisiting experimental data from Section 4.2.2 with $T = 50$ using reduced accuracy soft flip probabilities. The findings, illustrated in Figure 4.17, showcase the logical error rate ratio P_L^b / P_L^{64} for varying code distances and different bit resolutions b. We only show the results for the $|+z\rangle_L$ and the $|-x\rangle_L$ states for compactness, with the remaining data in Appendix A.3.

Additionally, we limited our evaluation to code distances up to $d = 33$ for the

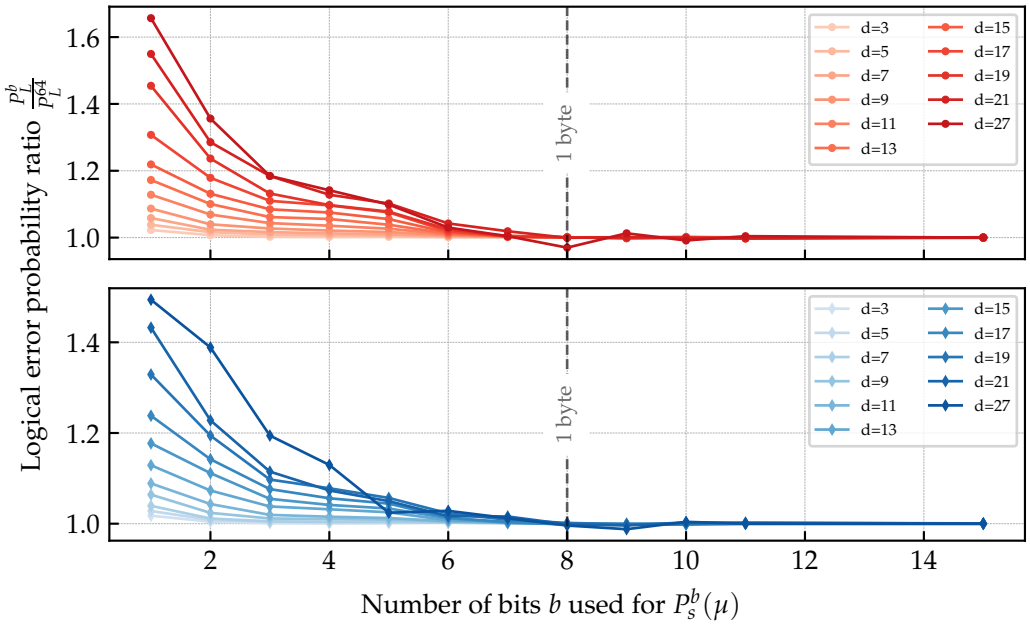


Figure 4.17: `IBM Sherbrooke` device data depicting the logical error probability ratio P_L^b / P_L^{64} for varying code distances as a function of the number of bits b used to truncate the soft flip probabilities. The states $|+x\rangle$ and $|-z\rangle$ are displayed in red and blue, respectively. Distances are limited to $d = 33$ for the X-basis and $d = 27$ for the Z-basis to maintain a sufficiently stable probability ratio.

X and $d = 27$ for the Z basis, maintaining high error rates necessary for stable ratio calculations. We imposed these constraints in line with the observed advantages of soft decoding on hardware in Section 4.2.2, where error rates beyond these thresholds were crucially lower, implying significantly variable ratios.

The hardware results mirror the simulation outcomes, with the logical error rate ratio converging to 1 across all distances and states. Again, more considerable 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 convergence point across all distances at a specific number of bits—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, this convergence point occurs 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.

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. This topic presents a potentially valuable area for future research.

4.4 Effects of leakage on soft decoding

As outlined in Section 3.3.5, leakage presents a significant challenge to quantum error correction by introducing correlated errors that critically impact fault tolerance. In this section, we will delve into the details that arise from leakage in the context of our repetition code experiments. Initially, we will assess the extent of leakage and its effect on our decoding processes. Subsequently, we will discuss how adeptly handling leakage significantly contributes to the advantages of the soft decoder over the hard decoder.

4.4.1 Quantifying leakage

As discussed in Section 3.3.5, many quantum error correction experiments chose to post-select data to exclude instances of leakage, ensuring it does not compromise error correction [32, 37, 9, 18]. However, this method becomes less feasible with larger system sizes and longer experiment durations as stray excitations accumulate, reducing experimental yield to nearly zero.

To quantify the increase in leakage from round to round, we analyzed the leakage population based on deviations of IQ points from the calibration data as described in Section 3.3.5. However, the analysis tool detects general anomalies and thus inadvertently categorizes outliers corresponding to the $|0\rangle$ and $|1\rangle$ states as leakage. To correct this overestimation and correctly quantify the leakage proportion from round to round, we started with the assumption of minimal leakage in the initial round to establish a baseline outlier ratio. This baseline is then subtracted from subsequent rounds to identify genuine leakage more accurately. Figure 4.18 plots this method's results, showing the detected leakage population over $T = 100$ rounds.

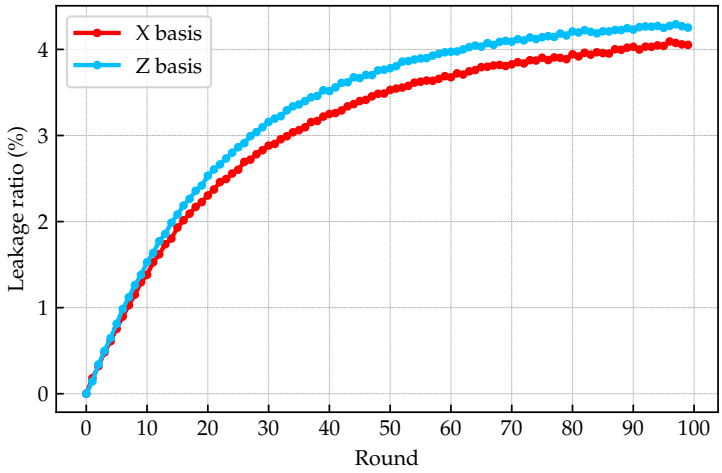


Figure 4.18: Detected leakage population ratio over $T = 100$ rounds of syndrome measurements of a repetition code on `IBM Sherbrooke`. The averaged ratios for the X and Z basis are shown in red and blue, respectively.

While the leakage rate steadily increases through the first 50 rounds, it begins to level off afterward. This stabilization is likely due to the system reaching a *steady state of excitations*. This behavior aligns with the findings of Miao *et al.*, who observed a similar trend in leakage rates over rounds [19]. Remarkably, by the 10th round, leakage affects more than 1% of all measurements, and this figure rises to more than 4% by the 100th round, highlighting a substantial increase in leakage throughout the experiment.

As discussed in Section 3.3.5, our outlier detection method inherently fails to detect a proportion of leaked states. A comparison with results from Sundaresan *et al.* reveals that while they analyze the proportion of leakage from repeated measurements excluding leakage from gates, they find a leakage population of around 10% after just 50 measurements on `IBM Peekskill` [37]. These results show a 2.5-fold higher leakage estimate, validating the assumption that our method underestimates the leakage population.

These results underscore the impossibility of post-selecting leakage-free data in large-scale experiments without dedicated leakage mitigation techniques.

4.4.2 Impact of leakage on decoding

Miao *et al.* have shown that within the context of repetition codes, leakage has a markedly more detrimental effect on code performance than Pauli errors, as depicted in Figure 4.19.

To demonstrate this effect in our experiments, we ran repetition codes similar to those presented in Section 4.2.2 but varied the number of rounds from $T = 10$ to $T = 100$. We charted the logical error per round ϵ_L against distance for various total

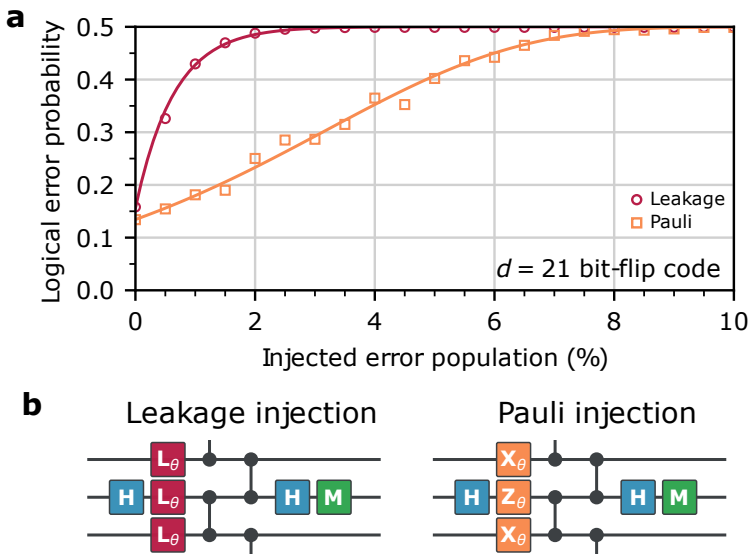


Figure 4.19: a. Detrimental effects of leakage on the decoding performance of a distance-21 repetition code over $T = 60$ rounds of measurements under either injected leakage (red) or injected Pauli errors (orange). We modified this figure from the original by removing additional curves to focus on comparing experimental data of repetition codes without resets. **b.** Circuits for the repetition code, showing the injection of leakage (left) and Pauli errors (right). Adapted and simplified from [19].

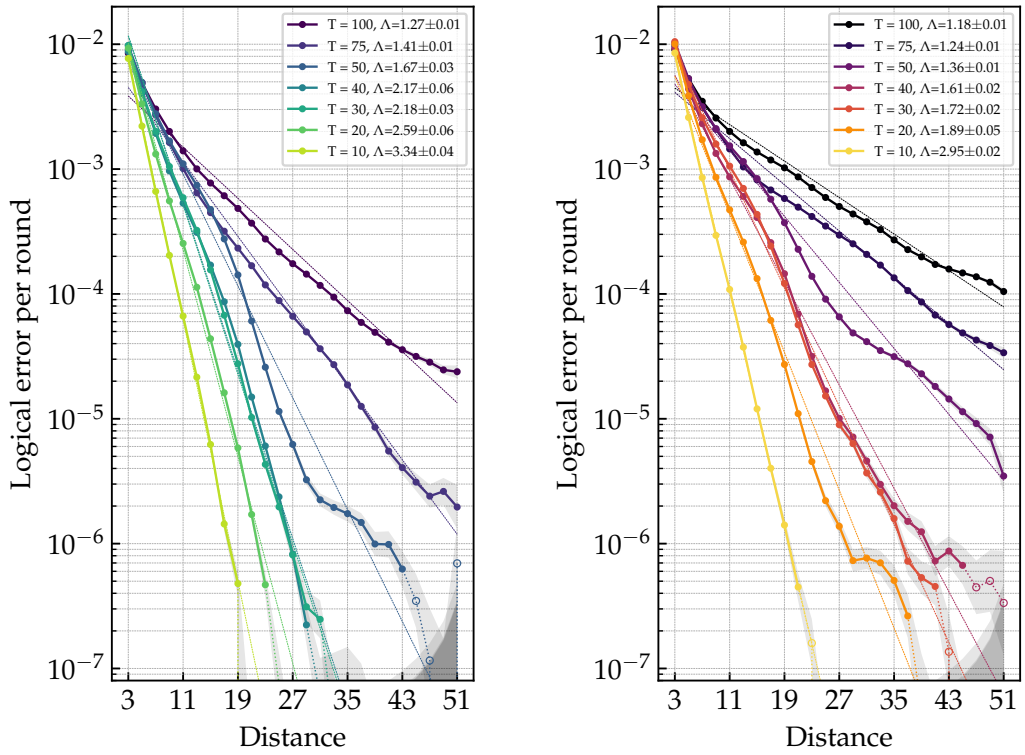
round counts T using both hard and soft decoders in Figure 4.20. Additionally, we depicted the fitted lambda factors for each round. For compactness, we only show the results for the $|+x\rangle$ state as the behavior is consistent across all prepared states, with the remaining data in Appendix A.4.

The logical error rate 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 reveals the substantial effect of leakage on both decoding strategies, even when handling leaked measurement outcomes as maximally ambiguous in soft decoding. Moreover, the observed increase in logical error rate is consistent with the findings from Kelly *et al.*, who noted a decrease in the Λ-factor from round to round when implementing repetition codes on hardware, as shown in [26].

4.4.3 Benefits of soft decoding in the presence of leakage

Despite treating leaked outcomes as maximally ambiguous, the soft decoder is significantly affected by leakage.

To illustrate the effect of assigning a maximally ambiguous soft flip probability p_s to leaked outcomes, we analyze the IQ points from a qubit particularly susceptible to leakage in our experiment. Figure 4.21 color-codes these points based on their



(a) Soft MWPM decoder.

(b) Hard MWPM decoder.

Figure 4.20: `IBM Sherbrooke` device data depicting the logical error per round plotted against distance using both soft and hard MWPM for varying total round counts T. Results are shown for the $|+x\rangle$ state, with consistent behavior across all prepared states.

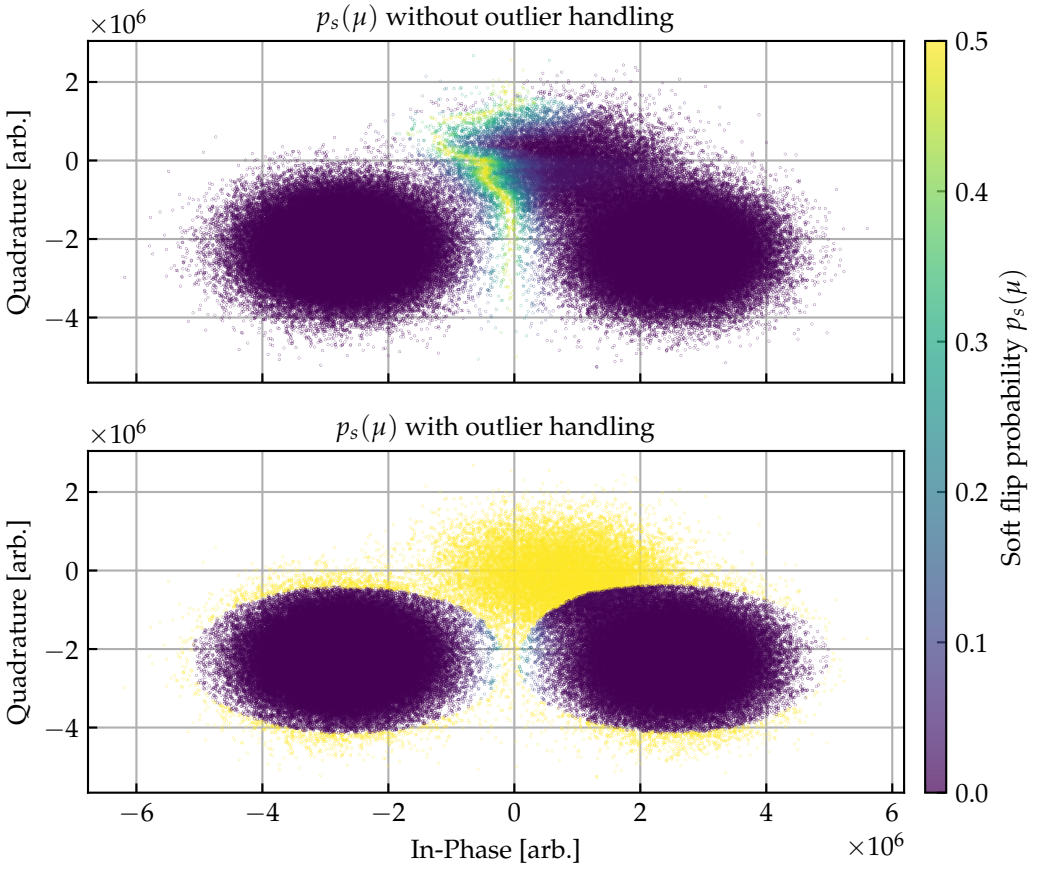
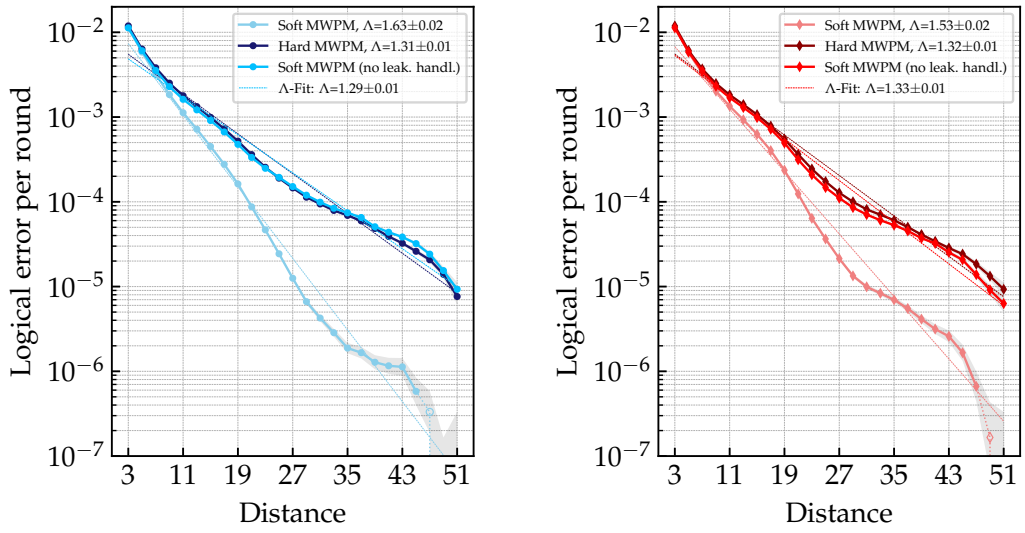


Figure 4.21: Colormap representation of stabilizer measurement outcomes μ for qubit 3 of `IBM Sherbrooke` which is particularly susceptible to leakage. The top row shows the original data, and the bottom row displays the data with leaked points treated as maximally ambiguous.

$p_s(\mu)$ values.

The analysis reveals that without setting the $p_s(\mu)$ value of leaked points to maximally ambiguous, they typically receive relatively low $p_s(\mu)$ values. Consequently, the decoder does not automatically treat these points as highly uncertain. Furthermore, our approach imperfectly selects leaked points, failing to adequately capture the Gaussian shapes for all three states and struggling to differentiate between the leaked $|2\rangle$ state and the excited $|1\rangle$ state. This shortcoming underscores the limitations of our original experimental setup, which was not intended for leakage detection, and highlights the retrospective development of our selection method, necessitating the use of existing data without the opportunity for new measurements.

However, despite these limitations, employing this method within the decoder offers substantial advantages. To highlight the significance of our leakage managing technique in the soft decoding process, we analyzed the $T = 50$ data with and



(a) Prepared logical state $|\psi\rangle_L = |+z\rangle_L$.

(b) Prepared logical state $|\psi\rangle_L = |-x\rangle_L$.

Figure 4.22: Comparison of logical errors per round for hard and soft decoding with and without treating leaked outcomes as maximally ambiguous. The results are based on experimental data from `IBM Sherbrooke` for $T = 50$ rounds.

without treating leaked outcomes as maximally ambiguous. The results, displayed in Figure 4.22, also include a comparison with the hard decoder. For compactness, we only depict the results for the $|+z\rangle$ and $|-x\rangle$ states, with the remaining data in Appendix A.5.

The findings show that the logical error rate is considerably higher when leaked outcomes are not treated as maximally ambiguous and are even approaching the error rates observed with the hard decoder. Moreover, the performance is worse than when using data-informed weights for the hard decoder, as shown in Figure 4.12. This demonstrates that failing to treat these points as maximally ambiguous substantially underestimates their uncertainty, resulting in less effective decoding. 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.

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.

Further analysis of the lambda factors for hard and soft decoders across rounds, plotted as averages over all prepared states in Figure 4.23, reveal several crucial insights.

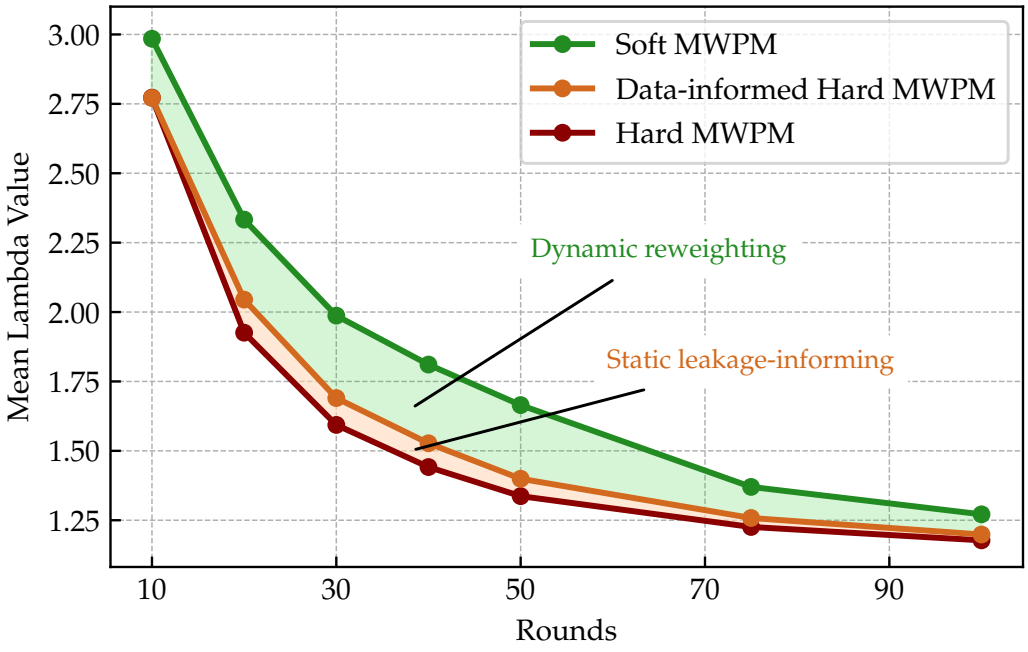


Figure 4.23: Mean lambda factors for hard, soft, and data-informed decoding methods for varying total round counts. The lambda factors are averaged over all prepared states and are calculated from `IBM Sherbrooke` device data.

Firstly, both data-informed and soft reweighting strategies consistently exhibit higher lambda factors across all rounds than the hard decoder, aligning with prior findings.

Secondly, the lambda factors for the hard decoder closely mirror the behavior of leakage populations, albeit inversely. Specifically, the lambda factor sharply decreases as rounds increase from 10 to 50 and then begins to stabilize or converge. This stabilization reflects the leakage population's convergence to a steady state observed in Figure 4.18.

Conversely, while following a similar trend at the extremes of low and high rounds, the soft decoder's lambda factors show a more flattened curve in the mid-range of rounds. This trend suggests that our leakage handling method effectively manages leakage up to a certain level in our soft decoder. Moreover, this flattening in the mid-rounds notably increases the difference in lambda factors between the soft and hard decoders compared to the lower or higher rounds.

To further demonstrate this variable advantage, we calculated the average threshold improvement as a function of rounds in Figure 4.24. Here, we see that the benefit of soft decoding peaks in the mid-range of rounds and diminishes for both lower and higher rounds.

We can attribute this pattern to several causes: with fewer rounds, two contributing factors emerge—fewer temporal edges are available for targeted adjustments,

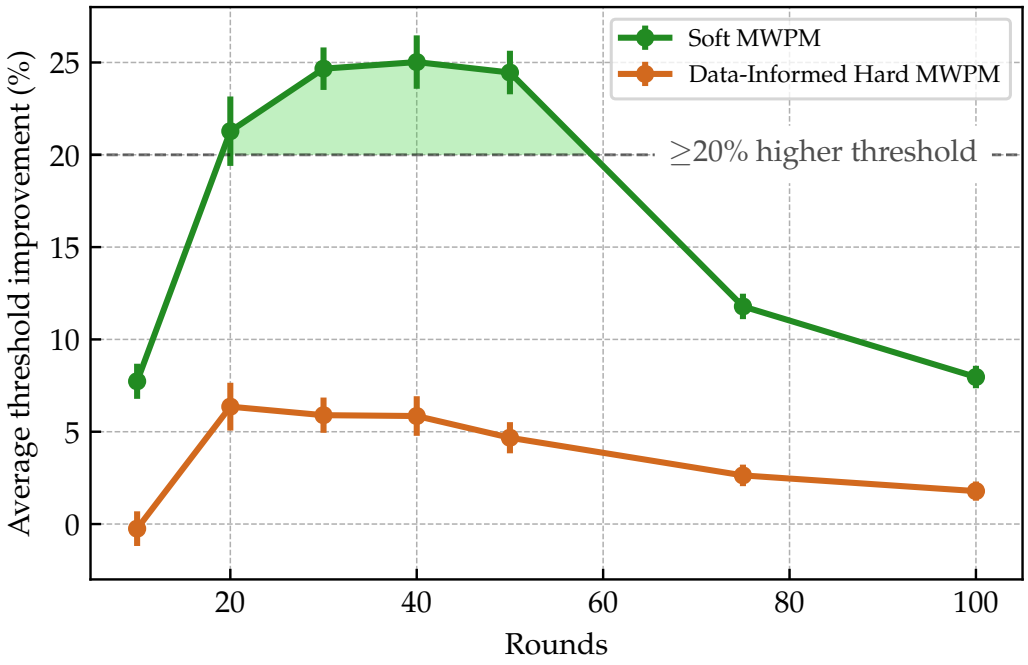


Figure 4.24: Average threshold improvement of soft and data-informed decoding over hard decoding as a function of the number of rounds. The results are based on experimental data from `IBM Sherbrooke`.

and the leakage population is lower, diminishing the soft decoder's advantage in handling leaked states. Conversely, excessive rounds lead to heightened 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. We previously observed this dynamic in Section 4.2.1 when comparing the simulated benefits of soft decoding under once and twice the physical error rates, where benefits decreased from 13.6% to 4.2% with increased error rates. 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 mid-rounds. In low rounds, leakage is minimal, so the mean ambiguity of the IQ points remains close to that obtained from calibration data. In mid-rounds, 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, in high rounds, extensive leakage affects surrounding qubits and gates, thus diminishing the benefits of informing the de-

coder about the mean ambiguity of measurement edges compared to mid-rounds.

4.5 Conclusion

In conclusion, our results demonstrate that soft decoding offers a definite advantage over hard decoding. By simulating the repetition code experiments for $T = 50$ rounds with a hardware-inspired model, we observed an exponential decrease in logical error rates, attributed to a 13.6% increase in the threshold of soft MWPM compared to hard MWPM.

In contrast, our hardware experiments demonstrate a more 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 than the hard decoder. This advantage primarily stems from dynamically reweighting based on measurement outcomes, as 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 any of the decoder's performance. This reduction underscores the feasibility of soft information decoding in real-time decoding scenarios. If needed by hardware requirements, we can further reduce the precision with tailored reduction schemes.

We observe that the enhanced benefit of soft over hard decoding in our device data originates from the presence of leakage and our heuristic method, as outlined in Section 3.3.5, to treat measurement outcomes stemming from leaked states as maximally ambiguous.

This procedure, though lacking a theoretical basis, markedly enhances the performance of the soft decoder compared to the hard decoder in our device data. This impact on soft decoding advantage is evident from the threshold improvements, which vary with the number of rounds and, thus, the leakage population. We observe that the benefit of soft decoding peaks in the mid-range 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 in lower rounds, diminishing the soft decoder's advantage in handling leaked states. Conversely, in higher 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.

This discovery is problematic because it implies that a significant portion of the observed advantage on hardware arises from our heuristic approach to managing leakage as maximally ambiguous rather than from the overlap of Gaussian distributions. This is further demonstrated by the difference in threshold increase of 13.6% in simulations and 24.4% in hardware due to the absence of leakage in our

simulation model. The lack of a theoretical foundation for this method not only raises concerns about whether a more refined approach could yield even more significant benefits but also whether our method is situation-specific and only effective in our particular experimental setting.

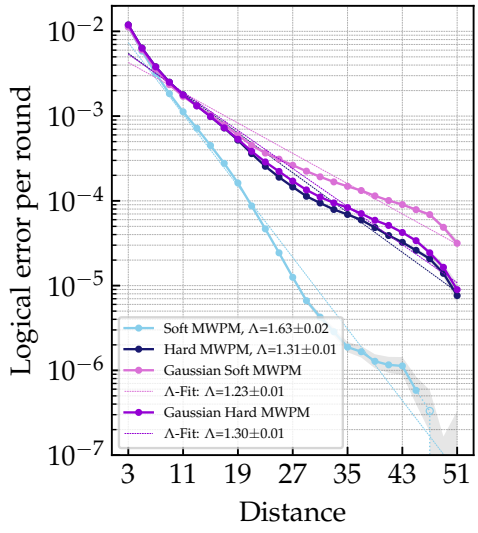
Despite these concerns, we believe that our heuristic method is a straightforward and efficient way to improve the performance of soft decoding in scenarios involving leakage. However, it is not a universal remedy, particularly in cases of severe leakage. Therefore, addressing significant leakage through 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.

Appendix A

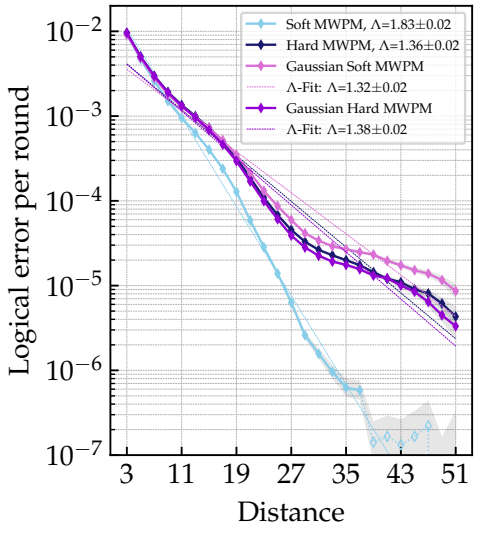
Additional data

In this appendix, we provide additional data that was not included in the main text.

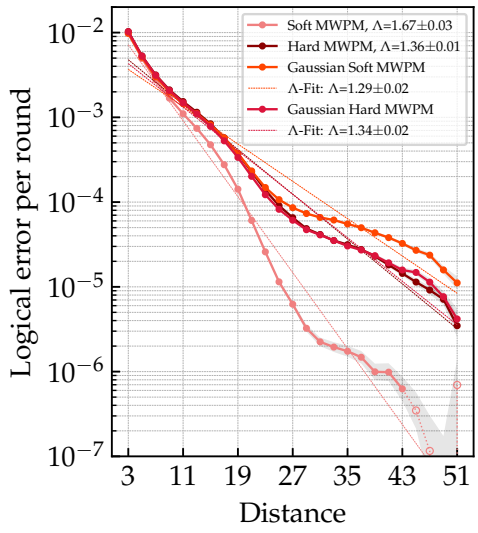
A.1 Gaussian 1D model decoding



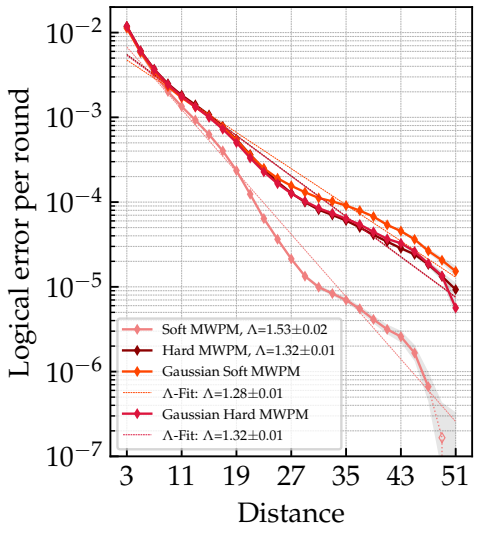
(a) Prepared logical state $|\psi\rangle_L = |+z\rangle$.



(b) Prepared logical state $|\psi\rangle_L = |-z\rangle$.



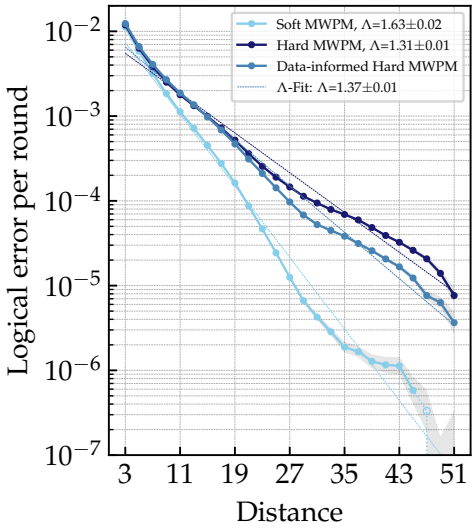
(c) Prepared logical state $|\psi\rangle_L = |+x\rangle$.



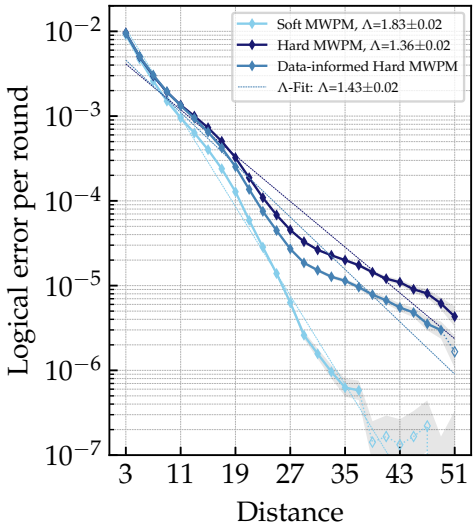
(d) Prepared logical state $|\psi\rangle_L = |-x\rangle$.

Figure A.1: Comparison of logical errors per round for hard and soft decoding using the 1D Gaussian model and kernel density estimation, based on experimental data from `IBM Sherbrooke` for $T = 50$ rounds of measurements.

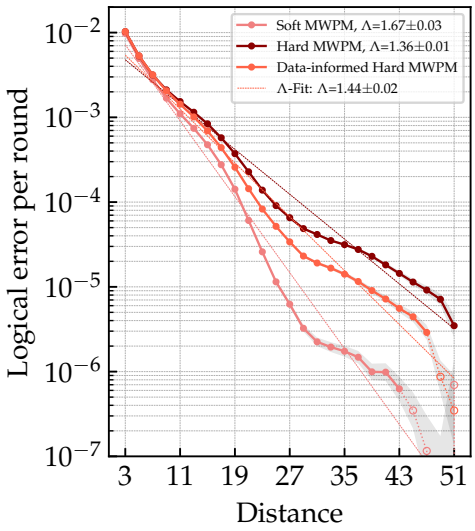
A.2 Data-informed hard decoding



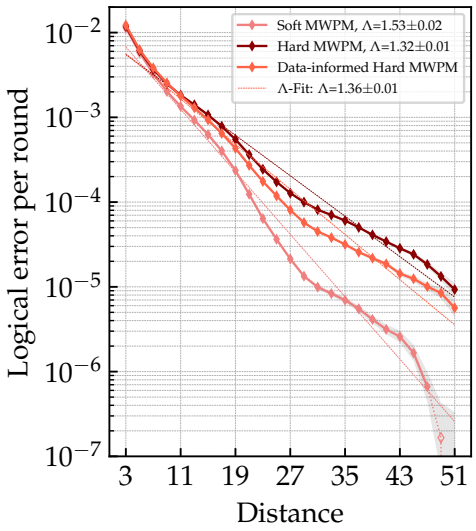
(a) Prepared logical state $|\psi\rangle_L = |+z\rangle$.



(b) Prepared logical state $|\psi\rangle_L = |-z\rangle$.



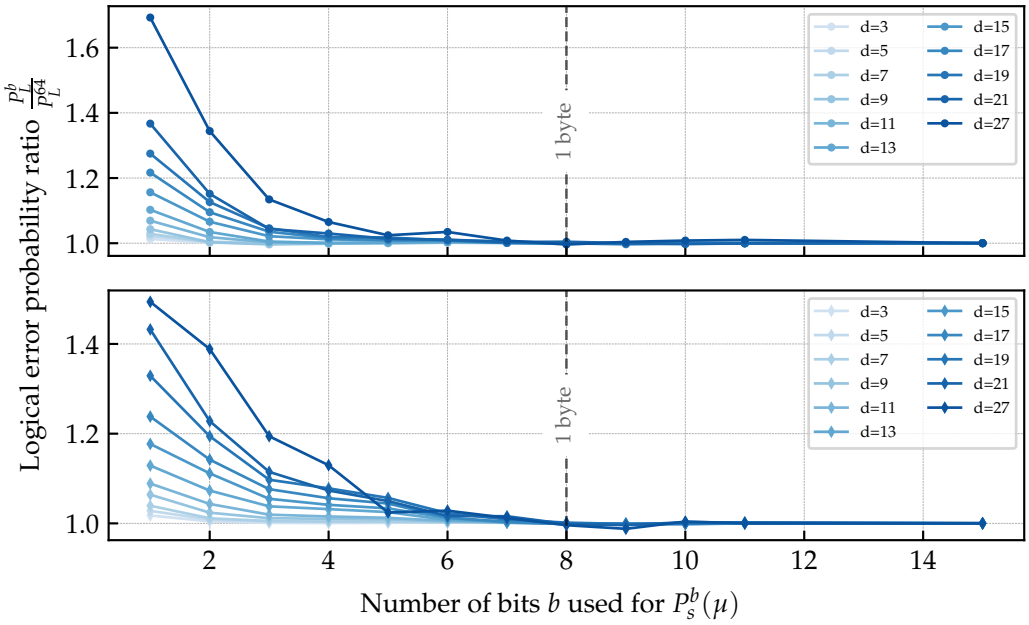
(c) Prepared logical state $|\psi\rangle_L = |+x\rangle$.



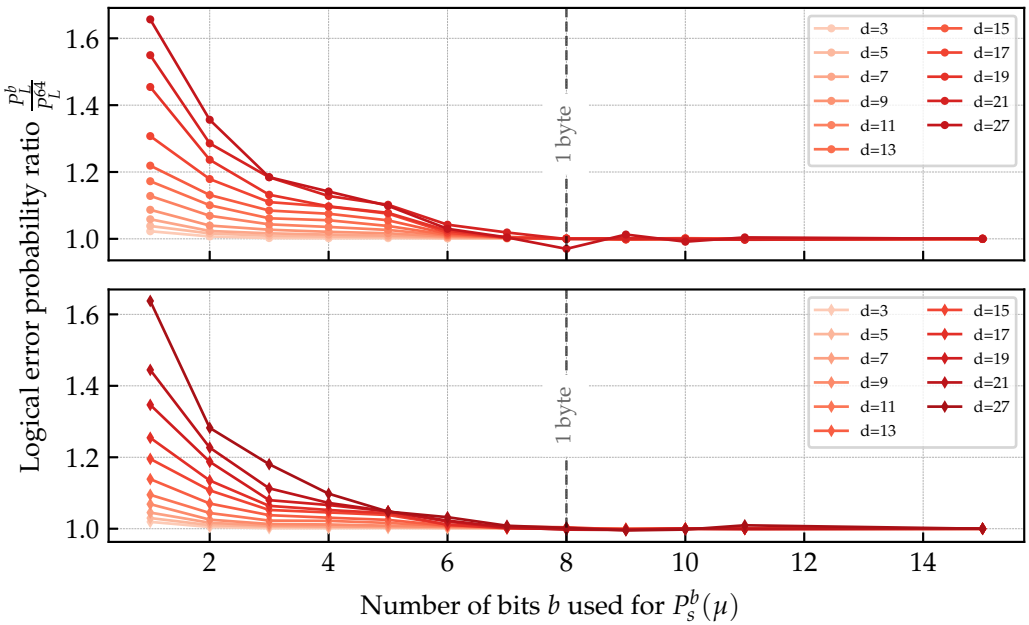
(d) Prepared logical state $|\psi\rangle_L = |-x\rangle$.

Figure A.2: 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.3 Information performance trade-off



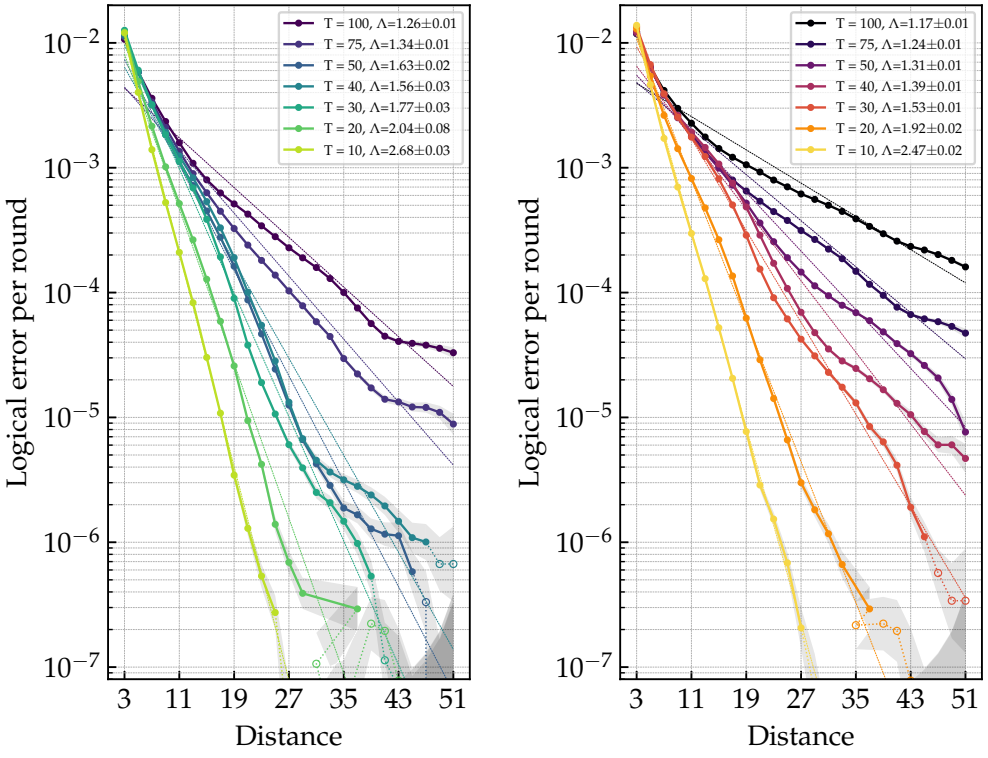
(a) Prepared logical states in the Z basis



(b) Prepared logical states in the X basis

Figure A.3: `IBM Sherbrooke` device data depicting the logical error probability ratio 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. Distances are limited to $d = 33$ for the X-basis and $d = 27$ for the Z-basis to maintain a sufficiently stable probability ratio.

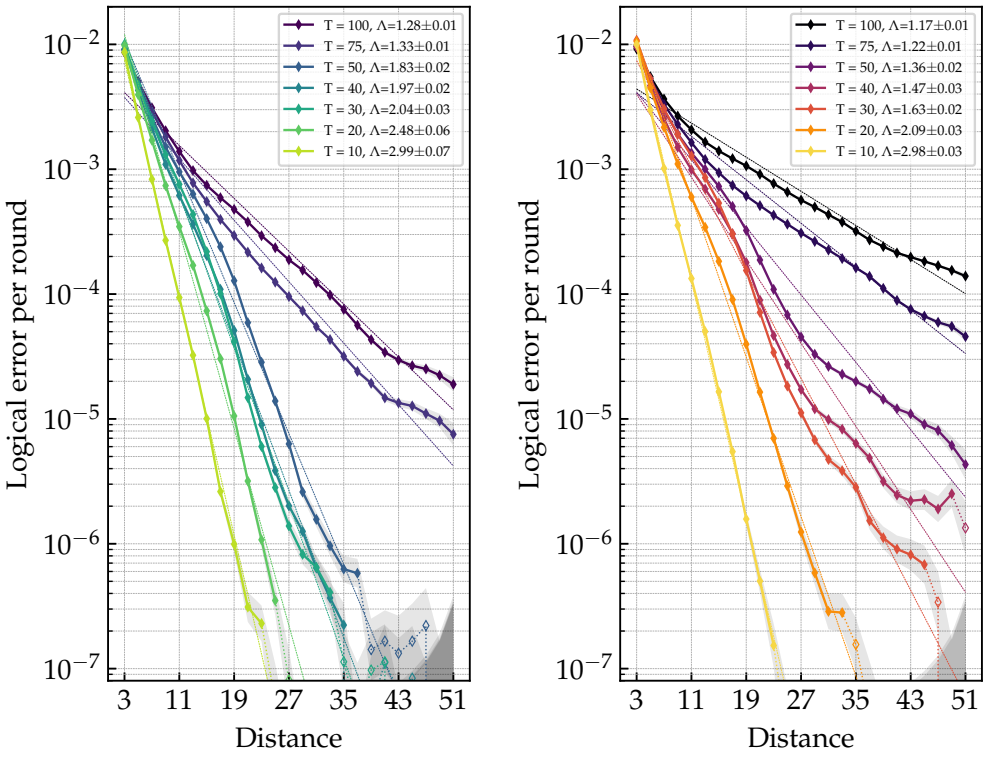
A.4 Logical error per round for different rounds



(a) Soft MWPM decoder.

(b) Hard MWPM decoder.

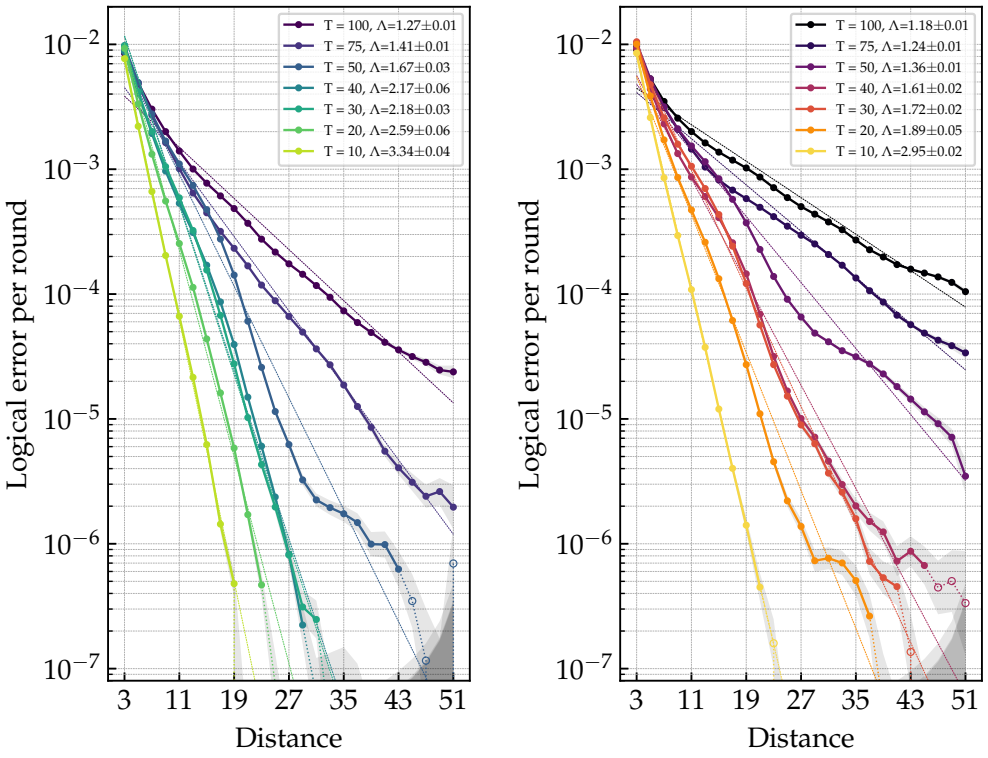
Figure A.4: `IBM Sherbrooke` device data depicting the logical error per round plotted against distance using both soft and hard MWPM for varying total round counts T. Results are shown for the $|+z\rangle$ state, with consistent behavior across all prepared states.



(a) Soft MWPM decoder.

(b) Hard MWPM decoder.

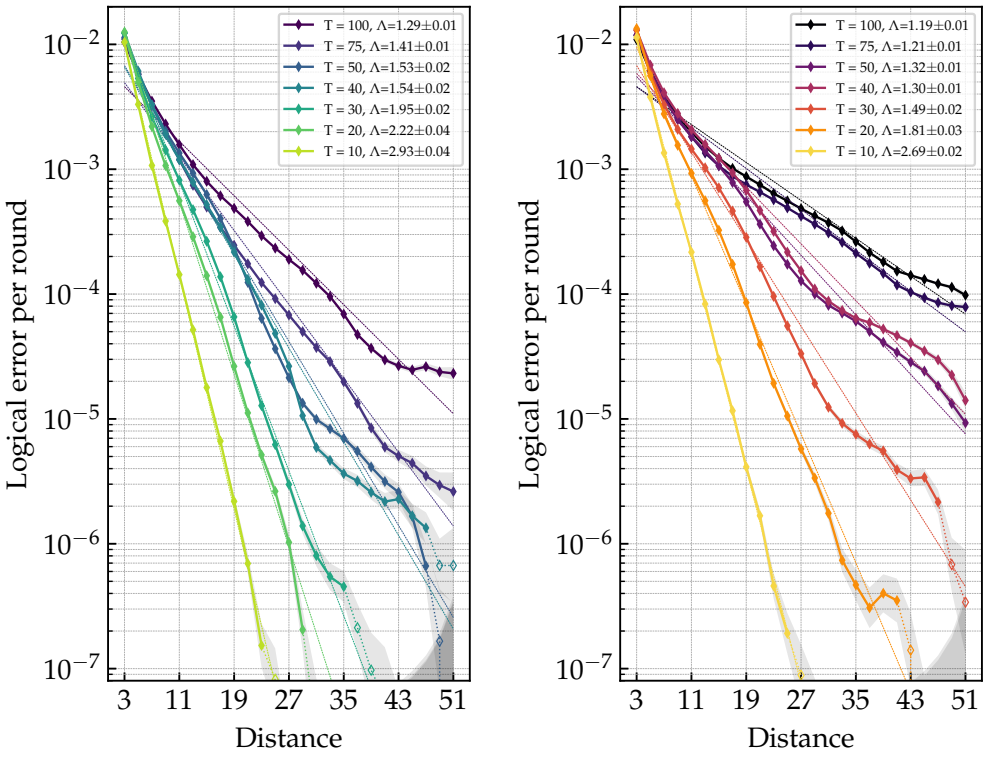
Figure A.5: `IBM Sherbrooke` device data depicting the logical error per round plotted against distance using both soft and hard MWPM for varying total round counts T. Results are shown for the $|-z\rangle$ state, with consistent behavior across all prepared states.



(a) Soft MWPM decoder.

(b) Hard MWPM decoder.

Figure A.6: `IBM Sherbrooke` device data depicting the logical error per round plotted against distance using both soft and hard MWPM for varying total round counts T. Results are shown for the $|+x\rangle$ state, with consistent behavior across all prepared states.

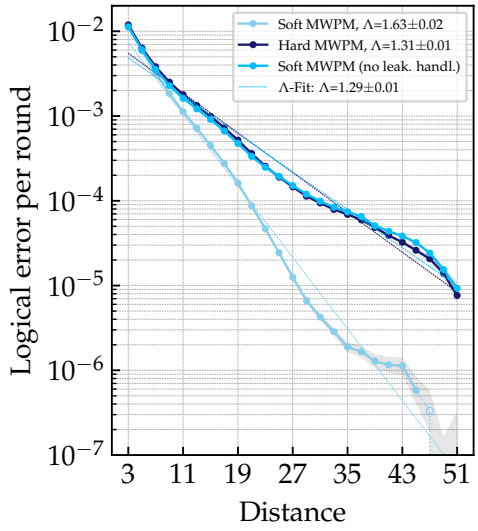


(a) Soft MWPM decoder.

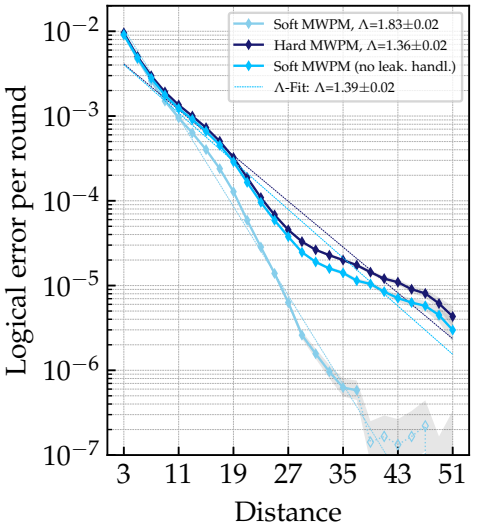
(b) Hard MWPM decoder.

Figure A.7: `IBM Sherbrooke` device data depicting the logical error per round plotted against distance using both soft and hard MWPM for varying total round counts T. Results are shown for the $|-x\rangle$ state, with consistent behavior across all prepared states.

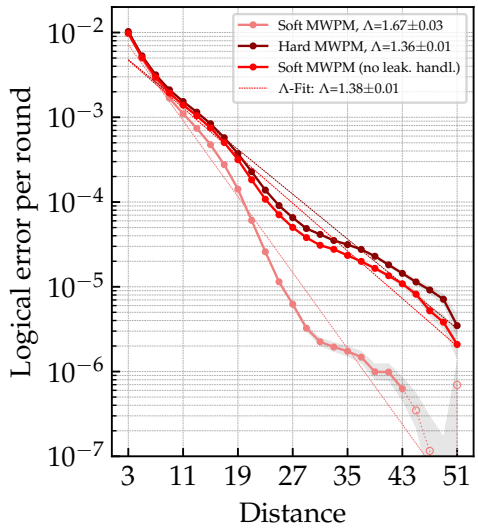
A.5 Leakage handling effect on performance



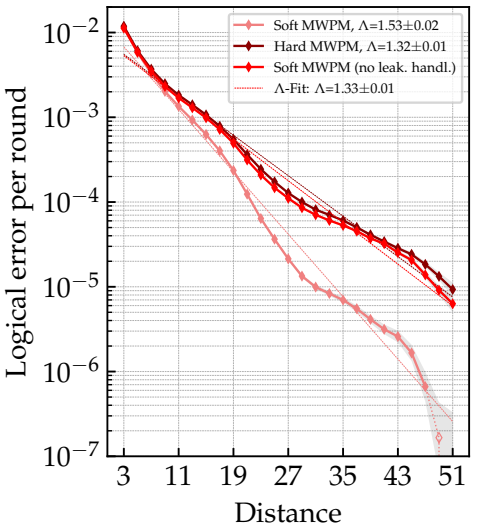
(a) Prepared logical state $|\psi\rangle_L = |+z\rangle$.



(b) Prepared logical state $|\psi\rangle_L = |-z\rangle$.



(c) Prepared logical state $|\psi\rangle_L = |+x\rangle$.



(d) Prepared logical state $|\psi\rangle_L = |-x\rangle$.

Figure A.8: Comparison of logical errors per round for hard and soft decoding with and without treating leaked outcomes as maximally ambiguous. The results are based on experimental data from `IBM Sherbrooke` for $T = 50$ rounds.

Bibliography

[1] Christopher A. Pattison, Michael E. Beverland, Marcus P. da Silva, and Nicolas Delfosse. Improved quantum error correction using soft information, 2021.

[2] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge university press, Cambridge, 10th anniversary edition edition, 2010.

[3] Andrew M. Childs and Wim Van Dam. Quantum algorithms for algebraic problems. *Reviews of Modern Physics*, 82(1):1–52, January 2010.

[4] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997.

[5] Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, August 1996.

[6] Hefeng Wang, Sabre Kais, AlÃ¡n Aspuru-Guzik, and Mark R. Hoffmann. Quantum algorithm for obtaining the energy spectrum of molecular systems. *Physical Chemistry Chemical Physics*, 10(35):5388, 2008. arXiv:0907.0854 [quant-ph].

[7] Markus Reiher, Nathan Wiebe, Krysta M. Svore, Dave Wecker, and Matthias Troyer. Elucidating reaction mechanisms on quantum computers. *Proceedings of the National Academy of Sciences*, 114(29):7555–7560, July 2017. arXiv:1605.03590 [quant-ph].

[8] Yuri Alexeev, Dave Bacon, Kenneth R. Brown, Robert Calderbank, Lincoln D. Carr, Frederic T. Chong, Brian DeMarco, Dirk Englund, Edward Farhi, Bill Fefferman, Alexey V. Gorshkov, Andrew Houck, Jungsang Kim, Shelby Kimmel, Michael Lange, Seth Lloyd, Mikhail D. Lukin, Dmitri Maslov, Peter Maunz, Christopher Monroe, John Preskill, Martin Roetteler, Martin J. Savage, and Jeff Thompson. Quantum computer systems for scientific discovery. *PRX Quantum*, 2(1):017001, February 2021.

[9] Sebastian Krinner, Nathan Lacroix, Ants Remm, Agustin Di Paolo, Elie Genois, Catherine Leroux, Christoph Hellings, Stefania Lazar, Francois Swiadek, Johannes Herrmann, Graham J. Norris, Christian Kraglund Andersen, Markus Müller, Alexandre Blais, Christopher Eichler, and Andreas Wallraff. Realizing repeated quantum error correction in a distance-three surface code. *Nature*, 605(7911):669–674, May 2022.

[10] Y. Zhao et al. Realization of an Error-Correcting Surface Code with Superconducting Qubits. *Physical Review Letters*, 129(3):030501, July 2022.

[11] Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614(7949):676–681, February 2023.

[12] Dolev Bluvstein, Simon J. Evered, Alexandra A. Geim, Sophie H. Li, Hengyun Zhou, Tom Manovitz, Sepehr Ebadi, Madelyn Cain, Marcin Kalinowski, Dominik Hangleiter, J. Pablo Bonilla Ataides, Nishad Maskara, Iris Cong, Xun Gao, Pedro Sales Rodriguez, Thomas Karolyshyn, Giulia Semeghini, Michael J. Gullans, Markus Greiner, Vladan Vuletić, and Mikhail D. Lukin. Logical quantum processor based on reconfigurable atom arrays. *Nature*, 626(7997):58–65, February 2024.

[13] M. P. da Silva, C. Ryan-Anderson, J. M. Bello-Rivas, A. Chernoguzov, J. M. Dreiling, C. Foltz, F. Frachon, J. P. Gaebler, T. M. Gatterman, L. Grans-Samuelsson, D. Hayes, N. Hewitt, J. Johansen, D. Lucchetti, M. Mills, S. A. Moses, B. Neyenhuis, A. Paz, J. Pino, P. Siegfried, J. Strabley, A. Sundaram, D. Tom, S. J. Wernli, M. Zanner, R. P. Stutz, and K. M. Svore. Demonstration of logical qubits and repeated error correction with better-than-physical error rates, April 2024. arXiv:2404.02280 [quant-ph].

[14] Riddhi S. Gupta, Neereja Sundaresan, Thomas Alexander, Christopher J. Wood, Seth T. Merkel, Michael B. Healy, Marius Hillenbrand, Tomas Jochym-O'Connor, James R. Wootton, Theodore J. Yoder, Andrew W. Cross, Maika Takita, and Benjamin J. Brown. Encoding a magic state with beyond break-even fidelity. *Nature*, 625(7994):259–263, January 2024.

[15] Boris M. Varbanov, Marc Serra-Peralta, David Byfield, and Barbara M. Terhal. Neural network decoder for near-term surface-code experiments, October 2023. arXiv:2307.03280 [quant-ph].

[16] Johannes Bausch, Andrew W. Senior, Francisco J. H. Heras, Thomas Edlich, Alex Davies, Michael Newman, Cody Jones, Kevin Satzinger, Murphy Yuezhen Niu, Sam Blackwell, George Holland, Dvir Kafri, Juan Atalaya, Craig Gidney, Demis Hassabis, Sergio Boixo, Hartmut Neven, and Pushmeet Kohli. Learning to Decode the Surface Code with a Recurrent, Transformer-Based Neural Network, October 2023. arXiv:2310.05900 [quant-ph].

[17] Xiao Xue, Benjamin D'Anjou, Thomas F. Watson, Daniel R. Ward, Donald E. Savage, Max G. Lagally, Mark Friesen, Susan N. Coppersmith, Mark A. Eriksson, William A. Coish, and Lieven M. K. Vandersypen. Repetitive Quantum Nondemolition Measurement and Soft Decoding of a Silicon Spin Qubit. *Physical Review X*, 10(2):021006, April 2020.

[18] Hany Ali, Jorge Marques, Ophelia Crawford, Joonas Majaniemi, Marc Serra-Peralta, David Byfield, Boris Varbanov, Barbara M. Terhal, Leonardo DiCarlo, and Earl T. Campbell. Reducing the error rate of a superconducting logical qubit using analog readout information, March 2024. arXiv:2403.00706 [cond-mat, physics:quant-ph].

[19] K. C. Miao et al. Overcoming leakage in scalable quantum error correction, November 2022. arXiv:2211.04728 [quant-ph].

[20] Barbara M. Terhal. Quantum Error Correction for Quantum Memories. *Reviews of Modern Physics*, 87(2):307–346, April 2015. arXiv:1302.3428 [quant-ph].

[21] Daniel Gottesman. An Introduction to Quantum Error Correction and Fault-Tolerant Quantum Computation, April 2009. arXiv:0904.2557 [quant-ph].

[22] Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. A series of books in the mathematical sciences. Freeman, New York [u.a], 27. print edition, 2009.

[23] Vladimir Kolmogorov. Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, July 2009.

[24] Nicolas Delfosse and Naomi H. Nickerson. Almost-linear time decoding algorithm for topological codes. *Quantum*, 5:595, December 2021. arXiv:1709.06218 [quant-ph].

[25] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, September 2002. arXiv:quant-ph/0110143.

[26] J. Kelly et al. State preservation by repetitive error detection in a superconducting quantum circuit. *Nature*, 519(7541):66–69, March 2015. arXiv:1411.7403 [cond-mat, physics:quant-ph].

[27] Yu Tomita and Krysta M. Svore. Low-distance surface codes under realistic quantum noise. *Physical Review A*, 90(6):062320, December 2014.

[28] Joydip Ghosh, Austin G. Fowler, and Michael R. Geller. Surface code with decoherence: An analysis of three superconducting architectures. *Physical Review A*, 86(6):062318, December 2012.

[29] M. Silva, E. Magesan, D. W. Kribs, and J. Emerson. Scalable protocol for identification of correctable codes. *Physical Review A*, 78(1):012347, July 2008.

[30] Daniel Gottesman. The Heisenberg Representation of Quantum Computers, July 1998. arXiv:quant-ph/9807006.

[31] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021. arXiv:2103.02202 [quant-ph].

[32] Google Quantum AI. Exponential suppression of bit or phase errors with cyclic error correction. *Nature*, 595(7867):383–387, July 2021.

[33] Colin D. Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M. Sage. Trapped-Ion Quantum Computing: Progress and Challenges. *Applied Physics Reviews*, 6(2):021314, June 2019. arXiv:1904.04178 [physics, physics:quant-ph].

[34] Guido Burkard, Thaddeus D. Ladd, John M. Nichol, Andrew Pan, and Jason R. Petta. Semiconductor Spin Qubits. *Reviews of Modern Physics*, 95(2):025003, June 2023. arXiv:2112.08863 [cond-mat, physics:physics, physics:quant-ph].

[35] V. A. Epanechnikov. Non-Parametric Estimation of a Multivariate Probability Density. *Theory of Probability & Its Applications*, 14(1):153–158, January 1969.

[36] Stephen T. Spitz, Brian Tarasinski, Carlo W. J. Beenakker, and Thomas E. O'Brien. Adaptive Weight Estimator for Quantum Error Correction in a Time-Dependent Environment. *Advanced Quantum Technologies*, 1(1):1800012, August 2018.

[37] Neereja Sundaresan, Theodore J. Yoder, Youngseok Kim, Muyuan Li, Edward H. Chen, Grace Harper, Ted Thorbeck, Andrew W. Cross, Antonio D. Córcoles, and Maika Takita. Demonstrating multi-round subsystem quantum error correction using matching and maximum likelihood decoders. *Nature Communications*, 14(1):2852, May 2023.

[38] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm. Simple Pulses for Elimination of Leakage in Weakly Nonlinear Qubits. *Physical Review Letters*, 103(11):110501, September 2009.

[39] Zijun Chen, Julian Kelly, Chris Quintana, R. Barends, B. Campbell, Yu Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Lucero, E. Jeffrey, A. Megrant, J. Mutus, M. Neeley, C. Neill, P. J. J. O'Malley, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, A. N. Korotkov, and John M. Martinis. Measuring and Suppressing Quantum State Leakage in a Superconducting Qubit. *Physical Review Letters*, 116(2):020501, January 2016.

[40] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and John M. Martinis. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500–503, April 2014.

[41] Fei Yan, Philip Krantz, Youngkyu Sung, Morten Kjaergaard, Daniel L. Campbell, Terry P. Orlando, Simon Gustavsson, and William D. Oliver. Tunable Coupling Scheme for Implementing High-Fidelity Two-Qubit Gates. *Physical Review Applied*, 10(5):054062, November 2018.

[42] M. A. Rol, F. Battistel, F. K. Malinowski, C. C. Bultink, B. M. Tarasinski, R. Vollmer, N. Haider, N. Muthusubramanian, A. Bruno, B. M. Terhal, and L. DiCarlo. Fast, High-Fidelity Conditional-Phase Gate Exploiting Leakage Interference in Weakly Anharmonic Superconducting Qubits. *Physical Review Letters*, 123(12):120502, September 2019.

[43] V. Negîrneac, H. Ali, N. Muthusubramanian, F. Battistel, R. Sagastizabal, M. S. Moreira, J. F. Marques, W. J. Vlothuizen, M. Beekman, C. Zachariadis, N. Haider, A. Bruno, and L. DiCarlo. High-Fidelity Controlled- Z Gate with Maximal Intermediate Leakage Operating at the Speed Limit in a Superconducting Quantum Processor. *Physical Review Letters*, 126(22):220502, June 2021.

[44] M. McEwen et al. Removing leakage-induced correlated errors in superconducting quantum error correction. *Nature Communications*, 12(1):1761, March 2021.

[45] M. Werninghaus, D. J. Egger, F. Roy, S. Machnes, F. K. Wilhelm, and S. Filipp. Leakage reduction in fast superconducting qubit gates via optimal control. *npj Quantum Information*, 7(1):14, January 2021.

[46] Martin Suchara, Andrew W. Cross, and Jay M. Gambetta. Leakage Suppression in the Toric Code, October 2014. arXiv:1410.8562 [quant-ph].

[47] Austin G. Fowler. Coping with qubit leakage in topological codes. *Physical Review A*, 88(4):042308, October 2013.

[48] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021. arXiv:2103.02202 [quant-ph].

[49] Oscar Higgott. PyMatching: A Python package for decoding quantum codes with minimum-weight perfect matching, July 2021. arXiv:2105.13082 [quant-ph].

[50] Yehuda Naveh, Elham Kashefi, James R. Wootton, and Koen Bertels. Theoretical and practical aspects of verification of quantum computers. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 721–730, Dresden, Germany, March 2018. IEEE.

[51] R. R. Curtin et al. mlpack 4: a fast, header-only C++ machine learning library. *Journal of Open Source Software*, 8(82):5026, February 2023. arXiv:2302.00820 [cs].

[52] Edwin B. Wilson. Probable Inference, the Law of Succession, and Statistical Inference. *Journal of the American Statistical Association*, 22(158):209–212, June 1927.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Title of work:

Soft Information Quantum Error Correction

Thesis type and date:

Master Thesis, April 2024

Supervision:

Dr. James Wootton
Dr. Joseph Renes
Prof. Dr. Renato Renner

Student:

Name: Maurice D. Hanisch

E-mail: mhanisc@student.ethz.ch

Legi-Nr.: 22-956-379

Statement regarding plagiarism:

By signing this statement, I affirm that I have read and signed the Declaration of Originality, independently produced this paper, and adhered to the general practice of source citation in this subject-area. Moreover, I confirm that I authored the work in question independently and in my own words, i.e. that no one helped me to athor it. Suggestions from the supervisor regarding language and content are excepted. I used generative artificial intelligence technologies[1]. In consultation with the supervisor, I did not cite them.

Declaration of Originality:

`https://ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/declaration-originality.pdf`

Zürich, 8. 5. 2024:

[1] E.g. ChatGPT, DALL E 2, Google Bard