

UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Improving the Belief-Propagation Decoder for Quantum LDPC Codes

Permalink

<https://escholarship.org/uc/item/1h86m6vk>

ISBN

9798263309145

Author

Liu, Xingrui

Publication Date

2025-08-29

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-NonCommercial License, available at <https://creativecommons.org/licenses/by-nc/4.0/>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Improving the Belief-Propagation Decoder for Quantum LDPC Codes

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Physics

by

Xingrui Liu

September 2025

Dissertation Committee:

Professor Leonid P. Pryadko, Chairperson

Professor Shan-Wen Tsai

Professor Peng Wei

The Dissertation of Xingrui Liu is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I first met my PhD advisor, Professor Leonid Pryadko, in the summer of 2019 at the QAS 2019 conference in Shenzhen, China. Although I didn't know it at the time, that meeting would mark the beginning of one of the most important mentorships in my academic journey.

Professor Pryadko is well known for his rigorous and intellectually demanding courses. Keeping up with his lectures required full concentration, but in doing so, I was able to learn a great deal. Early on, he encouraged me to take detailed notes, citing the example of his own PhD advisor, who kept extensive handwritten records. That advice helped shape my learning process.

When I first arrived in the U.S., my English was far from fluent. Professor Pryadko patiently encouraged me to spend more time practicing, and his support gave me the confidence to improve. His deep command of both mathematics and physics is something I continue to admire. I've learned a great deal simply by watching him work through problems with clarity and precision. He also dedicated an extraordinary amount of time to helping me revise my research papers, line by line, often over many sessions of one-on-one discussions. These interactions were invaluable—not only for improving the quality of the writing, but also for helping me grow as a researcher.

I am deeply grateful for his guidance, generosity, and high standards. Working with Professor Pryadko has been both a challenge and a privilege, and I have grown immensely under his mentorship. I would also like to express my heartfelt gratitude to the members of our research group. I am especially thankful to Weilei Zeng, my senior in the group, who first recommended that I join this team. When I had just arrived, he patiently explained

many essential things that helped me get started smoothly. I'm also grateful to Renyu Wang and her husband, Xianan Zhang, who kindly took us to set up phone plans and bank accounts when we first arrived in the U.S.—their help made those early, uncertain days much easier. I've also benefited a great deal from my many insightful discussions on quantum computing with Hsiang-Ku Lin, Pak Kau Lim, and Shalby Shalaby. I truly appreciate the intellectual companionship and collaborative spirit we shared. Hsiang-Ku, as a senior member of the group, generously shared not only her academic insights but also many valuable lessons about life and navigating graduate school, for which I am truly thankful. With Pak, our conversations often went beyond physics—we frequently exchanged thoughts on food, career planning, and life in general, which I found both helpful and uplifting. As for Shalby, I fondly remember the 2024 APS March Meeting, where we stayed at the same hotel. That experience was not only professionally enriching but also personally enjoyable, thanks to his thoughtful and cheerful company.

I am also grateful to the scholars and researchers outside my immediate group who have generously shared their time and insights with me. In particular, I would like to thank Prof. Ilya Dumer and Prof. Valentin Savin for the valuable academic discussions we've had. Their perspectives and deep understanding of the field have broadened my thinking and enriched my research.

I am equally grateful to my roommates, who made daily life warmer and more manageable. They generously gave me rides to campus many times when I didn't have a car, and they spent much time helping with practical matters such as apartment hunting and teaching me how to cook. Through our conversations, I also learned a great deal about

many topics, from cooking and physics to investing. I especially enjoyed going out to eat with them, and our practice of taking turns cooking every day saved us a lot of time and money. My heartfelt thanks go to Qiran Wu, Shan Zhong, Kening Zhao, Xiao Hong, Zhiyu Zhou, Zhuanlin Wang, Jingyuan Cui, Yiyang Sun, Ye Lin, and Yanhui Yang

I would also like to thank my fellow badminton partners—Haoyu Liu, Jingyan Guo, Qingyang Shi, Jinwei Zhang, Kunpeng Chen, Hengjie Chen, Qilin Zhang, Andy, and Professor Xiaoping Hu—for the fun and active moments we shared together on the court. Those games not only kept us healthy but also brought great joy and friendship into our lives.

I would also like to extend my heartfelt thanks to my friends from USTC, who welcomed me into their community and invited me to participate in several USTC alumni events. These gatherings not only brought a sense of belonging but also broadened my perspective beyond daily academic life. I am especially grateful to Qiran Wu, Haoyu Liu, Yingcong Li, Tianyi Ouyang, Junyu Tang, and Zhiyuan Song for helping build such a warm and inspiring network. Sharing those moments reminded me of the value of community and connection, even far from home.

I also greatly enjoyed my time with Muhammad Faisal Manzoor, my office mate, with whom I shared many thought-provoking discussions on serious literature.

I would also like to thank my middle school friend, Zhengcheng Song, who first sparked my interest in physics. His influence played a crucial role in my decision to pursue a career as a physicist. Over the years, we have shared many meaningful discussions about

emerging technologies and career choices, and our trip to Death Valley remains one of my most unforgettable adventures.

I am also grateful to my elementary school friend, Runzhou Tao, who, by a fortunate coincidence and shared passion, has also become a researcher in quantum computing. Our countless conversations about quantum theory, AI, and the future of the world have been filled with imagination and inspiration. We have also enjoyed many memorable fine-dining experiences together. I warmly congratulate him on his new position as an assistant professor at the University of Maryland.

I would also like to thank the medical professionals who helped me stay healthy throughout my graduate years. Jonathan D. Robbins, DPT, played a crucial role in my recovery from several injuries—thanks to his expert physical therapy, I was able to fully heal from issues with my knee, lower back, and shoulder. I am also deeply grateful to my chiropractor, Dr. Lynn Scecina, whose remarkable skills once allowed me to walk normally again almost immediately after a serious lower back injury that had left me barely able to stand. I would like to extend a special thanks to my advisor, Professor Leonid Pryadko, for recommending her to me during that difficult time.

My sincere thanks also go to Qiang Xu, Zixun Chen, Siyu Li, Shuyang Zhang, Li Yang, Yirui Gao, Wei-Cheng Liao, Fan Lu, Zijue Luo, Yuntai Song, Shuo Geng, Chujing Zheng, Yen-Wen Lu, Ta-Chou Juan and Yankang Liu. Whether through casual conversations, shared meals, or mutual support, each of you contributed in your own way to making this journey brighter and more memorable.

To anyone who is willing to read this thesis.

ABSTRACT OF THE DISSERTATION

Improving the Belief-Propagation Decoder for Quantum LDPC Codes

by

Xingrui Liu

Doctor of Philosophy, Graduate Program in Physics
University of California, Riverside, September 2025
Professor Leonid P. Pryadko, Chairperson

Belief-propagation (BP) decoding is known to underperform on degenerate quantum low-density parity-check (qLDPC) codes, where short cycles and error degeneracy induce early error floors—especially under realistic circuit-level noise. This dissertation systematically studies how to improve BP’s accuracy and runtime across multiple code families (rotated surface, toric, and generalized bicycle/GB codes) and noise models (Pauli/channel-capacity, phenomenological, and circuit-level). We provide a unified evaluation of previously proposed BP refinements—including BP+ordered-statistics decoding (OSD), randomized message-update schedules, randomized priors, freezing/decimation, and memory-augmented variants—and clarify when each method helps, how gains scale with code distance, and what computational costs are incurred.

Our simulations show that BP+OSD remains a strong baseline, often recovering the optimal or near-optimal log-log slope of logical-error rate versus physical-error rate (e.g., approaching $\lfloor (d+1)/2 \rfloor$), while purely BP-based modifications improve accuracy only

partially on surface and toric codes as d increases. For low-degeneracy GB codes, however, standard BP already approaches BP+OSD, leaving limited room for further gains. Motivated by the locality of checks and sub-threshold error clustering in many qLDPCs, we introduce a cluster-based *pre-decoder* that: (i) decomposes the measured syndrome into connected components on the syndrome-connectivity graph; (ii) attempts independent correction of each component via a precomputed hash lookup; and (iii) hands only the residual to BP(+OSD). This design reduces the *number of OSD invocations* and the *problem size seen by OSD*, yielding **orders-of-magnitude speedups** in the practically relevant low-error-rate regime with minimal accuracy loss. We characterize the trade-off via tunable parameters (e.g., component-width and exploration limits), showing that conservative settings preserve accuracy across codes and noise models, while more aggressive settings further accelerate decoding at the cost of a slight accuracy decrease in specific surface-code and circuit-level cases.

To mitigate extreme degeneracy, we also employ degeneracy-reducing matrix transformations (inverse-decoration and star-polygon) as a preprocessing step, ensuring BP corrects all weight-1 errors on transformed rotated-surface-code instances without altering maximum-likelihood performance. Overall, our results establish when BP refinements meaningfully improve accuracy, when OSD is indispensable, and how a simple cluster pre-decoder can act as an efficient front-end to modern qLDPC decoding pipelines. The methods and benchmarks presented here offer practical guidance for low-latency decoders needed in real-time fault-tolerant quantum computing under circuit-level noise.

Contents

List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Outline of the thesis	7
2 Background	9
2.1 Classical Linear Binary Codes	9
2.2 Quantum error correction	10
2.3 Stabilizer codes	11
2.4 Calderbank-Shor-Steane (CSS) codes	12
2.5 Multivariate Bernoulli Distribution	13
2.6 LDPC codes	16
2.7 General bicycle codes	18
2.8 Decoders for qLDPC codes	18
2.8.1 Minimum-weight perfect-matching decoder	19
2.8.2 Union-find decoder	21
2.9 Error models	23

2.10	Use matrix transformations to simplify the error model	28
3	Belief Propagation	31
3.1	BP algorithm	31
3.2	BP Decoder	33
3.3	BP schedules	38
3.4	Error Floors and Trapping sets	40
3.5	Ordered Statistics Decoder	41
4	Modified BP decoders	44
4.1	Random schedules	44
4.2	Randomizing probabilities priors	45
4.3	Freezing methods	46
4.4	BP with additional memory effects (MBP)	47
4.5	Pre-decoder	48
5	Numerical results	51
5.1	Comparison of modified BP methods	51
5.2	Pre-decoder	67
6	Conclusions	76
	Bibliography	78

List of Figures

2.2	BP decoding accuracy for simplified and regular rotated surface codes. . . .	30
5.1	BP+OSD for rotated surface codes	56
5.2	BP+OSD for toric codes	56
5.3	BP+OSD for toric codes under the phenomenological model	57
5.4	BP schedules for rotated surface codes	57
5.5	BP schedules for toric codes	58
5.6	BP schedules for toric codes under the phenomenological model	58
5.7	BP freezing methods for surface codes	59
5.8	BP freezing methods for toric codes	59
5.9	BP freezing methods for toric codes under the phenomenological model . .	60
5.10	Random-BP for roated surface codes	60
5.11	Random-BP for toric codes	61
5.12	Random-BP for toric codes under the phenomenological model	61
5.13	MBP for roated surface codes	62
5.14	MBP for toric codes	62
5.15	MBP for toric codes under the phenomenological model	63

5.16 BP+OSD for 3+3 GB codes	64
5.17 BP+OSD for 4+4 GB codes	64
5.18 BP+OSD for rotated surface codes circuit	65
5.19 Random-BP for rotated surface codes circuits	66
5.20 MBP for rotated surface codes circuits	66
5.21 Freezing methods for rotated surface codes circuits	67
5.22 Pre-decoder accuracy for the $d = 7$ rotated surface code	68
5.23 Pre-decoder accuracy for the $d = 7$ toric code	69
5.24 Pre-decoder accuracy for the 3 + 3 GB code	69
5.25 Pre-decoder accuracy for $d = 7$ rotated surface code circuit	70
5.26 Pre-decoder execution time for $d = 7$ rotated surface code	71
5.27 Pre-decoder execution time for $d = 7$ toric code	72
5.28 Pre-decoder execution time for 3+3 GB codes.	72
5.29 Pre-decoder execution time for $d = 7$ rotated surface code circuits	73
5.30 Pre-decoder execution time on failures for $d = 7$ rotated surface code	73
5.31 Pre-decoder execution time on failures for $d = 7$ toric code	74
5.32 Pre-decoder execution time on failures for GB codes	74
5.33 Pre-decoder execution time on 1 failure for $d = 7$ rotated surface code circuit	75

List of Tables

1.1	List of Acronyms	8
5.1	List of GB codes	53

Chapter 1

Introduction

In Ancient Greece, Thales of Miletus proposed that all things arise from water; Anaximenes argued for air; Heraclitus, for fire.

In the modern era, physicists have proposed that the universe is a system of interacting fields [1], a network of vibrating strings in higher-dimensional space [2], a computational process governed by cellular automata [3], or a quantum computer [4].

Simulating a single rock with a classical computer is practically impossible. This fact makes the existence of rocks in the universe highly suspicious, unless the universe itself possesses computational power that transcends that of a Turing machine. This idea was first suggested by Manin in 1980 [5], who proposed that quantum mechanics might enable physical models of computation more powerful than classical ones. In 1982, Feynman proposed that quantum systems could be used as computational devices to simulate other quantum systems [6]. In 1985, Deutsch formalized the concept of a universal quantum computer [7]. Since then, quantum simulation has long been considered a promising tool for accelerating

the discovery of new materials and drugs. However, in recent years, AI has significantly expanded the scale and complexity of quantum systems that can be simulated [8, 9, 10], including certain strongly correlated quantum systems [11]. This rapid progress has led to AI-based approaches becoming increasingly dominant in areas such as molecular design and material property prediction. Nevertheless, simulating many quantum systems using classical computers remains a significant challenge [12]. Recent advances have supported the promising vision of quantum simulation [13, 14, 15, 16].

While the question of whether quantum computers are hypercomputational remains open, it is well established that they offer provable advantages over classical computers in specific tasks—for instance, Grover’s algorithm [17] achieves a quadratic speedup over any classical unstructured search algorithm. Shor’s factoring algorithm [18] achieves an exponential speedup over the best-known classical algorithms for integer factorization, enabling quantum computers to break widely used RSA encryption and the elliptic curve cryptography employed in Bitcoin. While it has not been formally proven that no classical algorithm can match its efficiency, the consensus in the computational complexity community is that such a classical algorithm is unlikely to exist.

Building a practical quantum computer requires solving many challenges. The ability to maintain quantum coherence over sufficiently long timescales in physical systems is required. While there has been substantial progress in platforms such as superconducting qubits [19], trapped ions [20], neutral atoms [21, 22], and photonic systems [23], reducing physical error rates to a level comparable to those of classical computers remains far beyond

current capabilities. As a result, quantum error correction (QEC) is essential to protect quantum information against noise and decoherence.

The threshold theorem states that if the physical error rate of a physical qubit is kept below a certain threshold, it is possible to perform arbitrarily long quantum computations with arbitrarily low logical error rates by encoding logical quantum information using redundant physical qubits [24, 25, 26]. To realize this promise in practice, it is crucial to design quantum error-correcting codes that can operate effectively under realistic noise levels. Ideally, such codes should achieve a favorable balance among key properties such as high code distance, efficient encoding, locality of stabilizers, fault-tolerant logical gate implementation, and a high error threshold. Prominent candidates include topological codes, such as the surface code [27, 28] and color code [29]. Other important classes include quantum low-density parity-check (LDPC) codes [30], concatenated codes [31], and subsystem codes such as Bacon-Shor codes [32].

In each quantum error correction cycle, error syndromes are extracted by a syndrome extraction circuit, and a decoder processes the syndromes to correct errors. To prevent error accumulation and propagation, millions of such cycles are performed during the runtime of a quantum algorithm. Since non-Clifford gates are essential for universal quantum computing, real-time decoding is required, imposing strict requirements on the speed and accuracy of the decoder.

Recently, qLDPC codes have emerged as a particularly promising class of quantum codes, since they are currently the only known family of quantum codes that support fault-tolerant quantum error correction with both asymptotically constant rate and constant

overhead [33, 34]. Recent studies have demonstrated their high-threshold and low-overhead advantages over surface codes on hardware platforms including neutral atom arrays [35] and superconducting qubit architectures [36, 37]. Many decoders have been proposed for qLDPC codes. For topological codes, minimum-weight perfect matching (MWPM) [27, 38] and union-find (UF) [39] are widely used; both achieve near-linear complexity in code block length n . For general qLDPC codes, the standard approach is belief propagation (BP) followed by ordered statistics decoder (OSD) [40]. Other generic decoders include Tesseract [41] and decision tree decoders (DTDs) [42], which are maximum-likelihood (ML) decoders applicable to arbitrary qLDPC codes but incur super-linear complexity in n .

While BP achieves near-capacity performance with linear complexity for classical LDPC codes, its accuracy degrades significantly for qLDPC codes, particularly under circuit-level noise models. To address these failures, ordered statistics decoding (OSD) is commonly employed as a post-processing step. BP+OSD yields several orders of magnitude improvement in decoding accuracy across various code families, including hypergraph product codes, general bicycle codes, and topological codes [40, 43]. The limitation of OSD lies in its reliance on Gaussian elimination, with worst-case time complexity $\mathcal{O}(n^3)$. Although for qLDPC codes of fixed row weight the cost can be reduced to $\mathcal{O}(n^2)$, the decoder remains computationally expensive, since OSD is often invoked- especially for realistic circuit-level decoding, where Tanner graphs contain thousands of variable nodes. To address this, several alternative post-processing decoders have been proposed, such as Ambiguity Clustering decoder [44] and Localized Statistics decoder [45]. They achieve faster runtimes but their time complexities remain superlinear. The other post-processing decoders such as ordered

Tanner forest(OTF) [46] and closed-branch decoder (CB) [47] offer almost-linear complexity with sacrificed accuracy.

Another approach is to modify BP itself to improve its accuracy, thereby eliminating or at least reducing the need for post-processing. The earliest work in this direction was by Poulin and Chung [48], who introduced naive techniques such as freezing selected qubits and randomly perturbing prior probabilities. Subsequent methods include randomized message update schedules [49], more refined freezing techniques known as BP-guided decimation [50, 51], and other methods including MBP [52, 53], Relay-BP [54], SymBreak-BP [55]. Since these methods involve rerunning BP multiple times, a larger code size often leads to a larger number of reruns for large codes. An alternative approach proposed in [56] modifies the code structure itself, enabling BP to operate more effectively.

In this work, we conduct a detailed comparison and evaluation of various modified BP decoding methods across different code families and noise models, with a particular focus on their effectiveness under realistic circuit-level noise. For low-degeneracy codes such as generalized bicycle (GB) codes, we find that standard BP alone can already achieve decoding accuracy comparable to that of BP+OSD, leaving limited room for improvement by other methods.

Circuit-level noise models introduce high degeneracy in their Tanner graphs, causing the dual of the parity-check matrix to contain many weight-3 rows. Simulating these models in the low logical error rate regime is computationally expensive. However, we observe that rotated surface codes (with row weights 2 or 4) and toric codes (row weight 4) exhibit structural similarities to codes affected by circuit-level noise. As a result, we use

these codes as proxies in our simulations. For an optimal decoder, the log-log plot of physical error rate versus logical error rate is expected to exhibit a slope of $\lfloor \frac{d+1}{2} \rfloor$, where d is the code distance. While the modified BP methods do improve decoding accuracy, our results show that none of them achieve the optimal slope on surface or toric codes as d increases. Moreover, the extent of accuracy improvement varies across methods and is closely tied to their computational cost.

A well-known structural feature of many topological LDPC codes is that their parity checks are local, so physical errors below a certain percolation threshold probability p_c typically form only small, disconnected clusters on the lattice. Standard BP decoding, however, operates globally on the Tanner graph without directly exploiting this locality. To address these limitations, we propose a cluster-based pre-decoding strategy. By identifying and resolving small connected error clusters in advance, when combined with BP+OSD, the pre-decoder significantly decreases the number of times OSD needs to be invoked, thereby accelerating the overall decoding process while retaining competitive performance.

The pre-decoder decomposes the syndrome vector into smaller, disjoint clusters using the connectivity structure of the syndrome graph. Each cluster is then decoded independently using a pre-computed hash table, significantly reducing the decoding runtime. This approach is particularly effective in the low-error-rate regime, which is most relevant for practical fault-tolerant quantum computing.

Simulation results demonstrate that our pre-decoder can reduce decoding time by several orders of magnitude, with minimal impact on decoding accuracy in the sub-threshold

regime. These findings suggest that our method can serve as an efficient first-pass decoder or as a component in hybrid decoding architectures.

1.1 Outline of the thesis

The remainder of this dissertation is organized as follows.

Chapter 2 introduces the theoretical background necessary for this work. It reviews the basics of classical linear codes, quantum error correction, and stabilizer formalism, with emphasis on Calderbank–Shor–Steane (CSS) codes and quantum LDPC codes. We also discuss generalized bicycle codes, common error models, and several established decoding algorithms, including minimum-weight perfect matching and union–find decoders. This background sets the stage for understanding the challenges and opportunities in decoding quantum LDPC codes.

Chapter 3.2 focuses on the belief-propagation (BP) algorithm. We describe the standard BP decoding framework, message-passing rules, and the role of update schedules. The chapter also analyzes typical failure mechanisms of BP, such as error floors and trapping sets, and introduces ordered statistics decoding (OSD) as a post-processing technique.

Chapter 4 presents several modified BP-based strategies aimed at improving decoding performance. These include randomized update schedules, perturbed priors, freezing and decimation techniques, and variants of BP with memory effects. A central contribution of this work, the cluster-based pre-decoder, is also introduced in this chapter.

Chapter 5 provides a systematic comparison of these decoding methods across multiple code families (rotated surface, toric, and generalized bicycle codes) and noise models.

Simulation results are used to evaluate both the accuracy and the computational costs of different approaches. Special attention is given to the runtime advantages of the pre-decoder when combined with BP+OSD.

Finally, Chapter 6 summarizes the main findings and contributions of the thesis. It highlights the conditions under which BP refinements are most effective, the indispensable role of OSD in achieving near-optimal accuracy, and the practical benefits of a hybrid pre-decoder design. Broader implications for low-latency quantum error correction in future fault-tolerant quantum computers are also discussed.

Table 1.1: List of Acronyms

Acronym	Full Form
QEC	quantum error correction
CSS	Calderbank–Shor–Steane
LDPC	low-density parity-check
qLDPC	quantum low-density parity-check
ML	maximum-likelihood
BP	belief propagation
OSD	ordered statistics decoder
LLR	log-likelihood ratio
qllr	quantized log-likelihood ratio
MWPM	minimum-weight perfect matching
UF	union-find
EEG	error equivalence group

Chapter 2

Background

2.1 Classical Linear Binary Codes

A linear binary code with parameters $[n, k, d]$ encodes a k -bit string to a linear subspace C with dimension k of the vector space $\mathbb{F}_2^n$. The vectors in C are called codewords. The code distance d is the minimum number of elements where two different codewords differ. The weight of a codeword is the number of its non-zero elements.

A binary generator matrix G is used to encode a message $\mathbf{x}$, producing a codeword $\mathbf{y} = G^T \mathbf{x}$. The parity-check matrix H of C spans the kernel of G , satisfying $HG^T = 0$ and $\text{rank } H + \text{rank } G = n$.

After transmission through a noisy channel, some bits of the codeword may be flipped. This can be modeled by an error vector $\mathbf{e}$, resulting in a received vector $\mathbf{y}' = \mathbf{y} + \mathbf{e}$. To detect and correct errors, H is applied to the received word, yielding the syndrome

$\mathbf{s} = H\mathbf{y}' = H\mathbf{e}$. Then a decoder is applied to output an estimated error $\mathbf{e}'$. Decoding is considered successful if $\mathbf{e}' = \mathbf{e}$.

A decoder that tries to find an error $\mathbf{e}'$ that maximizes $p(\mathbf{e}' | \mathbf{s})$ is known as a maximum-likelihood (ML) decoder. Although ML decoding is computationally intractable for most codes, a variety of efficient approximate decoders have been developed for classical codes. Notable examples include the belief propagation (BP) decoder for LDPC codes [57], the Viterbi algorithm for convolutional codes [58], and successive cancellation decoders for polar codes [59], all of which achieve near-optimal performance with significantly reduced complexity.

2.2 Quantum error correction

In quantum error correction, information is stored in qubits, and noise is modeled as the action of a quantum channel $\mathcal{E}$ on a quantum state ρ , expressed as $\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger$, where $\{E_i\}$ are the Kraus operators representing possible error events. Unlike classical errors, quantum errors are continuous, however, any error operator on n qubits can be decomposed into a linear combination of n -qubit Pauli operators [60].

The n -qubit Pauli group $\mathcal{P}_n$ consists of all n -fold tensor products of Pauli matrices $\{I, X, Y, Z\}$, with overall phase factor of ± 1 or $\pm i$.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (2.1)$$

These operators either commute or anticommute with each other. The weight of a Pauli operator refers to the number of non-identity terms in the tensor product.

Since any error operator can be projected onto this discrete Pauli basis, it is sufficient for a quantum error-correcting code to correct Pauli X (bit-flip) and Z (phase-flip) errors, making it possible to map quantum codes to classical linear codes.

2.3 Stabilizer codes

An $[[n, k, d]]$ stabilizer code encodes k logical qubits into n physical qubits whose state space is the Hilbert space $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. The code is specified by an abelian subgroup $\mathcal{S} \subset \mathcal{P}_n$, known as the stabilizer group, and the codespace is defined as the joint $+1$ eigenspace of all operators in $\mathcal{S}$ [61]. Here $\mathcal{P}_n$ denotes the n -qubit Pauli group, consisting of tensor products of Pauli operators I, X, Y, Z with phases ± 1 or $\pm i$.

To detect and correct errors, the stabilizer generators are measured. These measurements yield a syndrome that depends on the error E , which is then used by a decoder to infer a likely correction E' . The decoder succeeds if E' and E differ by an element in the stabilizer group, i.e., $E' = ES_i$ for some $S_i \in \mathcal{S}$.

Certain Pauli operators, known as logical operators, commute with all stabilizer generators but do not belong to the stabilizer group itself. They act nontrivially on the encoded logical qubits while leaving the syndrome unchanged, and thus cannot be detected by stabilizer measurements. The distance d of a stabilizer code is defined as the minimum weight (i.e., number of non-identity components) of any nontrivial logical operator. Such

a stabilizer code is denoted as an $[[n, k, d]]$ code. In theory, any Pauli error of weight less than $\lfloor (d+1)/2 \rfloor$ can be corrected by a maximum-likelihood decoder.

Each n -qubit Pauli operator can be represented by a binary symplectic vector $(\mathbf{e}_X \mid \mathbf{e}_Z)$ in $\mathbb{F}_2^{2n}$, where $I \mapsto (0|0)$, $X \mapsto (1|0)$, $Z \mapsto (0|1)$, and $Y \mapsto (1|1)$. Under this mapping, the stabilizer generators form a binary parity-check matrix $H = (H_X \mid H_Z)$, and errors are similarly represented as binary vectors. The requirement that all stabilizers commute translates to the symplectic orthogonality condition:

$$H_X H_Z^T + H_Z H_X^T = 0 \quad (2.2)$$

In quantum error correction, a decoder tries to find the most likely equivalence class of errors under the stabilizer group, given a measured syndrome $\mathbf{s}$. Specifically, it aims to find a representative $\mathbf{e}'$ such that $\mathbf{e}' \simeq \mathbf{e}$, i.e., they differ by a stabilizer and act identically on the codespace.

2.4 Calderbank-Shor-Steane (CSS) codes

A Calderbank-Shor-Steane (CSS) code [62, 63] is a subclass of stabilizer codes where each stabilizer generator involves only Pauli X or only Pauli Z operators, its parity-check matrix takes the form:

$$H = \left[\begin{array}{c|c} H_X & 0 \\ \hline 0 & H_Z \end{array} \right] \quad (2.3)$$

and the commutativity of stabilizers imposes the constraint

$$H_X H_Z^T = 0.$$

The logical operators of a CSS code can similarly be described using generator matrices L_X and L_Z , satisfying the conditions:

$$L_X L_Z^T = I, \quad L_X H_Z^T = 0, \quad L_Z H_X^T = 0. \quad (2.4)$$

When an error $\mathbf{e} = (\mathbf{e}_X \mid \mathbf{e}_Z)$ occurs, the syndrome measurement yields two binary vectors: $s_Z = H_X \mathbf{e}_Z$ and $s_X = H_Z \mathbf{e}_X$. A decoder then attempts to infer error estimates $\mathbf{e}'_X$ and $\mathbf{e}'_Z$ that satisfy the same syndromes:

$$H_X \mathbf{e}'_Z = s_Z, \quad H_Z \mathbf{e}'_X = s_X. \quad (2.5)$$

Decoding is considered successful if the difference between the true error and the estimated error lies in the stabilizer group, corresponding to: $L_X(\mathbf{e}_Z + \mathbf{e}'_Z) = 0$ and $L_Z(\mathbf{e}_X + \mathbf{e}'_X) = 0$.

2.5 Multivariate Bernoulli Distribution

When the error rates of each bit are correlated, multivariate Bernoulli distribution can be used to describe the probability distribution of $\mathbf{e} \in \mathbb{F}_2^n$ [64]. This distribution is specified by the 2^n individual probabilities $p_{\mathbf{e}}$ of error vector $\mathbf{e}$:

$$\mathbb{P}(\mathbf{e}) = p_{00\dots 0}^{(1-e_1)(1-e_2)\dots(1-e_n)} p_{00\dots 01}^{(1-e_1)(1-e_2)\dots(1-e_{n-1})e_n} \dots p_{11\dots 1}^{e_1 e_2 \dots e_n}, \quad (2.6)$$

Using a normalization constant Z , the probability distribution can be expressed in the form of a Boltzmann distribution:

$$\mathbb{P}(\mathbf{e}) = \frac{1}{Z} e^{-E(\mathbf{e})}. \quad (2.7)$$

Since each variable e_i is idempotent ($e_i^l = e_i, \forall l > 0$), the corresponding energy $E(\mathbf{e}) = -\ln \mathbb{P}(\mathbf{e})$ can be expanded as a linear polynomial in the binary variables $\{e_i\}$ [64], which is a sum over all possible multi-bit combinations:

$$E(\mathbf{e}) = - \sum_{r=1}^n \sum_{j_1, j_2, \dots, j_r=1}^n \left(\prod_{i=1}^r e_{j_i} \right) f^{j_1 j_2 \dots j_r}. \quad (2.8)$$

where each term corresponds to the product of r bits $e_{j_1} e_{j_2} \dots e_{j_r}$ labeled by the indices $0 < j_1 < j_2 < \dots < j_r < n+1$, then weighted by a coefficient $f^{j_1 j_2 \dots j_r}$ that represents the energy contribution of those bits. The coefficients $f^{j_1 j_2 \dots j_r}$ are determined by the probabilities of $\mathbf{e}$ having an even or odd number of zeros in the positions indexed by $(j_1, j_2, \dots, j_r)$, calculated by:

$$f^{j_1 j_2 \dots j_r} = \log \frac{\prod p(\mathbf{e} \text{ with even number of 0s in position } j_i\text{s, other bits are 0})}{\prod p(\mathbf{e} \text{ with odd number of 0s in position } j_i\text{s, other bits are 0})} \quad (2.9)$$

For example, when $n = 4$:

$$f^{1,2} = \log \frac{p(0000)p(1100)}{p(1000)p(0100)}.$$

The coefficients $f^{j_1 j_2 \dots j_r}$ vanish if and only if the error rates of bits $e_{j_1}, e_{j_2}, \dots, e_{j_n}$ are independent element-wise.

As a result, when all bit errors are independent, the distribution is $p(\mathbf{e}) = \prod_{a=1}^n p_a^{e_a} (1 - p_a)^{1-e_a}$, where p_a is the error rate of bit e_a . In this case, $f^{j_1 j_2 \dots j_r} = 0, \forall r > 1$. The only nonzero coefficients are the first-order terms f^i , which is the log-likelihood ratio (LLR) of bit i :

$$f^i = \text{LLR}_i \equiv \log \frac{1 - p_i}{p_i}. \quad (2.10)$$

And the energy function is:

$$E(\mathbf{e}) = - \sum_{i=1}^n e_i \text{LLR}_i. \quad (2.11)$$

When the dependencies among the bit error rates are not overly complex, a coupling matrix θ can be used to approximate their correlations. Suppose there are m non-zero f coefficients, then θ is an $n \times m$ binary matrix, where the first n columns form an n -dimensional identity matrix. Each product $\prod_{i=1}^r e_{j_i}$ is replaced by the corresponding entry $[\mathbf{e}\theta]_b$, where the product is computed modulo 2, and b indexes the resulting vector $[\mathbf{e}\theta]$. Let f_b denote the coefficient for the b th term, the energy can be rewritten as [56]:

$$E(\mathbf{e}) = - \left(\sum_b f_b (-1)^{[\mathbf{e}\theta]_b} \right), \quad (2.12)$$

In this formulation, maximizing the posterior probability $p(\mathbf{e}' | \mathbf{s})$ is equivalent to minimizing the energy $E(\mathbf{e}' | \mathbf{s})$. Therefore, ML decoding is also minimum-energy decoding for classical codes.

For a quantum code with n qubits, $\theta = I_{2n}$ for independent X and Z errors. And for the depolarizing channel with error probability p :

$$\theta = \begin{pmatrix} I_n & 0 & I_n \\ 0 & I_n & I_n \end{pmatrix}, \quad e^{4f} = \frac{3(1-p)}{p}. \quad (2.13)$$

Let the parity-check matrix of a quantum code be H , and its null space be spanned by the rows of the generator matrix G , then any error $\mathbf{e}' = \mathbf{e} + \boldsymbol{\alpha}G$ is equivalent to $\mathbf{e}$, $\forall \boldsymbol{\alpha}$. An ML decoder thus seeks to maximize the total probability over the equivalence class of $\mathbf{e}$:

$$p(\mathbf{e}' \simeq \mathbf{e} | \mathbf{s}) = \sum_{\mathbf{e}' \simeq \mathbf{e}} p(\mathbf{e}' | \mathbf{s}). \quad (2.14)$$

Equivalently, this corresponds to minimizing the free energy:

$$E(\mathbf{e}' \simeq \mathbf{e}) = -\log \left(\sum_{\boldsymbol{\alpha}} e^{E(\mathbf{e} + \boldsymbol{\alpha}G)} \right) \equiv -\log Z(\boldsymbol{\alpha}). \quad (2.15)$$

2.6 LDPC codes

Low-density parity-check (LDPC) codes are a class of linear error-correcting codes defined by a sparse parity-check matrix, where small constants bound both the row and

column weights. It is possible to construct a classical LDPC code by a randomly generated parity-check matrix [65].

For quantum error-correcting codes, the orthogonality condition $H_X H_Z^\top = 0$ prevents the use of randomly constructed parity-check matrices. A systematic way to satisfy this constraint is through topological constructions: codes are defined on lattices and only interactions between nearby qubits are possible. In this picture, logical operators correspond to nontrivial homology classes of error chains. Examples include surface codes, toric codes, and color codes [66, 67, 27, 29]. Topological codes have local checks, and information is stored in global degrees of freedom. Topological codes exhibit dimensionality-dependent tradeoff bounds. In particular, for 2D topological codes one has $kd^2 < \mathcal{O}(n)$ [68], while higher-dimensional codes obey related but weaker restrictions.

Nevertheless, due to the locality of the parity checks, isolated errors remain correctable as long as they do not form extended connected structures spanning the lattice. As a result, topological codes can correct most errors of weight $\mathcal{O}(n)$, with logical failures typically arising only from errors that percolate across the code.

Another approach is to give up locality and construct a code using an algebraic ansatz, such as those for hypergraph product code [69] or Generalized Bicycle (GB) codes [70].

2.7 General bicycle codes

GB codes are defined by parity-check matrices:

$$H_X = (A, B), \quad H_Z = (B^T, A^T),$$

where A and B are $\ell \times \ell$ binary circulant matrices generated by polynomials $a(x)$ and $b(x)$ over $\mathbb{F}_2[x]$. k is determined by the greatest common divisor (GCD) of $a(x)$, $b(x)$, and $x^\ell - 1$, given by the auxiliary polynomial [40]

$$h(x) = \gcd(a(x), b(x), x^\ell - 1), \quad k = 2 \deg h(x), \quad (2.16)$$

Distance bounds for GB codes can be found in [71].

GB codes offer several advantages. Most importantly, they tend to yield relatively short codes with very good parameters compared to other constructions. In addition, they naturally admit planar realizations with straightforward qubit layouts, which is highly desirable for physical implementations. Furthermore, GB codes possess an overcomplete set of minimum-weight stabilizer generators that can enhance fault-tolerance performance, and their highly regular algebraic structure facilitates efficient encoding and decoding procedures.

2.8 Decoders for qLDPC codes

Decoding a classical linear code optimally is known to be NP-complete. For a quantum code, if one attempts to decode it by treating it as a classical code, the problem

remains NP-complete. However, this approach is suboptimal, as it fails to account for the error degeneracy. Taking degeneracy into consideration, the optimal quantum decoding problem has been shown to be $\#P$ -complete [72], which is much harder than decoding a classical code. In practice, a sub-optimal decoding algorithm is often employed to balance the computational cost and accuracy.

2.8.1 Minimum-weight perfect-matching decoder

The minimum-weight perfect-matching (MWPM) decoder can be applied to CSS codes where each qubit is connected to at most two X -checks and at most two Z -checks. Under this condition, any error can be represented as a chain on the decoding graph. The decoder then proceeds by constructing chains whose endpoints correspond to pairs of non-trivial checks (defects).

For each type of check, the decoder constructs a decoding subgraph (also called a matching graph or detector graph in the literature) in which nodes represent checks and virtual checks, while edges represent data qubits [73]. Virtual checks are introduced to connect any weight-1 qubits, ensuring that every qubit is incident on exactly two checks in the subgraph, which is necessary for the matching problem to be well-defined. All edges in the decoding graph have equal weight (commonly set to 1) under the assumption that all data qubits share the same error rates. In more general noise models with non-uniform error rates, the edge weights are defined as the log-likelihood ratios of the corresponding qubit error rates.

Given a syndrome, two syndrome subgraphs are constructed from the corresponding decoding subgraphs [74]. In each syndrome subgraph, nodes represent non-trivial syndrome checks and virtual checks. Edges connect all possible pairs of nodes, with edge weights given by the minimum-weight path between the endpoint checks in the corresponding decoding subgraph. These shortest paths can be computed using Dijkstra’s algorithm on the sparse syndrome subgraph, which for N non-trivial checks requires $O(Nn \log n)$ operations in the worst case, where n is the number of checks in the code. In this way, the decoding problem is mapped to a minimum-weight perfect-matching problem, which can be solved efficiently using Edmonds’ blossom algorithm [27].

A matching is a set of edges that do not share endpoints, and a perfect matching covers all vertices. The MWPM decoder selects the perfect matching with the minimum total weight $\sum_{(i,j) \in M} w_{ij}$. The resulting set of matched pairs corresponds to a set of correction paths, whose application removes the observed syndrome and restores the code state.

MWPM decoding with Edmonds’ blossom algorithm has a worst-case complexity of $O(N^3 \log N)$ in the number of nodes N in the syndrome graph. For practical decoding instances, the typical runtime has been observed empirically around $O(N^2)$ [74].

The current state-of-the-art implementation is provided by the `PyMatching` package, whose second version employs the sparse blossom algorithm [73]. The sparse blossom algorithm performs matching directly on the decoding graph by dynamically growing clusters around non-trivial checks and discovering only the necessary connections, thereby avoiding the explicit construction of a complete decoding graph. Empirically, this implementation achieves nearly linear runtime scaling for surface codes below the threshold.

Other approaches have also been developed to accelerate MWPM decoding. A closely related method is the Fusion Blossom decoder [75], which also applies the blossom algorithm directly on the decoding graph.

More recently, Micro Blossom [76] has been proposed as the first publicly available exact MWPM decoder with hardware acceleration. Building on the ideas of Sparse Blossom and Fusion Blossom, Micro Blossom implements the blossom algorithm in a hardware-friendly manner and supports stream decoding. A prototype FPGA implementation demonstrates sub-microsecond latency.

In fault-tolerant operation with measurement noise, checks are repeated and defects live in space–time. The decoder matches detection events both across space and across rounds (time), again minimizing a sum of edge weights [77].

2.8.2 Union-find decoder

Similar to the MWPM decoder, the Union-Find (UF) decoder also reformulates the decoding task as a graph problem [39]. The UF decoder grows clusters around non-trivial syndrome checks and converts Pauli errors to erasure errors, which can then be corrected using a peeling decoder [78]. A qubit erasure is an error whose location is known: the affected qubit is assumed to undergo one of the Pauli operators I , X , Y , or Z with equal probability.

The UF decoder operates on the same decoding graph as the MWPM decoder. For each non-trivial check, it initializes a cluster containing that check together with its neighboring qubits. The clusters are then expanded iteratively by incorporating adjacent checks. Whenever two clusters overlap, they are merged into a single cluster. A cluster

stops growing (i.e., is frozen) under either of the following conditions: (i) it contains an even number of non-trivial checks, or (ii) it includes a virtual qubit.

To apply the peeling decoder, each cluster must first be a spanning tree, which is a tree that includes all vertices of the cluster. For each cluster, one spanning tree is chosen, and one of its check leaves is chosen as the root. The decoding starts by selecting a non-root leaf for each cluster, then proceeds by iteratively removing non-root leaves according to the following rules:

- **If the leaf is a non-trivial check:** the edge incident to it is recorded as part of the matching (corresponding to a decoded non-trivial Pauli error). The adjacent vertex is flipped (trivial $\leftrightarrow$ non-trivial), and both the leaf and its connecting edge are removed from the spanning tree.
- **If the leaf is a trivial check:** the leaf and its incident edge are simply removed from the spanning tree.

This peeling process continues from the leaves inward until only the root vertex remains, at which point the decoding is complete.

The worst-case runtime complexity of the UF decoder is almost linear, much lower than that of the MWPM decoder. However, this gain comes at the cost of reduced decoding accuracy compared to MWPM. From a theoretical perspective, the UF decoder can be interpreted as an approximate realization of the blossom algorithm employed in MWPM [79].

2.9 Error models

The threshold and accuracy of QEC codes depend on the underlying noise model [80]. In this work, we consider three commonly used models :

1. Code capacity noise model.

This model assumes perfect measurements, requiring only a single round of syndrome extraction. One choice is applying independent single-qubit Pauli errors to the data qubits with physical error rate p . Another widely used model is the depolarizing channel, which models the loss of quantum information by replacing the state of a qubit with a completely mixed state with probability p :

$$\mathcal{E}(\rho) = (1 - p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z),$$

where ρ is the input density matrix.

In our simulations, we focus on pure phase-flip errors, i.e., independent Pauli Z errors applied to each data qubit with probability p :

$$\mathcal{E}(\rho) = (1 - p)\rho + pZ\rho Z,$$

2. Phenomenological noise model.

The phenomenological noise model accounts for errors on both data qubits and syndrome measurements, necessitating multiple rounds of measurements. In our simulations, we consider a CSS code defined by parity-check matrices $[H_X]_{r \times n}$ and $[H_Z]_{r' \times n}$.

Suppose m rounds of syndrome measurements are performed. Assume independent Pauli Z errors on data qubits with probability p , as well as independent measurement (syndrome bit-flip) errors with the same probability p . It is further assumed that the final measurement round contains no error.

Let $\mathbf{e}_j \in \mathbb{F}_2^n$ denote the data qubit Z error and $\boldsymbol{\epsilon}_j \in \mathbb{F}_2^r$ denote the syndrome error occurred in the j th round (both represented as column vectors). Assume that the final measurement round contains no error. Then the syndrome measured in the j th round is:

$$\mathbf{s}_j = H_X(\mathbf{e}_1 + \mathbf{e}_2 + \dots + \mathbf{e}_j) + \boldsymbol{\epsilon}_j, \quad \boldsymbol{\epsilon}_m \equiv \mathbf{0},$$

and the errors can be detected locally by the difference

$$\sigma_j \equiv \mathbf{s}_j - \mathbf{s}_{j-1} = H_X \mathbf{e}_j + \boldsymbol{\epsilon}_j - \boldsymbol{\epsilon}_{j-1}, \quad \boldsymbol{\epsilon}_0 \equiv \mathbf{0}.$$

It is convenient to combine data errors and measurement errors in two separate blocks,

$$\mathbf{e}^T = (\mathbf{e}_1^T, \mathbf{e}_2^T, \dots, \mathbf{e}_m^T | \boldsymbol{\epsilon}_1^T, \boldsymbol{\epsilon}_2^T, \dots, \boldsymbol{\epsilon}_{m-1}^T) \in \mathbb{F}_2^{mn+(m-1)r}. \quad (2.17)$$

The combined vector $\sigma^T = (\sigma_1, \sigma_2, \dots, \sigma_m)$ of triggered detectors can be computed using the check matrix

$$\mathbf{H}_X = (I_m \otimes H_X | R_m^T \otimes I_r), \quad \sigma = \mathbf{H}_X \mathbf{e}. \quad (2.18)$$

where R_m is a check matrix of the classical repetition code with distance m ; it has the dimensions $(m-1) \times m$ and the elements $[R_m]_{ij} = \delta_{i,j} + \delta_{i,j-1}$.

The goal of the decoder is to identify an error vector $\mathbf{e}'_\Sigma$ that is equivalent to the accumulated error after N rounds $\mathbf{e}_\Sigma = (\mathbf{e}_1 + \mathbf{e}_2 + \dots + \mathbf{e}_m)$, satisfying $L_X(\mathbf{e}_\Sigma + \mathbf{e}'_\Sigma) = 0$. This condition can also be formulated in terms of a combined error vector $\mathbf{e}'$ such that $\mathbf{L}_X(\mathbf{e} + \mathbf{e}') = 0$ where $\mathbf{L}_X$ is defined as:

$$\mathbf{L}_X = (\mathbf{1}_m \otimes L_X | 0_{k \times (m-1)r}). \quad (2.19)$$

Here $0_{a \times b}$ is an $a \times b$ zero matrix, $\mathbf{1}_m \equiv (1, 1, \dots, 1)$ is the unique non-zero codeword of the repetition code.

$\mathbf{H}_Z$ is chosen to span the null space of

$$\mathbf{G}_X = \begin{pmatrix} \mathbf{H}_X \\ \mathbf{L}_X \end{pmatrix}, \quad (2.20)$$

and is therefore defined as:

$$\mathbf{H}_Z = \left(\begin{array}{c|c} R_m \otimes I_n & I_{m-1} \otimes H_X^T \\ \hline I_m \otimes H_Z & 0 \end{array} \right), \quad (2.21)$$

Similarly, the logical operator matrix $\mathbf{L}_Z$ is selected such that $\mathbf{G}_Z = (\mathbf{H}_Z^T | \mathbf{L}_Z^T)^T$ spans the null space of $\mathbf{H}_X$, and thus $\mathbf{L}_Z$ is defined as:

$$\mathbf{L}_Z = (L_Z, 0_{k \times (m-1)n} | 0_{k \times (m-1)r}). \quad (2.22)$$

As a result, the constructed CSS code has block length $N = mn + (m - 1)r$ and encodes $K = k$ logical qubits. In this construction, the relevant distance parameter is D_Z , which equals the d_Z of the original small code.

3. Circuit-level noise model.

The circuit-level noise model reflects errors in realistic error correction circuits. In our simulations, we follow a "standard" circuit-level depolarizing noise model: i.i.d. single-qubit Pauli X errors are applied after each qubit reset and before each measurement, as well as 1- and 2-qubit depolarizing errors are introduced after each 1- and 2-qubit gate, respectively, including trivial gates (idling) with probability p .

The measurement circuits were analyzed and simulated using the `Stim` package [81], which compiles circuits into detector error models (DEMs). In a DEM, detectors are added to the circuit, each defined as a set of measurements. The parity (i.e., the sum modulo 2) of those measurement results is a value called "detection event", which forms the syndrome vector $\mathbf{s}$. In the absence of noise, these detection events are deterministic. Most detectors in an error correction circuit are constructed from two measurements of the same stabilizer in consecutive rounds. As a result, sampling

detection events offers better asymptotic complexity than directly sampling measurement outcomes [81]. This is because a single error on a qubit may affect many future stabilizer measurements involving that qubit, but it influences only two nearby detectors. Hence, non-local error effects on measurements are transformed into local effects on detection events. Moreover, sampling detection events is computationally more efficient in the low-error-rate regime.

There are many equivalent errors in a circuit; selecting one representative from each equivalence class defines the error equivalence group (EEG) of the circuit. As a result, the corresponding Tanner graph represents a highly degenerate code, leading to many weight-3 rows in the dual of the constructed binary parity-check matrix $\mathbf{H}$, where rows correspond to detectors and columns to merged error locations. A binary observable matrix $\mathbf{L}$ is formed from logical operator matrices L_X and L_Z of the original code. Usually $\text{rank } \mathbf{H} + \text{rank } \mathbf{L}$ is smaller than the number of error positions (i.e. columns of those matrices). Errors can then be represented by a binary vector $\mathbf{e}$, where each component e_i is an independent Bernoulli distribution random variable with probability p_i . The goal of the decoder is to solve the linear system $\mathbf{s} = \mathbf{H}\mathbf{e}$ with an output $\mathbf{e}'$ such that $\mathbf{L}(\mathbf{e} + \mathbf{e}') = 0$.

The circuits for the rotated surface code are taken directly from the built-in circuits provided by `Stim`. For toric codes, the circuits are adapted from those of the rotated surface code by modifying boundary conditions accordingly.

2.10 Use matrix transformations to simplify the error model

For highly degenerate quantum codes such as rotated surface codes, BP decoding can fail even on some weight-1 errors. To address this issue, degeneracy-reducing transformations have been proposed in Ref. [56]. These transformations reduce the degeneracy of a quantum code while preserving its maximum-likelihood decoding performance. These correspond to inverse decoration or star-triangle (more generally, star-polygon) transformations in the equivalent Ising models. Specifically, the inverse decoration combines two identical columns in the matrices H_X and L_X to remove a weight-2 row in H_X . The star-triangle transformation removes weight-3 rows, and more general star-polygon transformations can eliminate higher-weight rows. However, removing degeneracy vectors of weight $\omega > 2$ increases the size of H_X , which leads to more decoding overhead.

The row-reduction transformation can be described in terms of the parity-check matrix $\mathbf{H}$ and its dual generator matrix $\mathbf{G}$. To eliminate a row of $\mathbf{G}$ with weight w , the following steps are performed:

- (i) Suppose this row of $\mathbf{G}$ has ones in positions $i_1, i_2, \dots, i_w$, and zeros elsewhere. Add the i_1 -th column of $\mathbf{G}$ to columns $i_2, i_3, \dots, i_w$; as a result, the row has a non-zero entry only in the i_1 -th column.
- (ii) Drop the i_1 -th column and this row from the modified $\mathbf{G}$.
- (iii) If $w > 2$, all possible linear combinations of the columns $i_2, i_3, \dots, i_w$ are added to $\mathbf{G}$, so $\mathbf{G}$ gets $2^{w-1} - w$ more columns. These combinations can be added one by one such that each newly added column is the sum of two existing columns in the modified $\mathbf{G}$.

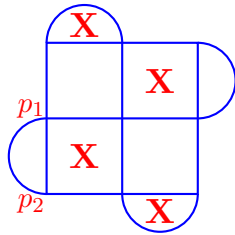
The corresponding transformation steps for $\mathbf{H}$ are as follows:

- (i') Add columns $i_2, i_3, \dots, i_w$ to the i_1 -th column of $\mathbf{H}$, making it identically zero.
- (ii') Drop the all-zero i_1 -th column from the modified $\mathbf{H}$.
- (iii') For each new column b added to $\mathbf{G}$ as the sum of two existing columns b_1 and b_2 , add a new row to $\mathbf{H}$ with support on $\{b_1, b_2, b\}$ to represent this linear dependency.

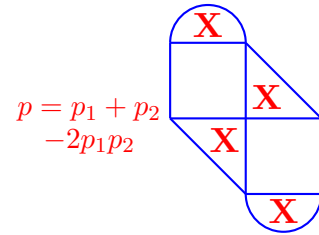
The error rates associated with each bit or qubit must be updated consistently under this transformation.

As a result, the rank of $\mathbf{G}$ is reduced by one. Meanwhile, the rank increase in $\mathbf{H}$ equals the number of columns added in step (iii'), thereby preserving the exact duality.

Prior to our simulations on rotated surface codes, we apply inverse decoration transformations to eliminate weight-2 rows. This preprocessing step ensures that BP decoding can successfully correct all weight-1 errors on the transformed code.



(a) $d = 3$ rotated surface codes X stabilizer generators



(b) $d = 3$ rotated surface codes X stabilizers after transformation

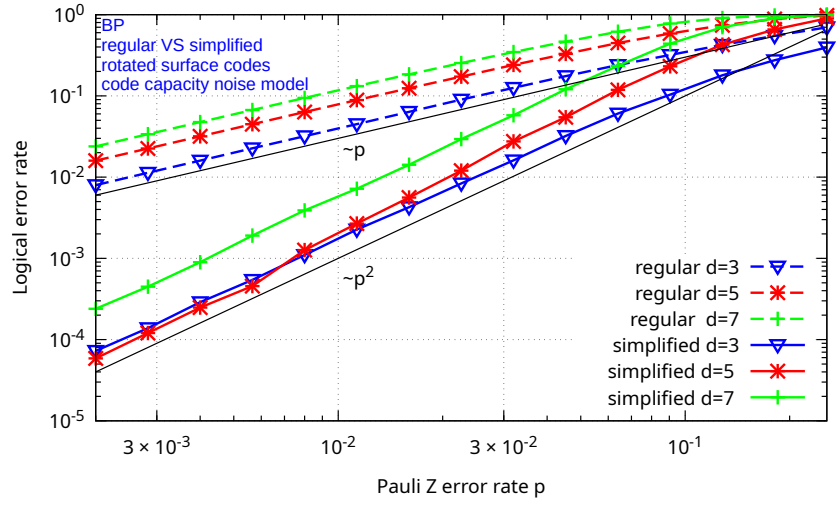


Figure 2.2: BP decoding accuracy for simplified and regular rotated surface codes.

Chapter 3

Belief Propagation

3.1 BP algorithm

Belief propagation (BP) is an iterative algorithm for estimating the marginal distributions of a joint probability distribution:

$$p(x_a) = \sum_{x_1, \dots, x_{a-1}, x_{a+1}, \dots, x_n} p(\mathbf{x}). \quad (3.1)$$

Here, the joint distribution $p(\mathbf{x})$ is defined over n variables $\mathbf{x} = (x_1, \dots, x_n)$, where each variable takes values in a finite alphabet χ . Direct marginal computation requires exponential time complexity $\mathcal{O}(|\chi|^n)$, but when the corresponding Tanner graph is a tree, BP can estimate marginals with linear complexity $\mathcal{O}(n)$.

Assume the distribution factorizes as follows:

$$p(\mathbf{x}) = \prod_{i=1}^m f_i(\mathbf{x}_{\partial f_i}). \quad (3.2)$$

where $\mathbf{x}_{\partial_{f_i}}$ is a subset of variables on which the factor f_i depends, and ∂_{f_i} is the set of their indices. Similarly, $f_{\partial_{x_a}}$ is a subset of factors that depend on the variable x_a , and ∂_{x_a} is the set of their indices.

A Tanner graph is a bipartite graph that represents the factorization structure of a distribution. It consists of two types of nodes: variable nodes $\{x_a\}$ corresponding to the variables and factor nodes $\{f_i\}$ corresponding to the factor functions. An edge connects a variable node x_a to a factor node f_i if and only if $x_a \in \mathbf{x}_{\partial_{f_i}}$.

In belief propagation, two messages are associated with each edge in the Tanner graph. One message is sent from a variable node to a factor node, $\mathbf{m}_{x_a \rightarrow f_i}(x_a)$, and the other is sent in the opposite direction, from a factor node to a variable node, $\mathbf{m}_{f_i \rightarrow x_a}(x_a)$. Each message can be represented by a vector of dimension χ , labeled by the value of x_a . By convention, these messages are initialized to $\frac{1}{|\chi|}$. This choice does not affect the result of the BP algorithm, as it only influences the normalization factor.

The messages are then updated iteratively according to the following rules, at iteration t :

$$\mathbf{m}_{x_a \rightarrow f_i}^{(t)}(x_a) = \prod_{j \in \partial_{x_a} \setminus i} \mathbf{m}_{f_j \rightarrow x_a}^{(t-1)}(x_a), \quad (3.3)$$

$$\mathbf{m}_{f_i \rightarrow x_a}^{(t)}(x_a) = \sum_{\mathbf{x}_{\partial_{f_i} \setminus x_i}} f_i(\mathbf{x}_{\partial_{f_i}}) \prod_{b \in \partial_{f_i} \setminus a} \mathbf{m}_{x_b \rightarrow f_i}^{(t-1)}(x_b), \quad (3.4)$$

Where $\partial_{x_a} \setminus i$ denotes removing an element i from the set ∂_{x_a} , i.e. $\partial_{x_a} \setminus \{i\}$.

After t iterations, the marginal of x_a is estimated by:

$$p(x_a) = \frac{1}{Z} \prod_{a \in \partial x_a} \mathbf{m}_{f_i \rightarrow x_a}^{(t)}(x_a), \quad (3.5)$$

with the partition function Z ensuring normalization.

BP computes the exact marginals when the Tanner graph is a tree. Otherwise, it often remains highly effective, which can be attributed to the fact that BP works well when the Tanner graph is locally a tree, which is often the case in error-correcting codes.

3.2 BP Decoder

In classical decoding, given a parity-check matrix H and a syndrome $\mathbf{s} = H\mathbf{e}^T$, the goal is to infer the most likely error vector $\mathbf{e}$. Assuming independent bit errors, the probability distribution $p(\mathbf{e})$ is:

$$p(\mathbf{e}) = \prod_{a=1}^n p_a^{e_a} (1 - p_a)^{1-e_a}. \quad (3.6)$$

And BP operates on the conditional distribution:

$$p(\mathbf{e} \mid \mathbf{s}) = \frac{1}{Z} \prod_{a=1}^n p_a^{e_a} (1 - p_a)^{1-e_a} \prod_{i=1}^r \mathbf{1}(H_i \mathbf{e} = s_i \pmod{2}), \quad (3.7)$$

where H_i is the i -th row of H , r is the number of rows of the matrix H , and $\mathbf{1}(\cdot)$ is the indicator function.

A Tanner graph is constructed where:

- Variable nodes: $v_a \in \{0, 1\}$, representing the error bits e_a ;
- Prior factor nodes p_a : for $p_a^{e_a}(1 - p_a)^{1-e_a}$, fixed throughout iterations;
- Check factor nodes c_i : for parity-check constraints $\mathbf{1}(H_i \mathbf{e} = s_i)$.

Since messages from p_a to v_a remain fixed, only messages passing between variable nodes and check nodes are considered.

For binary codes, the messages can be represented in terms of scalars. Define

$r_{v_a \rightarrow c_i} = \frac{\mathbf{m}_{v_a \rightarrow c_i}(0)}{\mathbf{m}_{v_a \rightarrow c_i}(1)}$ and $r_{c_i \rightarrow v_a} = \frac{\mathbf{m}_{c_i \rightarrow v_a}(0)}{\mathbf{m}_{c_i \rightarrow v_a}(1)}$, then the updating rules can be simplified to

$$r_{v_a \rightarrow c_i} = \frac{1 - p_a}{p_a} \prod_{j \in \partial_{v_a} \setminus i} r_{c_j \rightarrow v_a}. \quad (3.8)$$

And

$$\begin{aligned} r_{c_i \rightarrow v_a} &= \frac{\sum_{b \in \partial_{c_i} \setminus a} [\mathbf{1}(\sum_b e_b = s_i \mod 2) \prod_b \mathbf{m}_{v_b \rightarrow c_i}(e_b)]}{\sum_{b \in \partial_{c_i} \setminus a} [\mathbf{1}(\sum_b e_b = s_i + 1 \mod 2) \prod_b \mathbf{m}_{v_b \rightarrow c_i}(e_b)]} \\ &= \frac{\sum_{b \in \partial_{c_i} \setminus a} [\mathbf{1}(\sum_b e_b = s_i \mod 2) \prod_b \frac{\mathbf{m}_{v_b \rightarrow c_i}(e_b)}{\mathbf{m}_{v_b \rightarrow c_i}(1)}]}{\sum_{b \in \partial_{c_i} \setminus a} [\mathbf{1}(\sum_b e_b = s_i + 1 \mod 2) \prod_b \frac{\mathbf{m}_{v_b \rightarrow c_i}(e_b)}{\mathbf{m}_{v_b \rightarrow c_i}(1)}]} \end{aligned} \quad (3.9)$$

In the term $\prod_b \frac{\mathbf{m}_{v_b \rightarrow c_i}(e_b)}{\mathbf{m}_{v_b \rightarrow c_i}(1)}$, e_b can take values of 0 or 1, however, regardless of its value, $\prod_b \frac{\mathbf{m}_{v_b \rightarrow c_i}(e_b)}{\mathbf{m}_{v_b \rightarrow c_i}(1)} = \prod_b r_{v_b \rightarrow c_i}^{1-e_b}$. Here, the superscript $1 - e_b$ serves as the exponent.

Then apply the formula [82]

$$\begin{aligned}
& 2\left(\sum_{b \in \partial_{c_i} \setminus a} \mathbf{1}(\sum_b e_b = 0 \pmod{2}) \prod_b r_{v_b \rightarrow c_i}^{1-e_b}\right) = \\
& \prod_{b \in \partial_{c_i} \setminus a} (r_{v_b \rightarrow c_i} + 1) + \prod_{b \in \partial_{c_i} \setminus a} (r_{v_b \rightarrow c_i} - 1)
\end{aligned} \tag{3.10}$$

and

$$\begin{aligned}
& 2\left(\sum_{b \in \partial_{c_i} \setminus a} \mathbf{1}(\sum_b e_b = 1 \pmod{2}) \prod_b r_{v_b \rightarrow c_i}^{1-e_b}\right) = \\
& \prod_{b \in \partial_{c_i} \setminus a} (r_{v_b \rightarrow c_i} + 1) - \prod_{b \in \partial_{c_i} \setminus a} (r_{v_b \rightarrow c_i} - 1)
\end{aligned} \tag{3.11}$$

The equation 3.9 becomes:

$$\begin{aligned}
r_{c_i \rightarrow v_a} &= \left(\frac{\prod_{b \in \partial_{c_i} \setminus a} (r_{v_b \rightarrow c_i} + 1) + \prod_{b \in \partial_{c_i} \setminus a} (r_{v_b \rightarrow c_i} - 1)}{\prod_{b \in \partial_{c_i} \setminus a} (r_{v_b \rightarrow c_i} + 1) - \prod_{b \in \partial_{c_i} \setminus a} (r_{v_b \rightarrow c_i} - 1)} \right)^{1-2s_i} \\
&= \left(\frac{1 + \prod_{b \in \partial_{c_i} \setminus a} \frac{(r_{v_b \rightarrow c_i} - 1)}{(r_{v_b \rightarrow c_i} + 1)}}{1 - \prod_{b \in \partial_{c_i} \setminus a} \frac{(r_{v_b \rightarrow c_i} - 1)}{(r_{v_b \rightarrow c_i} + 1)}} \right)^{1-2s_i}.
\end{aligned} \tag{3.12}$$

Define the log-likelihood ratio $L_{c_i \rightarrow v_a} = \log(r_{c_i \rightarrow v_a})$, then $\frac{r_{e_l \rightarrow c_i} - 1}{r_{e_l \rightarrow c_i} + 1} = \tanh(L_{c_i \rightarrow v_a}/2)$,

combine with $\tanh^{-1}(x) = \frac{1}{2} \log\left(\frac{1+x}{1-x}\right)$, the updating rules are

$$\begin{aligned}
L_{c_i \rightarrow v_a}^{(t)} &= (1 - 2s_i) 2 \tanh^{-1} \left(\prod_{b \in \partial_{c_i} \setminus a} \tanh(L_{v_b \rightarrow c_i}^{(t-1)}/2) \right) \\
&= (-1)^{s_i} 2 \tanh^{-1} \left(\prod_{b \in \partial_{c_i} \setminus a} \tanh(L_{v_b \rightarrow c_i}^{(t-1)}/2) \right),
\end{aligned} \tag{3.13}$$

$$L_{v_a \rightarrow c_i}^{(t)} = \log \left(\frac{1 - p_a}{p_a} \right) + \sum_{j \in \partial_{v_a} \setminus i} L_{c_j \rightarrow v_a}^{(t-1)}. \tag{3.14}$$

After each iteration, LLR at each variable node is calculated by:

$$\text{LLR}(v_a) = \log \left(\frac{1 - p_a}{p_a} \right) + \sum_{j \in \partial_{v_a}} L_{c_j \rightarrow v_a} = \log \left(\frac{p(e_a = 0)}{p(e_a = 1)} \right). \tag{3.15}$$

A hard decision is made as: $e'_a = 1$ if $\text{LLR}(v_a) < 0$ and $e'_a = 0$ if $\text{LLR}(v_a) \geq 0$. If the estimated error vector $\mathbf{e}'$ satisfies the syndrome condition $H\mathbf{e}' = \mathbf{s}$, decoding terminates. If $\mathbf{e}' = \mathbf{e}$, decoding is successful. If no valid $\mathbf{e}'$ is found after the maximum number of iterations, decoding is considered failed.

For quantum codes, decoding is performed on a similar binary check matrix H , with an observable matrix L (in the case of CSS code, that is the corresponding logical matrix). The goal is to find the likely error degeneracy class. Decoding is considered successful if the condition $L(\mathbf{e} + \mathbf{e}') = 0 \pmod{2}$ is satisfied.

In practice, the equation 3.13 is often rewritten using the box-plus operator $\boxplus$, defined as:

$$x \boxplus y \equiv \log \frac{1 + e^{x+y}}{e^x + e^y} = 2 \tanh^{-1} \left(\tanh \frac{x}{2} \tanh \frac{y}{2} \right). \tag{3.16}$$

The box-plus operator satisfies:

$$x_1 \boxplus x_2 \boxplus \cdots \boxplus x_n = 2 \tanh^{-1} \left(\prod_{i=1}^n \tanh \frac{x_i}{2} \right). \quad (3.17)$$

Accordingly, equation 3.13 becomes

$$L_{c_i \rightarrow v_a}^{(t)} = (-1)^{s_i} \boxplus_{b \in \partial_{c_i} \setminus a} (L_{v_b \rightarrow c_i}^{(t-1)} / 2). \quad (3.18)$$

In a practical implementation of BP, the box-plus operation is often calculated by:

$$x \boxplus y = \text{sign}(x) \text{sign}(y) \cdot \min(|x|, |y|) + \log \left(1 + e^{-|x+y|} \right) - \log \left(1 + e^{-|x-y|} \right). \quad (3.19)$$

This expression is often preferred due to its numerical stability and efficiency. The hyperbolic tangent expression can suffer from loss of precision when $\tanh(x/2)$ approaches ± 1 .

A more efficient but less accurate approximation is the min-sum operation:

$$x \boxplus y \approx \text{sign}(x) \text{sign}(y) \cdot \min(|x|, |y|) = \frac{|x+y| - |x-y|}{2}. \quad (3.20)$$

In a practical implementation, the messages are usually quantized by the datatype **qLLR** (quantized log-likelihood ratio). Here the exact LLR value of type 'double' is approximated as

$$(\text{double}) \text{LLR} = (2^{-\text{Dint1}}) \times (\text{int}) \text{qLLR}.$$

The parameter `Dint1` controls the quantization scale and determines the resolution of the integer representation. Then a lookup table with `Dint2` entries is used to accelerate box-plus operation. The resolution of this table is given by

$$\text{resolution} = 2^{-(\text{Dint1}-\text{Dint3})},$$

where `Dint3` controls the interpolation granularity. This integer-based scheme enables efficient and fast computation at the cost of reduced numerical precision.

To capture correlations between errors, BP with a larger alphabet can be employed [83, 53]. However, in this work, we restrict ourselves to binary BP, as both the parity-check matrix `H` and the logical operator matrix `L` generated by `Stim` are binary.

As a part of the decoder implementation, all numerical computations involving $\boxplus$ are performed using the hyperbolic tangent formulation and `double` precision. To ensure numerical stability when the argument of the hyperbolic tangent function becomes too large, we clip the absolute value of LLRs to a maximum of 20 in the implementation.

3.3 BP schedules

The sequence in which messages are updated during an iteration is termed the schedule or updating rule. In cycle-free Tanner graphs, convergence is guaranteed within $D/2$ iterations, where D is the length of the longest path in the graph. In such cases, an optimal schedule exists that requires exactly $2|E|$ message updates, where $|E|$ is the number of edges in the graph.

The most straightforward update scheme is the parallel (or flooding) schedule, in which all messages are updated simultaneously based on those from the previous iteration, as shown in equations (3.13) and (3.14). This is equivalent to first updating all messages from check nodes to variable nodes, followed by updating all messages from variable nodes to check nodes, using messages from the current iteration.

$$L_{c_i \rightarrow v_a}^{(t)} = (-1)^{s_i} \boxplus_{b \in \partial_{c_i} \setminus a} (L_{v_b \rightarrow c_i}^{(t)} / 2). \quad (3.21)$$

$$L_{v_a \rightarrow c_i}^{(t)} = \log\left(\frac{1 - p_a}{p_a}\right) + \sum_{j \in \partial_{v_a} \setminus i} L_{c_j \rightarrow v_a}^{(t)}. \quad (3.22)$$

For a cycle-free graph, the parallel schedule achieves convergence exactly after $D/2$ iterations.

Serial schedules, also known as 'sequential' schedules, tend to promote faster and sometimes better accuracy by updating messages based on a certain order of nodes. Goldberger and Kfir [84] suggest two types of serial schedules: serial-V [85] and serial-C [86].

In the serial-V schedule, variable nodes are processed sequentially. For each variable node v_a , all incoming messages $L_{c_i \rightarrow v_a}$ are first updated, followed by the update of all outgoing messages $L_{v_a \rightarrow c_i}$. The process then moves to the next variable node v_{a+1} . In the serial-C schedule, check nodes are processed sequentially. For each check node c_i , all incoming messages $L_{v_a \rightarrow c_i}$ are updated first, followed by the update of all outgoing messages $L_{c_i \rightarrow v_a}$. This continues node-by-node until all check nodes have been updated.

3.4 Error Floors and Trapping sets

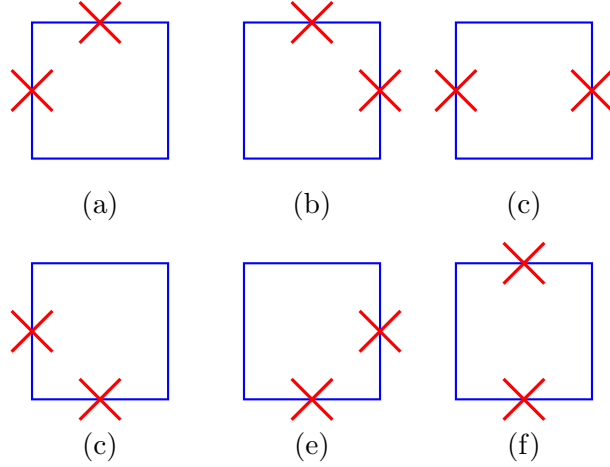
BP decoding is known to suffer from an early error floor when applied to qLDPC codes, such as toric codes and surface codes [48]. The error floor refers to the phenomenon where the logical error rate decreases more slowly once it falls below a certain threshold [87]. This is primarily due to BP’s inability to correct certain low-weight errors.

In classical LDPC decoding, this behavior is attributed to structures known as trapping sets [87], which are subsets of variable nodes that the decoder fails to correct. The presence of trapping sets depends on both the Tanner graph structure and the behavior of the decoding algorithm. For the binary erasure channel, all the trapping sets are stopping sets. A stopping set $\mathcal{S}$ is a subset of variable nodes such that all their neighboring check nodes are connected to $\mathcal{S}$ at least twice [88]. For the additive white Gaussian noise (AWGN) channel, absorbing sets describe the dominant decoding failures [89]. An (a, b) absorbing set is a subgraph induced by a variable nodes, and b is the number of check nodes in the subgraph with odd degrees. In this subgraph, each variable node is connected to strictly more even-degree check nodes than odd-degree check nodes. Other harmful structures include near-codewords [90] and pseudocodewords [91].

In qLDPC decoding, the commutativity constraint $H_X H_Z^T = 0$ leads to an increased number of short cycles in the Tanner graph, which introduces more trapping sets. Moreover, even when the Tanner graph is cycle-free—for instance, in the H_X Tanner graph for $d = 3$ rotated surface code, BP still fails to decode degenerate errors that have the same weight (or energy). Such highly degenerate errors are especially prevalent in Tanner graphs corresponding to circuit error models, leading to poor BP performance. These two

mechanisms—short cycles and degeneracy—contribute to the failures of BP and give rise to quantum trapping sets [92].

For example, when using the parallel schedule, BP fails to correct certain weight-2 errors that fall into symmetric configurations, as illustrated in the following patterns:



For a toric code with distance d , there are $4d^2 + 2d^2 = 6d^2$ such weight-2 errors that the parallel-schedule BP decoder fails to correct.

Applying the serial-C schedule successfully decodes pattern (c) and pattern (f), but still fails on others. The serial-V schedule performs even better when a large number of BP iterations is allowed.

3.5 Ordered Statistics Decoder

To improve the decoding accuracy of BP, a standard approach is to apply the ordered statistics decoder (OSD) as a post-processing step [93, 40].

When BP decoding fails, OSD uses the LLRs output by BP to construct a solution. The simplest version of OSD, known as OSD-0, is illustrated by the following procedure:

1. Upon BP failure:
 - Sort the LLRs in ascending order, and apply the corresponding column permutation matrix P to the parity-check matrix $H: H \rightarrow HP$
2. Use row transformations to transform HP into row-echelon form $\tilde{H}$, ensuring that the first $\text{Rank}(HP)$ columns are linearly independent; if necessary, additional column permutations may be applied to move linearly independent columns to the leading positions.
3. Partition the matrix as $\tilde{H} = (H_S \mid H_T)$, where H_S is a square invertible matrix. The permuted error vector is correspondingly partitioned as $P^T \mathbf{e} = \begin{pmatrix} \mathbf{e}_S \\ \mathbf{e}_T \end{pmatrix}$, and a candidate solution (known as OSD-0) is obtained by:

$$\mathbf{e}_S = H_S^{-1} \mathbf{s}, \quad \mathbf{e}_T = 0.$$

4. Reorder the error to its original sequence: $\mathbf{e} = P \begin{pmatrix} \mathbf{e}_S \\ 0 \end{pmatrix}$.

Higher order OSD involves different choices of $\mathbf{e}_T$. For any such $\mathbf{e}_T$, the following vector also satisfies the syndrome constraint:

$$P^T \mathbf{e} = \begin{pmatrix} \mathbf{e}_S + H_S^{-1} H_T \mathbf{e}_T \\ \mathbf{e}_T \end{pmatrix}.$$

A greedy search can be performed over $\mathbf{e}_T$ to find a solution of minimum weight. In our simulation, we adopt the combination sweep strategy [43], which searches over all

weight-1 configurations of $\mathbf{e}_T$ and all weight-2 configurations in the first λ bits of $\mathbf{e}_T$. For all the simulations in this work, we set $\lambda = 100$.

Since the OSD algorithm relies on Gaussian elimination, its worst-case time complexity is $\mathcal{O}(n^3)$, where n is the number of columns in H . For the LDPC code families whose parity-check matrices have constant row-weight, the time complexity can be reduced to $\mathcal{O}(n^2)$, however, the computational cost remains high in practice.

To better understand the scaling, consider the probability that OSD is triggered. For a code with M weight-two errors that BP fails to decode, the triggering probability is approximately Mp^2 at a small error rate p . More generally, for stabilizer generators of weight $w = 2m$, there are typically errors of weight $w/2 = m$ that lead to OSD activation with probability of order np^m , where n is the block length. This explains why toric codes under the phenomenological error model ($m = 3$) and GB codes ($m \geq 3$) tend to perform better [40, 48]. The overall expected complexity thus scales as

$$\mathcal{O}(n) + \mathcal{O}(Np^m) \cdot \mathcal{O}(n^3).$$

Under circuit-level noise, however, one always has $w = 3$ and $m = 2$ (provided that code transformations are applied to eliminate identical columns from H), so the probability of triggering OSD is significantly larger in this setting.

Chapter 4

Modified BP decoders

4.1 Random schedules

As discussed earlier, fixed-order schedules are unable to resolve all decoding failures associated with low-weight symmetric errors. To address this, Du et al. [49] proposed a random serial schedule. In this scheme, during each BP iteration, the update order of nodes is randomly permuted. After completing updates involving one node, the decoder randomly selects the next unvisited node and performs its message updates.

In particular, the random serial-V schedule updates messages associated with one variable node at a time. At each step of an iteration, a variable node v_a that has not yet been updated in the current iteration is randomly selected. All outgoing messages from v_a to its neighboring check nodes c_i are then updated. This process continues until all variable nodes have been updated once.

Simulation results show that the random serial-V schedule enables BP to successfully decode all weight-2 errors in toric codes. However, the improvement is limited: weight-3 errors can still cause decoding failures, indicating that the random schedule alone is insufficient for full error correction in more challenging cases.

4.2 Randomizing probabilities priors

Poulin and Chung [48] first suggested that BP failures due to degeneracy can be mitigated by randomly perturbing the prior error probabilities associated with qubits involved in frustrated checks.

Building on this idea, the enhanced feedback (EFB) decoder introduces a more systematic strategy [94]. It leverages aggregated information from stabilizer measurements, syndromes, the locations of frustrated checks, and the underlying channel model to compute updated prior probabilities. This feedback-based refinement leads to improved decoding accuracy over depolarizing quantum channels.

A related approach is proposed in [95], which introduces decoder diversity by iteratively reattempting BP decoding with different randomized priors. Each decoder instance in the ensemble exhibits distinct convergence behavior, allowing the collective system to correct a wider range of error patterns than any single decoder alone.

In more recent work, Pradhan et al. [96] employ recurrent neural networks (RNNs) to learn prior biases and message-passing weights within min-sum BP decoders, specifically

for lifted-product codes. These trained decoders are applied sequentially, and under certain assumptions, are guaranteed to successfully correct all errors below a specified weight threshold.

In our study, we evaluate a simplified version of this general idea. When BP decoding fails, we randomly redistribute the prior error probabilities across all qubits using a uniform distribution and re-run BP. This process is repeated multiple times to increase the chance of successful decoding.

4.3 Freezing methods

The earliest freezing strategy was proposed by Poulin and Chung [48]. The method selects an unsatisfied check, and then randomly chooses one of its neighboring qubits. The prior error probability of this qubit is then forced to zero, and BP is rerun for a fixed number of iterations. If the same check remains unsatisfied, the previously frozen qubit is restored, and another neighboring qubit is frozen instead. If the check is no longer unsatisfied but the overall decoding halting condition is not satisfied, a different unsatisfied check is selected and the process continues.

Yao et al. [50] introduced a BP-guided decimation method that freezes qubits based on their LLR values. After running BP for a few iterations, the qubit with the largest absolute LLR is identified and frozen according to the sign of its LLR: if $\text{LLR} > 0$, its prior error probability is fixed to 0; if $\text{LLR} < 0$, it is fixed to 1 (In practice, this corresponds to setting the LLR to a predefined maximum or minimum value, respectively, to ensure

numerical stability.). BP is then continued for a few more iterations, and the process is repeated—freezing one qubit at a time—until all qubits are fixed or decoding succeeds.

Gong et al. [51] observed that the method in [50] tends to select qubits with positive LLRs. To address this, they proposed freezing qubits with the smallest LLR, targeting the qubits most likely to be erroneous instead. Their approach is further enhanced with a guided guessing mechanism, which follows a more sophisticated freezing rule.

A related method, called stabilizer inactivation, was introduced by Du et al. [97]. It works by selecting a set of qubits that support a check in the dual code and “inactivating” them—that is, excluding them from BP. BP is then rerun on the reduced graph. Once the remaining qubits are decoded, the values of the inactivated qubits are inferred by solving a small linear system.

4.4 BP with additional memory effects (MBP)

Kuo et al. [52] introduced a variant of BP with memory effects (MBP), which was originally developed for decoding quaternary quantum codes. When applying this method to binary BP, the update rule for $L_{v_a \rightarrow c_i}$ is modified as follows:

$$L_{v_a \rightarrow c_i} = \log\left(\frac{1 - p_a}{p_a}\right) + \frac{1}{\alpha} \sum_{c_j \in \mathbf{c}_{\partial v_a}} L_{c_j \rightarrow v_a} - L_{c_i \rightarrow v_a}. \quad (4.1)$$

The parameter α is a tunable step-size parameter. When $\alpha = 1$, the equation reduces to the standard BP update rule.

The accuracy of MBP is sensitive to the choice of α , and the optimal value can vary across different codes and noise models. Adopting the technique of diversity decoding,

using a list of scaling factors $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ can further improve decoding accuracy at the cost of increased decoding time. When BP fails under one setting α_i , the decoder switches to the next candidate α_{i+1} and restarts the decoding process. This strategy can also be parallelized to reduce latency.

4.5 Pre-decoder

While the previously discussed methods can improve the decoding success rate of BP-based decoders, many of them incur a significant increase in computational complexity or runtime. To mitigate this, we propose a lightweight **pre-decoder** designed to accelerate decoding by quickly handling low-weight error configurations. This pre-decoder has been implemented as part of the open-source software package **vecdec**, available on GitHub [98] and has already been used in [99, 100].

The rationale behind the pre-decoder is that, for topological LDPC codes, physical error rates below a percolation threshold probability p_c typically give rise to errors that form only small, disconnected clusters on the underlying lattice. In this regime, most error patterns can be decomposed into local clusters whose syndromes can be identified and corrected independently [34]. Exploiting this structural property enables the pre-decoder to resolve a large fraction of error events with negligible computational cost, while passing only the more challenging residual syndromes to BP+OSD. This significantly reduces the number of expensive OSD invocations, leading to substantial runtime improvements.

The pre-decoder attempts to match the observed syndrome vector to a union of small, isolated error clusters. This is achieved by decomposing the syndrome into clusters

using a connectivity graph derived from the parity-check matrix, and then matching each cluster’s syndrome against a precomputed hash table of error-syndrome pairs.

The parameter uW sets the maximum weight of error vectors stored in the hash table (with $uW = 0$ corresponding to disabling the pre-decoder). Another parameter uR sets the maximum variable-variable (v-v) graph distance between non-zero bits in stored error vectors (with $uR = 0$ corresponding to no limit). This helps restrict the hash table to spatially localized errors, improving memory efficiency and cluster matching relevance.

The decoding process proceeds as follows:

1. Prior to decoding, compute the syndrome vectors for all error vectors of weight less than or equal to uW , and store them in a hash table.
2. At runtime, identify all non-zero syndrome bits (unsatisfied check nodes), and form initial clusters by including each such check and all neighboring variable nodes.
3. Merge overlapping clusters. Here, the labeling scheme originally designed for simulating percolation is used to ensure fast updates of the merged clusters
4. For each merged cluster, search the hash table for a matching error vector that reproduces the cluster’s syndrome.
5. If the combined error reproduces the full syndrome, output the result and terminate.

Otherwise, invoke the standard decoder.

In addition, the uX tunable parameter is introduced to control the scope and behavior of the pre-decoder, as described below.

- **uX**: a bitmap parameter that controls additional decoding options. The default value is $\mathbf{uX} = 0$, which disables all optional features. Non-zero values enable the following behaviors:
 - **Bit 0 set** (e.g., $\mathbf{uX} = 1$): attempt to match the full syndrome as a single low-weight error before splitting to clusters. This may improve decoding success for certain codes and can occasionally offer a small speed-up.
 - **Bit 1 set** (e.g., $\mathbf{uX} = 2$): if some clusters fail to match, discard the matched clusters and pass only the unmatched portions of the syndrome to the standard decoder. This strategy can improve decoding success in codes where degenerate errors frequently confuse BP decoders. However, it is not recommended for surface codes, where syndrome clusters are highly interdependent.
 - **Bit 2 set** (e.g., $\mathbf{uX} = 4$): when $\mathbf{uW} > 1$, expand each cluster one additional step by including all neighboring check nodes of the variable nodes already in the cluster, followed by another merging step. This option can improve success rates for certain codes such as surface codes for $\mathbf{uW} > 1$.

Chapter 5

Numerical results

5.1 Comparison of modified BP methods

We evaluate the accuracy of various BP-based decoding strategies under three noise models: the code capacity model, the phenomenological noise model, and the circuit-level noise model. These simulations are performed using rotated surface codes, toric codes, and constructed GB codes [99] shown in the table 5.1.

For the code capacity noise model, we assume that each qubit independently undergoes a Pauli Z error with probability p . Pauli X errors are not considered in this setting.

For the phenomenological noise model, we set the error probabilities for both data qubits and syndrome measurements to be equal to p . We set the number of repeated syndrome measurements to $m = d$, where d is the code distance.

For the circuit-level noise model, all gate depolarizing rates and single-qubit X error rates are set to p . To remain consistent with `Stim`'s treatment of circuit-level noise—where identical columns in the parity-check matrix are automatically merged—we apply a similar

column-merging transformation to the rotated surface code, as described in Section 2.10. The resulting version is denoted as “rotated surface code (simplified)” in all figures.

We first analyze decoding performance for rotated surface codes and toric codes. Under circuit-level noise models, the resulting Tanner graphs are significantly larger, making simulations computationally expensive. We therefore begin with smaller codes to enable systematic comparison across decoding methods.

In addition, rotated surface and toric codes possess structural features that partially resemble those of Tanner graphs corresponding to circuit error models. In particular, rotated surface codes include parity-check rows of weight 2 and 4, while toric codes consist of uniform weight-4 checks. Similarly, Tanner graphs under circuit-level noise typically contain many weight-3 checks. These similarities make topological codes useful proxies for predicting decoder performance under circuit-level noise.

We also include results for GB codes to provide a contrasting benchmark. Unlike topological codes, GB codes are constructed using random circulant matrices and do not exhibit local structure. As a result, their stabilizer generators typically do not produce low-weight degenerate errors, which are prevalent in topological codes and known to challenge BP decoders. Due to this structural difference, GB codes exhibit decoding behavior that is markedly different under BP. To highlight this contrast, we present simulation results on GB codes alongside those for surface and toric codes.

The GB codes used in our simulations are constructed from two sparse polynomials, each with 3 or 4 non-zero terms. We refer to these as “3+3 GB codes” and “4+4 GB codes”, respectively. The table below summarizes the code parameters used in our experiments.

Table 5.1: List of GB codes

$[[n,k,d]]$	Wt	$a(x)$	$b(x)$
$[[30,4,6]]$	6	$1 + x^{10} + x^{11}$	$1 + x + x^8$
$[[56,6,8]]$	6	$1 + x^3 + x^8$	$1 + x^{15} + x^{17}$
$[[84,6,10]]$	6	$1 + x^{10} + x^{36}$	$1 + x^{15} + x^{17}$
$[[24,4,5]]$	8	$1 + x^3 + x^4 + x^9$	$1 + x^3 + x^5 + x^{10}$
$[[42,6,7]]$	8	$1 + x^7 + x^{15} + x^{16}$	$1 + x + x^7 + x^{18}$
$[[70,8,10]]$	8	$1 + x^{15} + x^{16} + x^{18}$	$1 + x + x^{24} + x^{27}$

We evaluate the performance of various BP-based decoding strategies on surface codes and toric codes first for several reasons.

First, under realistic circuit-level noise models, the corresponding Tanner graphs are significantly larger, making simulations computationally expensive. To facilitate comparison across decoding strategies, we focus on smaller codes first.

Second, these codes exhibit features that are present in circuit-level noise models. Specifically, the rotated surface code includes parity-check rows of weight 2 and 4, while the toric code consists of weight-4 checks. Similarly, circuit-level noise models generate parity-check matrices with a large number of weight-3 checks. As a result, these codes serve as useful proxies for analyzing and predicting the performance of decoding strategies under realistic circuit-level noise.

We compare the following decoding strategies:

BP + OSD. As a reference for near-optimal decoding performance, we evaluate the combination of BP followed by OSD. The search depth parameter λ is set to 100 in all simulations. The general time complexity of the OSD step is $\mathcal{O}(n^3)$, but for quantum LDPC codes with fixed row weight, this can be reduced to $\mathcal{O}(n^2)$ in our implementation.

BP with different message update schedules. We implement four schedules:

- **Parallel (Flooding):** All messages are updated simultaneously based on the previous iteration.
- **Serial-C:** Check nodes are updated sequentially. For each check node, all incoming messages are updated first, followed by all outgoing messages.
- **Serial-V:** Variable nodes are updated sequentially. For each variable node, all incoming messages are updated first, followed by all outgoing messages.
- **Random Serial-V:** At the beginning of each iteration, a random permutation of variable nodes is selected. The serial-V schedule is then applied according to this randomized order.

Regardless of the chosen schedule, the BP decoding procedure has a time complexity of $\mathcal{O}(n)$.

Random-BP. Upon a BP decoding failure, the prior error probabilities are randomly perturbed, and BP is rerun. In our simulations, we allow up to $m = 10$ decoding attempts. The overall time complexity is $\mathcal{O}(nm)$ if the error rate is large.

Freezing methods. We evaluate two freezing methods, in which a qubit is frozen after a fixed number of BP iterations based on specific selection rules. BP then continues until all qubits are frozen. The two selection strategies are:

- **Freeze max-abs-LLR (BPGD):** At each step, the qubit with the largest absolute LLR is frozen according to its sign.
- **Freeze min-LLR:** At each step, the qubit with the smallest LLR is frozen, targeting the qubits most likely to be erroneous.

For both methods, the overall time complexity is $\mathcal{O}(n^2)$.

Memory BP (MBP). We run BP with additional memory effects, as described in Section 4.4. We select a list of parameters $\alpha \in \{1.00, 0.99, 0.98, \dots, 0.50\}$. If decoding fails for one α , the decoder retries using the next value in the list. When executed in parallel across multiple CPU cores, the overall time complexity remains $\mathcal{O}(n)$.

To better assess the structural effectiveness of each decoding method, we plot the logical error rate versus the physical error rate on a log-log scale and include reference lines with fixed integer slopes (e.g., 2, 3, 4, etc.) in each plot. The slope of the curve at low error rates reflects the decoder’s ability to handle fixed-weight errors. For example, a slope of 2 implies that some weight-2 errors remain uncorrected. This allows us to distinguish between methods that merely *shift* the error rate curve downward—without improving the slope, and those that truly increase the decoder’s effective distance.

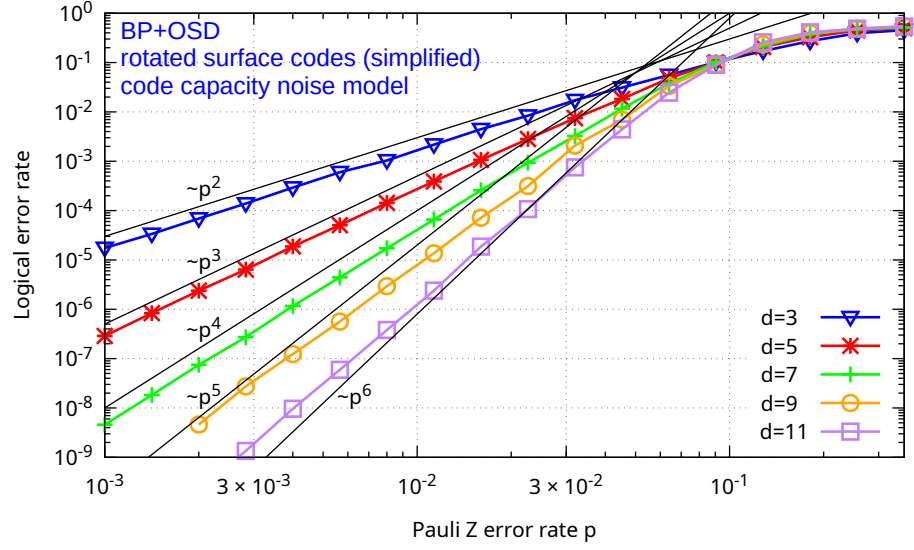


Figure 5.1: Logical error rate versus Pauli Z error rate for rotated surface codes (simplified) with distances as indicated. BP+OSD achieves the optimal logarithmic slope of $\lfloor \frac{d+1}{2} \rfloor$

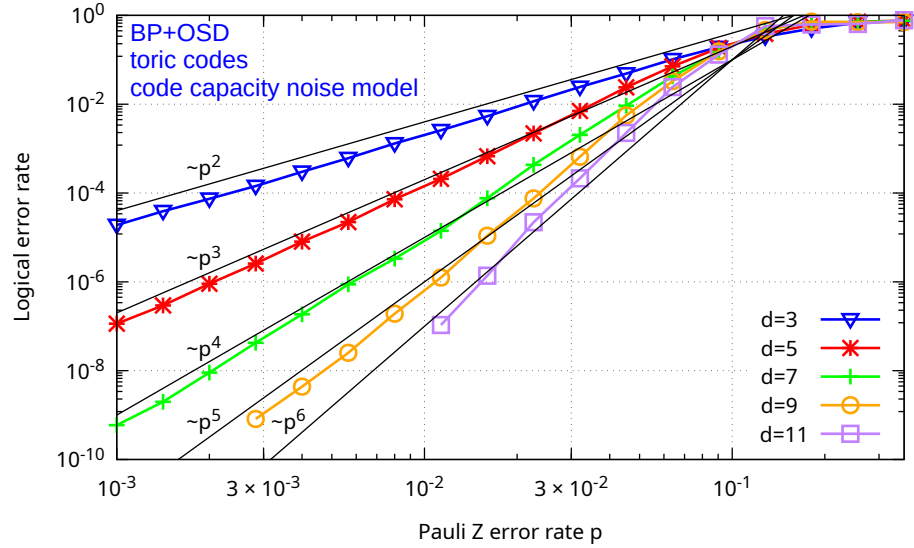


Figure 5.2: Logical error rate versus Pauli Z error rate for toric codes with distances as indicated. BP+OSD achieves the optimal logarithmic slope of $\lfloor \frac{d+1}{2} \rfloor$

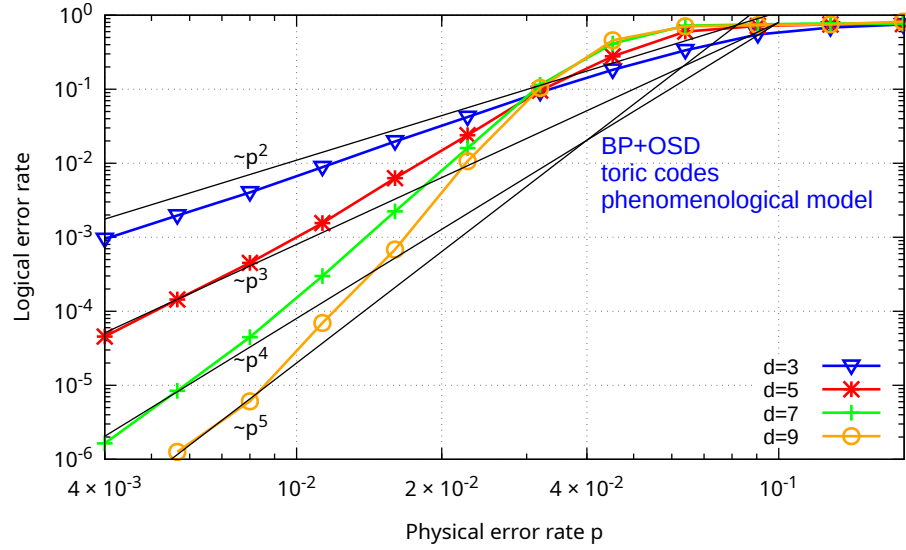


Figure 5.3: Logical error rate versus physical error rate for toric codes under the phenomenological noise model with distances as indicated. BP+OSD achieves the optimal logarithmic slope of $\lfloor \frac{d+1}{2} \rfloor$

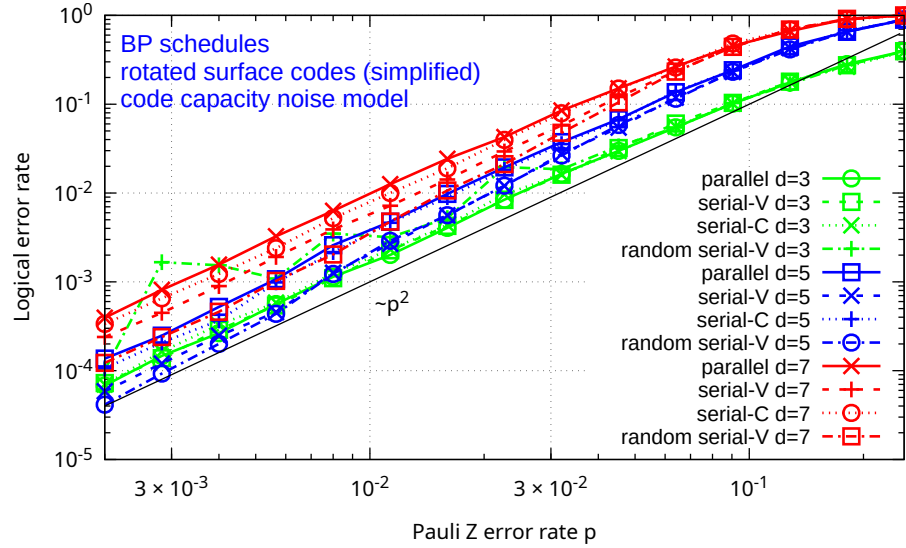


Figure 5.4: Comparison of different BP decoding schedules for rotated surface codes (simplified). Different schedules do not alter the slope for rotated surface codes (simplified).

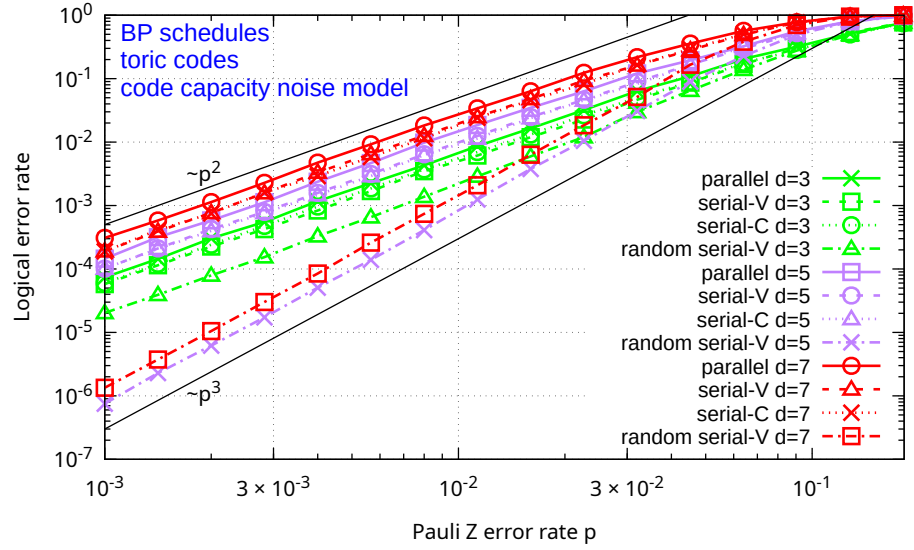


Figure 5.5: Comparison of different BP decoding schedules for toric codes. The random serial-V schedule improves the decoding slope from 2 to 3.

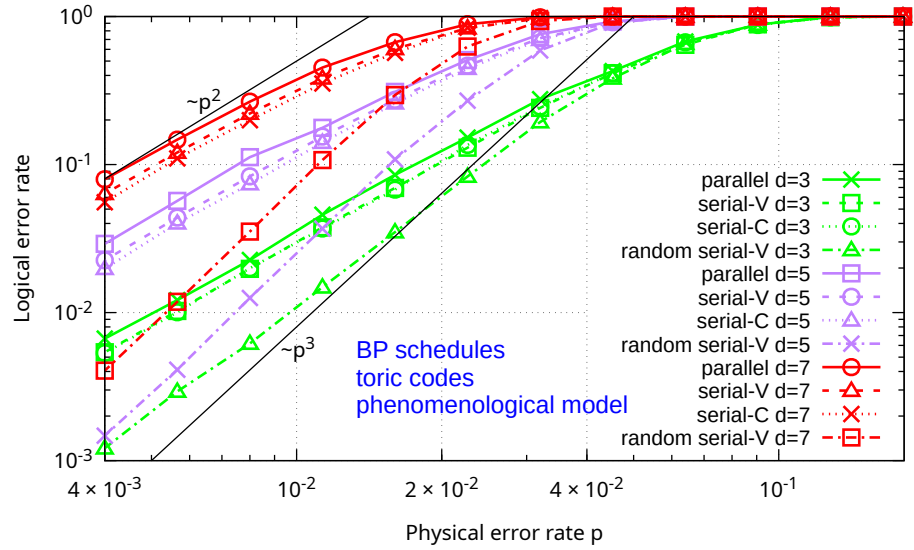


Figure 5.6: Comparison of different BP decoding schedules for toric codes under the phenomenological noise model. The random serial-V schedule improves the decoding slope from 2 to 3.

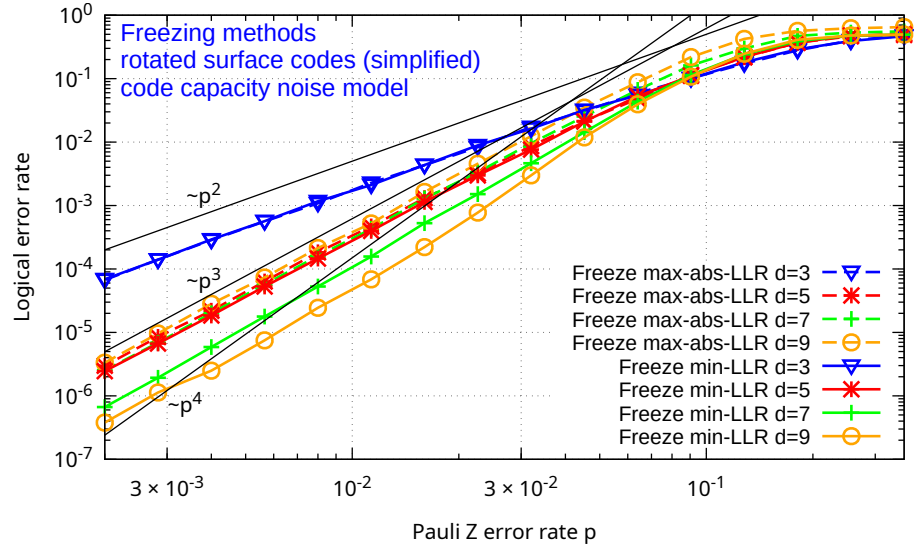


Figure 5.7: Comparison of different freezing methods for rotated surface codes (simplified). Both methods saturate at a maximum slope of 3.

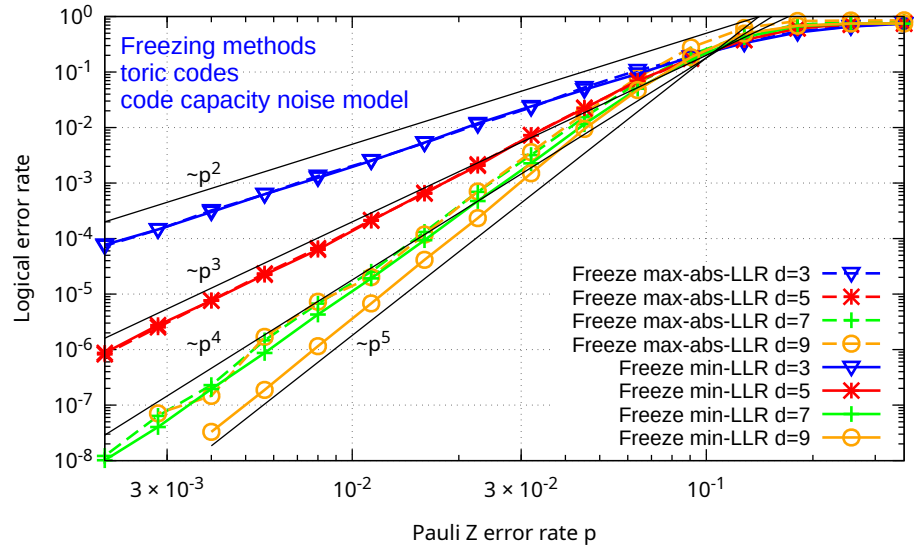


Figure 5.8: Comparison of different freezing methods for toric codes.

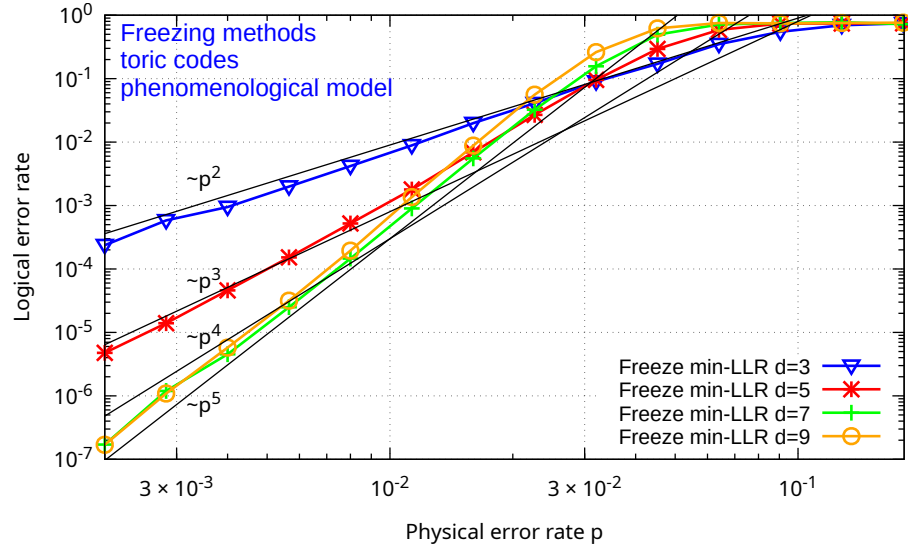


Figure 5.9: Comparison of different freezing methods for toric codes under the phenomenological noise model.

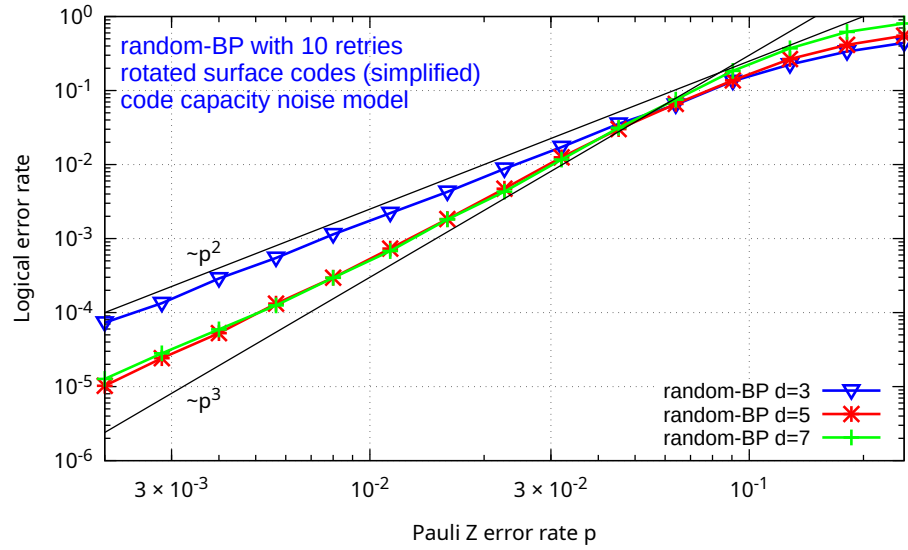


Figure 5.10: Random-BP for rotated surface codes (simplified). This method fails on some weight-2 errors, as indicated by the logarithmic slope of the plots.

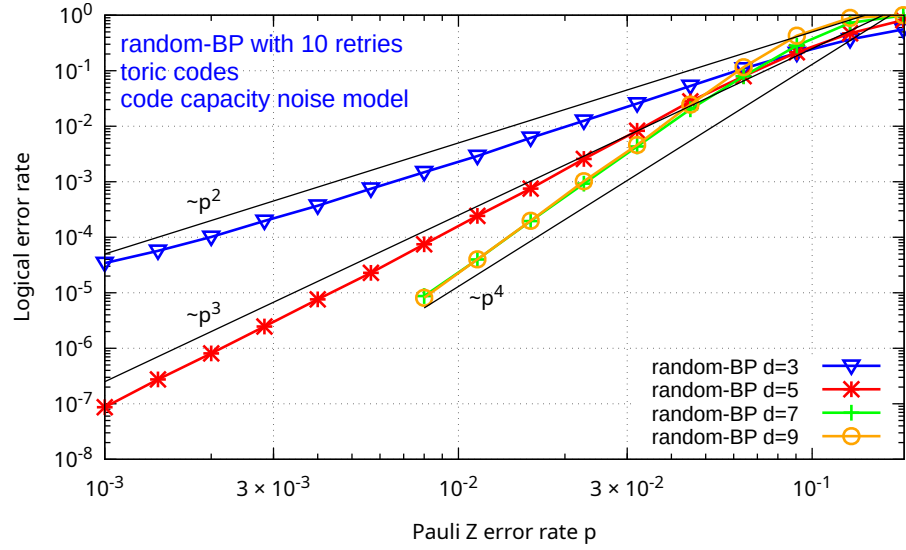


Figure 5.11: Random-BP for toric codes. This method saturates at a maximum slope of 4.

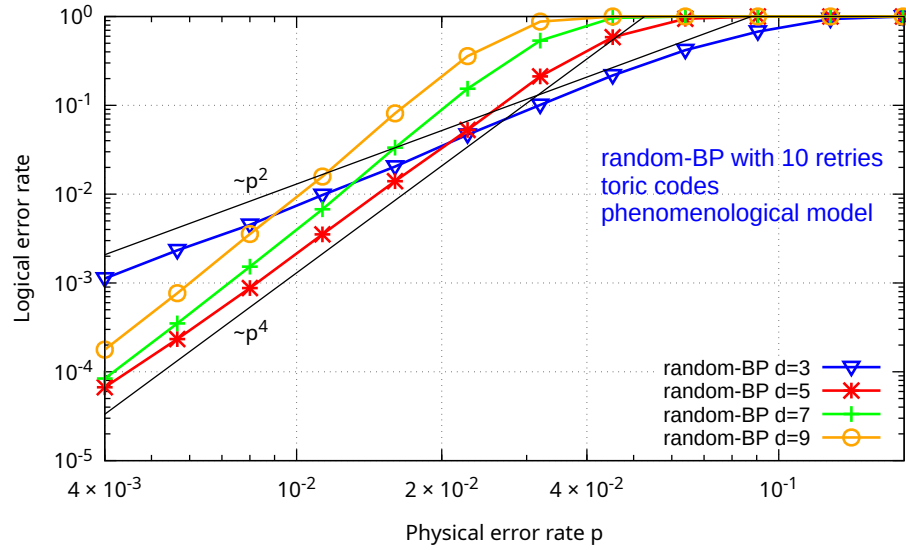


Figure 5.12: Random-BP for toric codes under the phenomenological noise model. This method saturates at a maximum slope of 4.

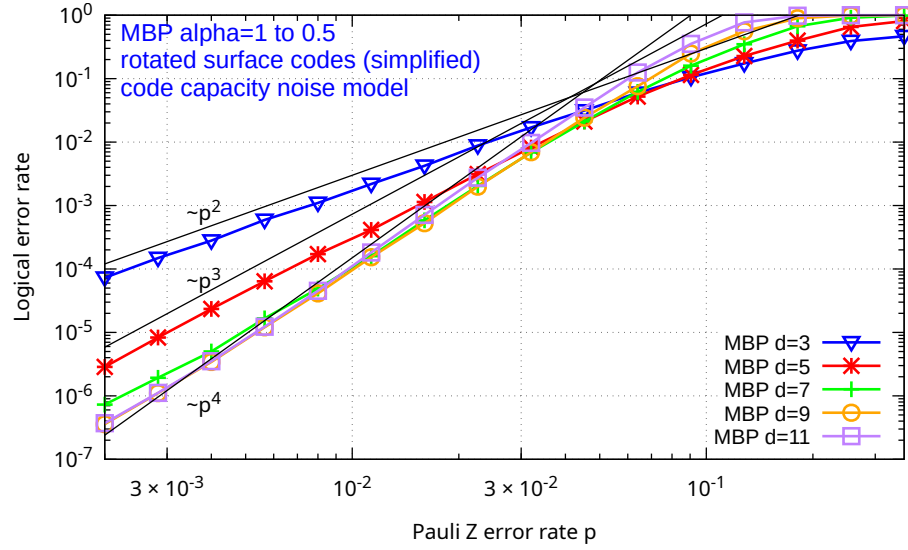


Figure 5.13: Binary MBP for rotated surface codes (simplified). This method saturates at a maximum slope of 4.

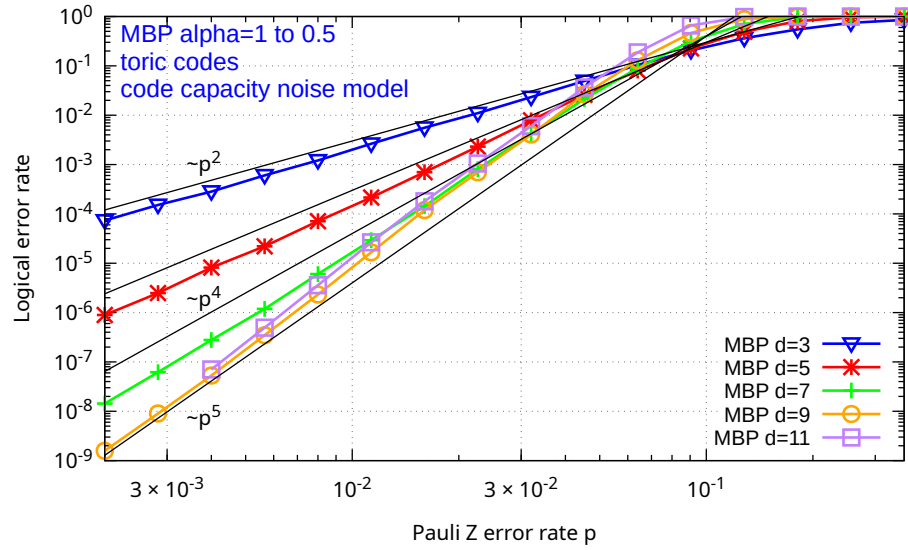


Figure 5.14: Binary MBP for toric codes. This method saturates at a maximum slope of 5.

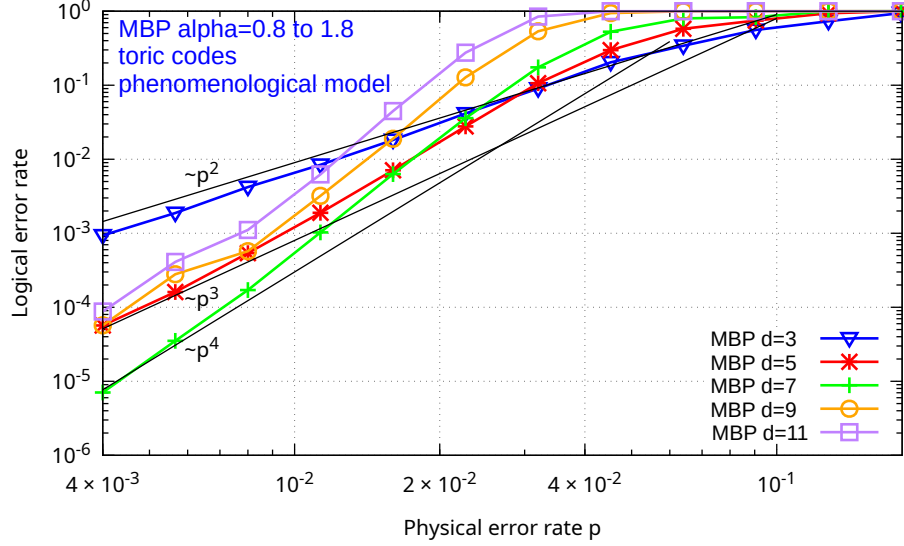


Figure 5.15: Binary MBP for toric codes under the phenomenological noise model. This method saturates at a maximum slope of 5.

Across all simulations on surface codes and toric codes, we observe that those BP-based decoding methods improve decoding accuracy over standard BP. However, these improvements generally come at the cost of increased computational complexity and longer decoding time. In practice, we find a trade-off: methods that yield greater performance gains tend to be slower.

Moreover, none of the tested methods besides BP+OSD are able to achieve the optimal decoding slope of $\lfloor \frac{d+1}{2} \rfloor$ at higher code distances. As the distance d increases, all methods exhibit earlier error floors, indicating persistent decoding failures for certain low-weight error patterns.

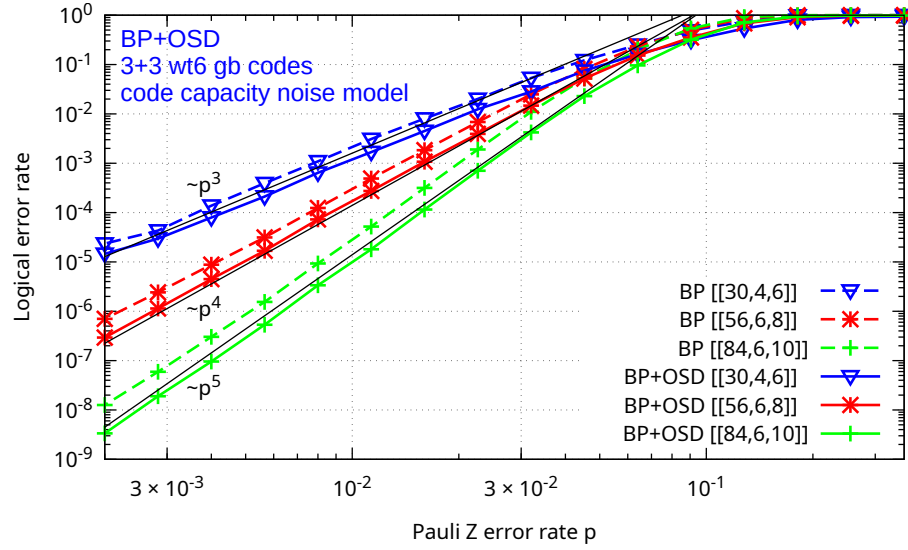


Figure 5.16: Comparison of BP and BP+OSD for 3+3 GB codes. For these codes, BP alone achieves a slope greater than $\lfloor \frac{d+1}{2} \rfloor - 1$, approaching the accuracy by BP+OSD.

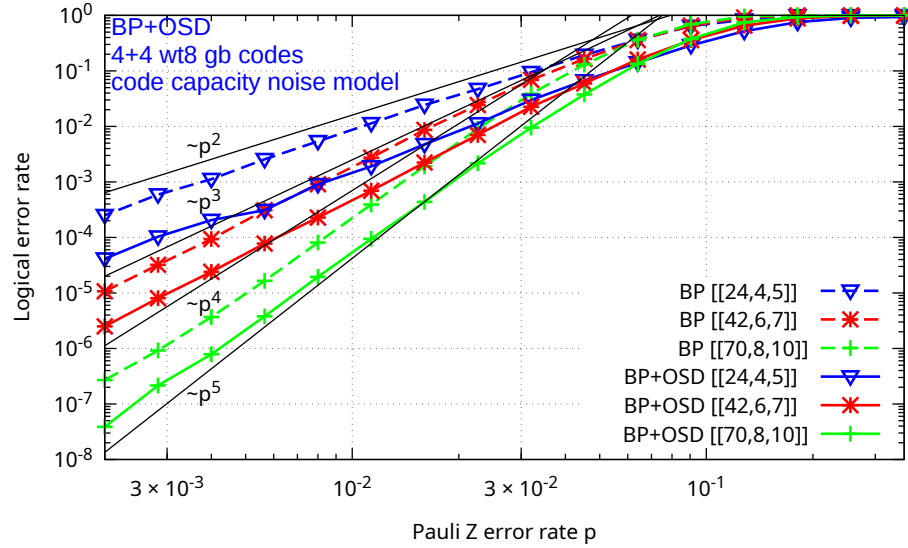


Figure 5.17: Comparison of BP and BP+OSD for 4+4 GB codes. For these codes, BP alone achieves a slope greater than $\lfloor \frac{d+1}{2} \rfloor - 1$, approaching the accuracy by BP+OSD.

As shown in Fig. 5.17, for GB codes without low-weight degeneracy, the BP decoder already achieves high accuracy, with decoding slopes exceeding $\lfloor \frac{d+1}{2} \rfloor - 1$. In such cases, further improvements using techniques such as OSD or other enhancements offer only marginal gains, with performance nearly matching that of BP+OSD. This suggests that for non-degenerate codes, standard BP can be highly effective on its own.

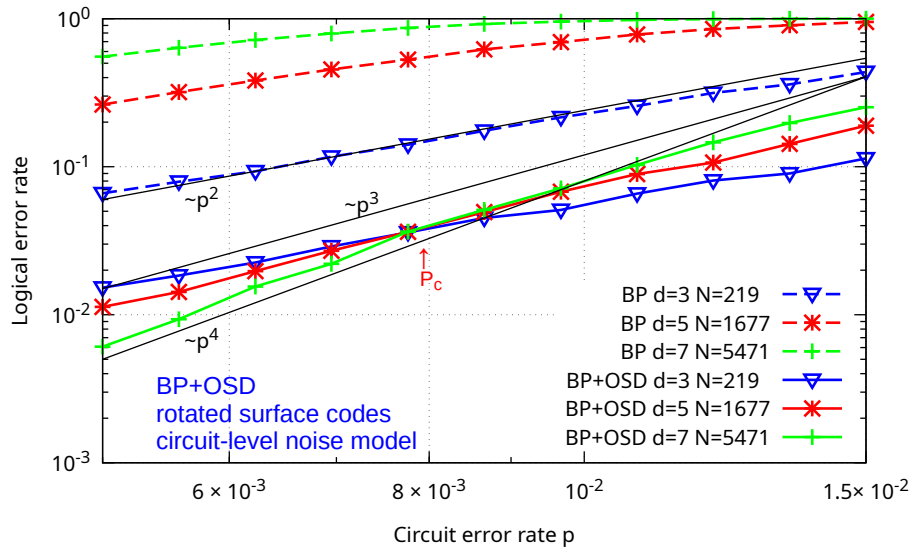


Figure 5.18: BP+OSD for rotated surface codes under circuit-level noise model, which achieves an optimal slope of $\lfloor \frac{d+1}{2} \rfloor$.

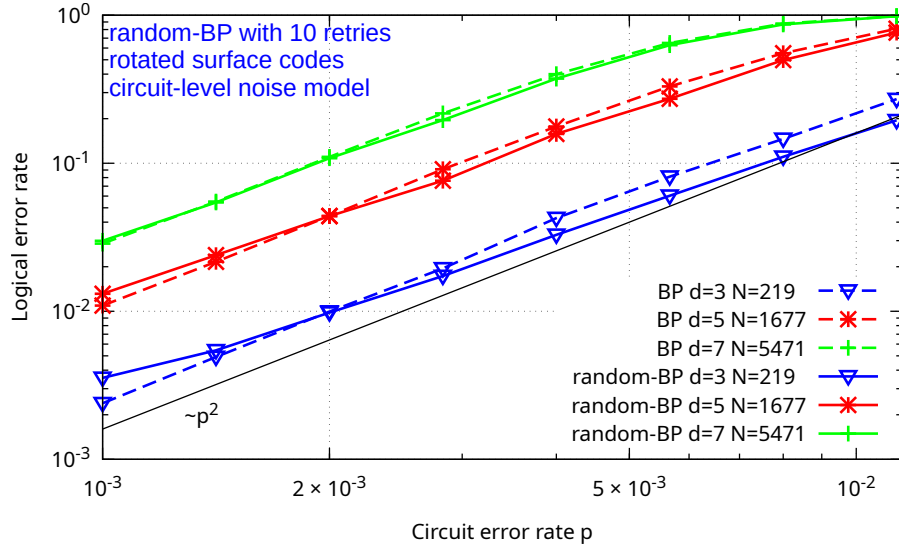


Figure 5.19: Random-BP for rotated surface codes under the circuit-level noise model. This method does not show improvement compared to BP.

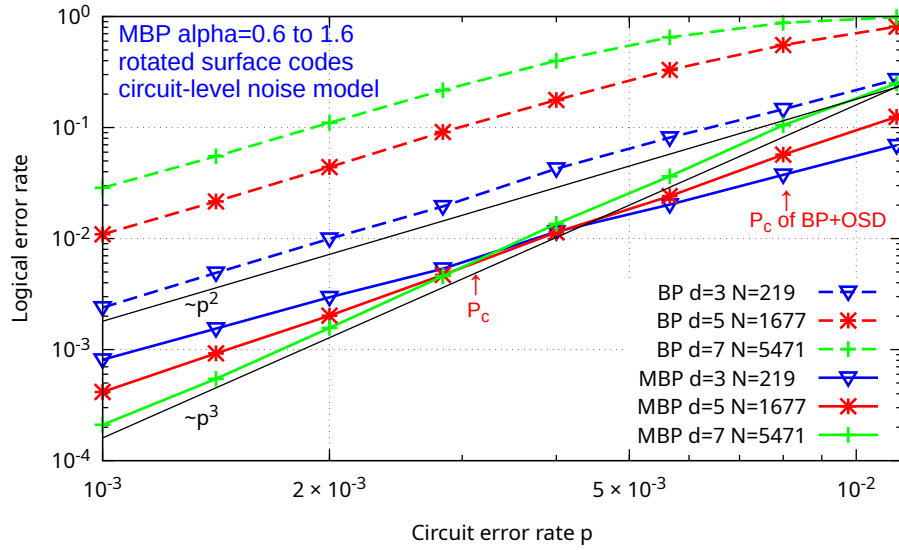


Figure 5.20: Binary MBP for rotated surface codes under circuit-level noise model. This method saturates at a slope of 3, indicating some weight-3 errors cannot be decoded.

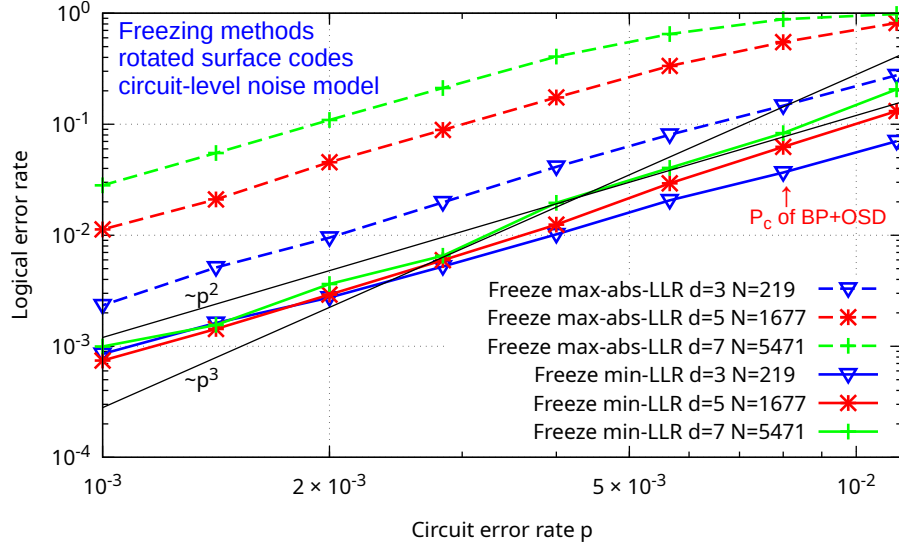


Figure 5.21: Freezing methods for rotated surface codes under circuit-level noise model. This method saturates at a slope of 2, indicating some weight-2 errors cannot be decoded.

5.2 Pre-decoder

As discussed earlier, circuit-level decoding results in significantly larger Tanner graphs, making BP decoding computationally expensive. Although the various improvement techniques can substantially improve decoding accuracy, they tend to further increase the decoding time due to repeated decoding attempts.

To address this efficiency bottleneck, we evaluate the performance of the pre-decoder described in Section 4.5. The pre-decoder attempts to match the syndrome vector with precomputed low-weight error patterns via cluster decomposition and hash lookup, thereby bypassing full-scale decoding in many cases. When successful, it dramatically reduces decoding time without sacrificing accuracy.

We observe that setting $uW = 1$ can dramatically improve decoding speed without sacrificing accuracy. Furthermore, the pre-decoder is particularly effective at low physical error rates, in such cases, a substantial fraction of decoding attempts can be resolved entirely within the pre-decoder, bypassing more expensive decoding steps. The uX parameter is chosen to have the best accuracy and runtime.

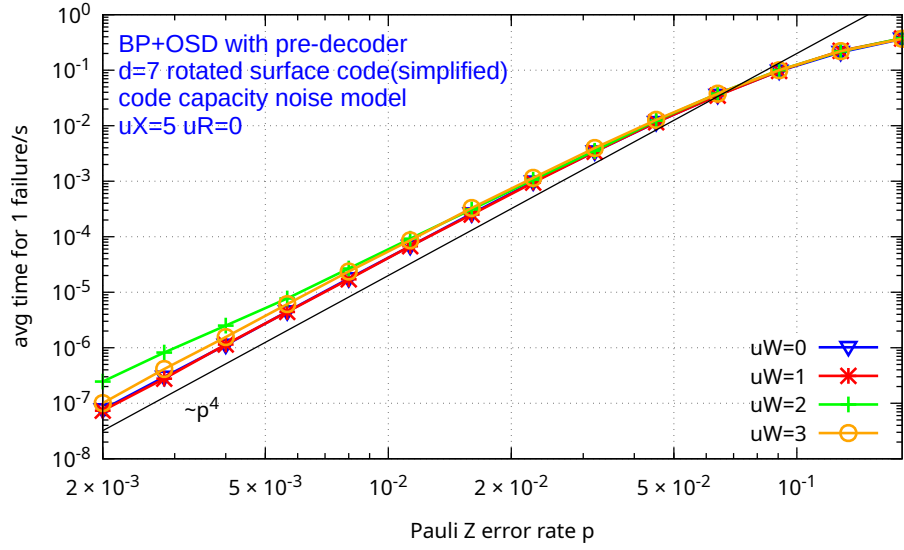


Figure 5.22: Logical error rate versus Pauli Z error rate for the $d = 7$ rotated surface code(simplified), decoded by BP+OSD with pre-decoder. With $uW = 1$, decoding accuracy is preserved, while larger uW slightly degrade accuracy.

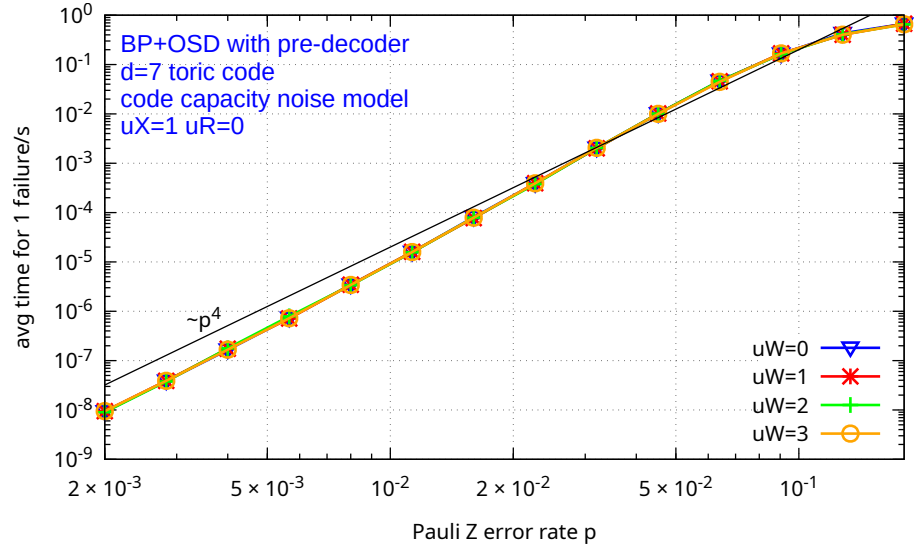


Figure 5.23: Logical error rate versus Pauli Z error rate for the $d = 7$ toric code, decoded by BP+OSD with pre-decoder. Different uW values give the same decoding accuracy.

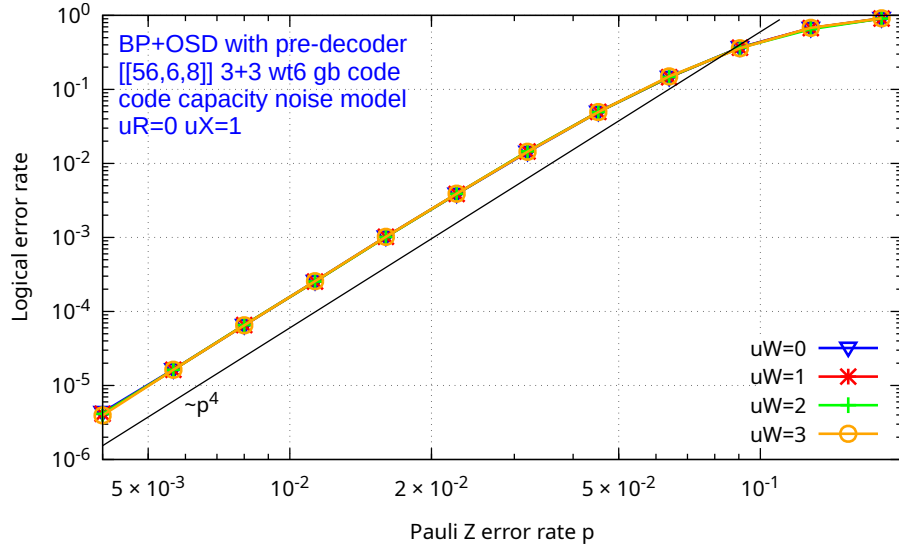


Figure 5.24: Logical error rate versus Pauli Z error rate for the $[[56, 6, 8]]$ 3+3 GB code, decoded by BP+OSD with pre-decoder. Different uW values give the same decoding accuracy.

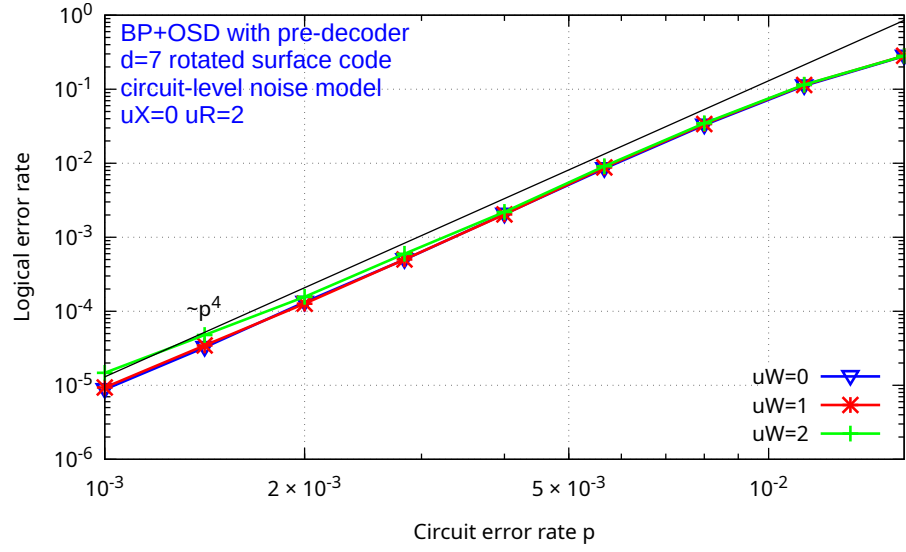


Figure 5.25: Decoding accuracy for $d = 7$ rotated surface code circuit, decoded by BP+OSD with pre-decoder. With $uW = 1$, decoding accuracy is preserved, while $uW = 2$ slightly degrades accuracy.

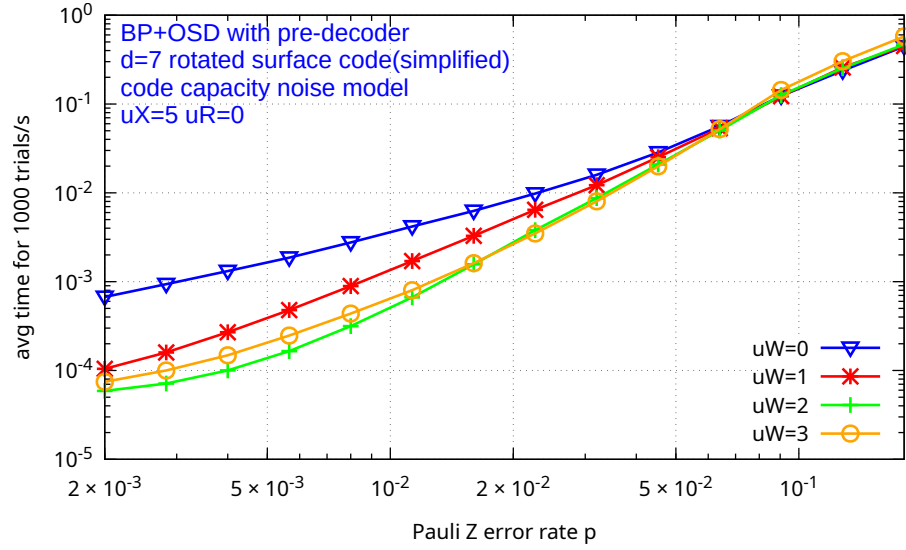


Figure 5.26: Average execution time on 1000 trials for the $d = 7$ rotated surface code (simplified), decoded by BP+OSD with pre-decoder. Larger uW values result in faster decoding for a range of p values; at smaller values of p , high-weight clusters are not encountered often enough to make a difference.

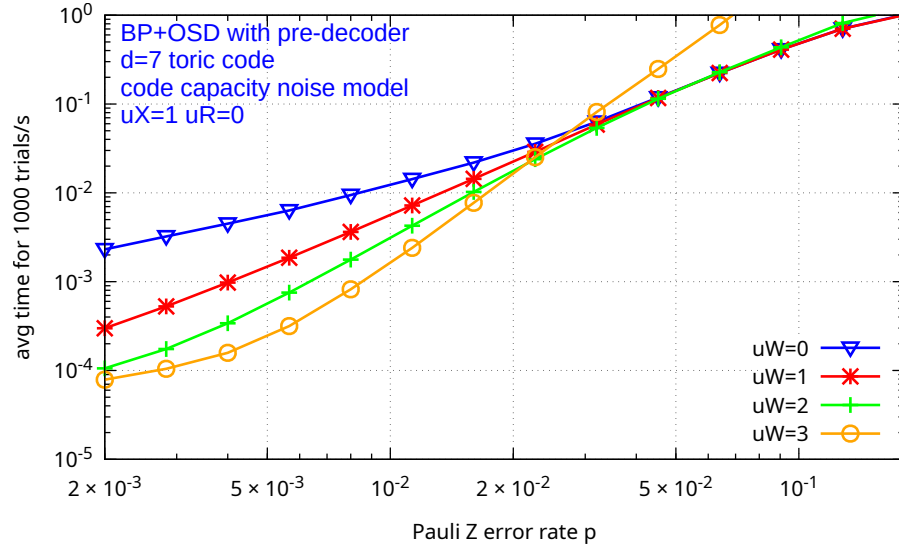


Figure 5.27: Average execution time on 1000 trials for the $d = 7$ toric code, decoded by BP+OSD with pre-decoder.

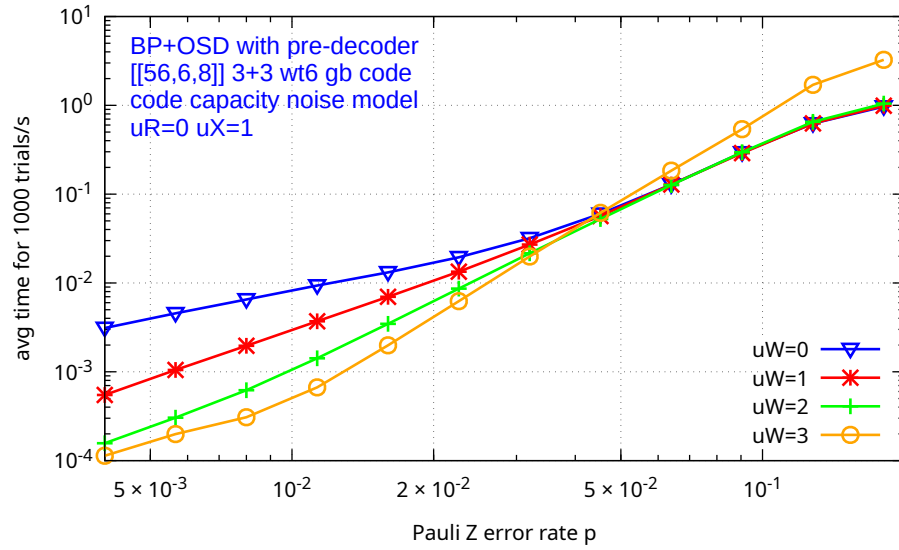


Figure 5.28: Average execution time on 1000 trials for $[[56, 6, 8]]$ 3+3 wt6 GB code, decoded by BP+OSD with pre-decoder.

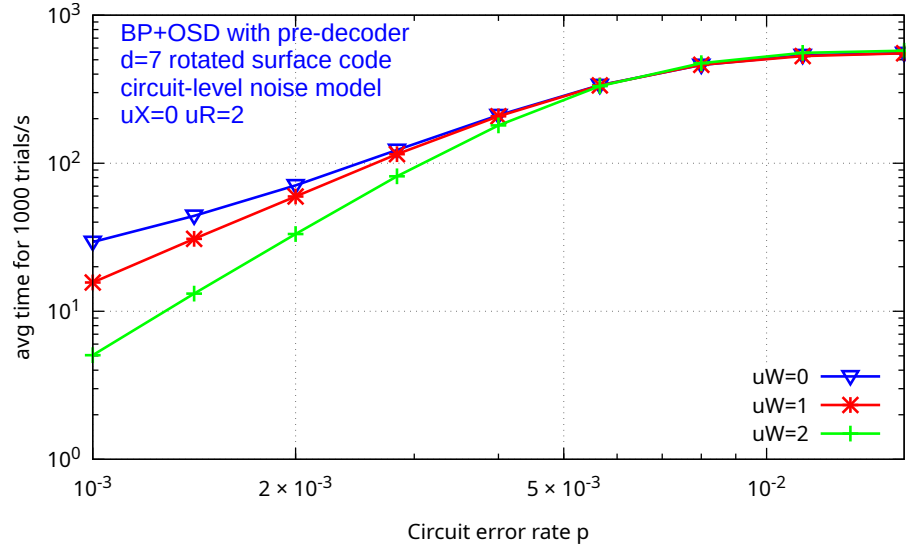


Figure 5.29: Average execution time on 1000 trials for $d = 7$ rotated surface code under circuit-level noise model, decoded by BP+OSD with pre-decoder. Larger uW values result in faster decoding.

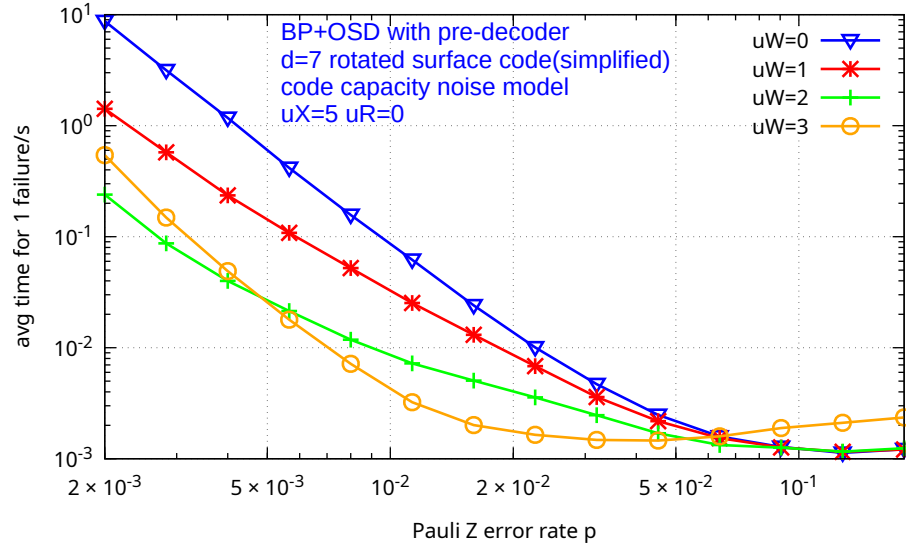


Figure 5.30: Average execution time on 1 failure for $d = 7$ rotated surface code (simplified) decoded by BP+OSD with pre-decoder.

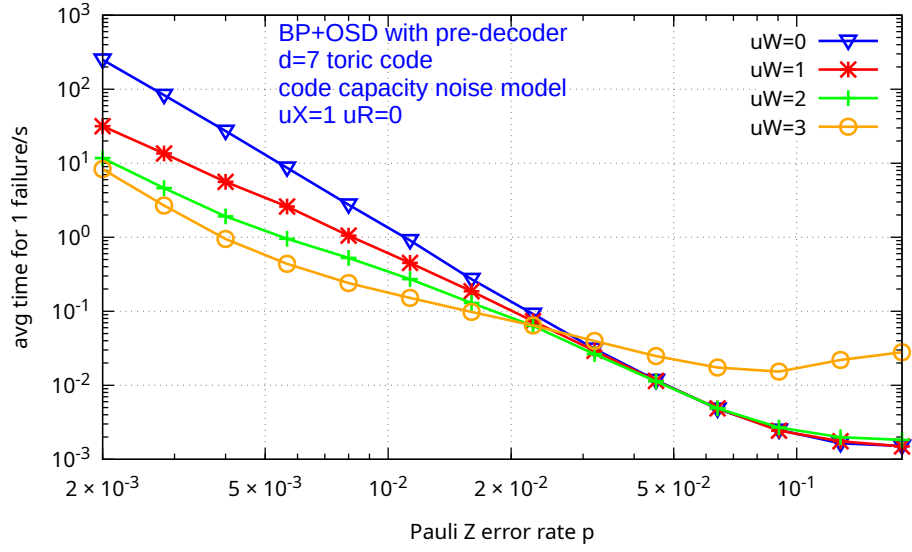


Figure 5.31: Average execution time on 1 failure for $d = 7$ toric code decoded by BP+OSD with pre-decoder.

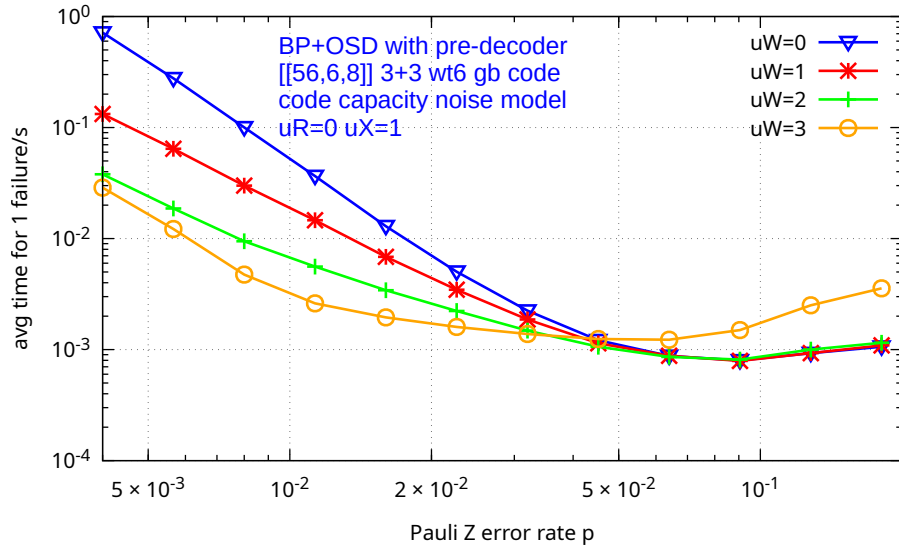


Figure 5.32: Average execution time on 1 failure for $[[56, 6, 8]]$ 3+3 wt6 GB code decoded by BP+OSD with pre-decoder.

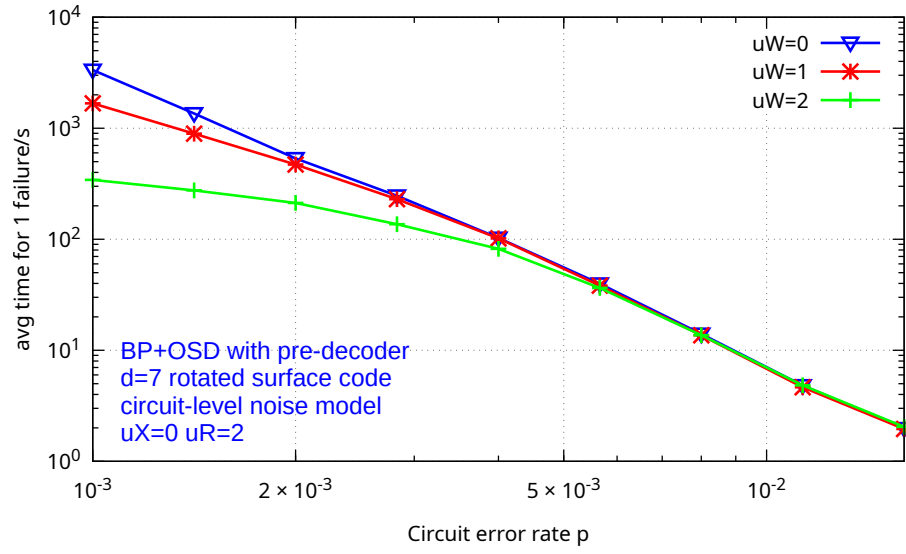


Figure 5.33: Average execution time on 1 failure for $d = 7$ rotated surface codes under circuit-level noise model, decoded by BP+OSD with pre-decoder. Larger uW values result in faster decoding.

Chapter 6

Conclusions

This dissertation investigated how to improve belief-propagation (BP) decoding for quantum LDPC codes across representative code families (rotated surface, toric, and generalized bicycle) and noise models (code-capacity, phenomenological, and circuit-level). Through a unified evaluation, we clarified when established BP refinements help, how their gains scale with code distance, and what computational costs they incur. On low-degeneracy generalized bicycle codes, plain BP already approaches the performance of BP+OSD, leaving limited headroom for further accuracy gains. In contrast, on rotated-surface and toric codes, none of the tested BP variants attained the optimal log-log slope of the logical error rate versus the physical error rate as distance increases.

To address latency without sacrificing much accuracy, we introduced a simple cluster-based pre-decoder that decomposes each measured syndrome into connected components, solves each component via hash-based lookups, and forwards only the residual

to BP (optionally followed by OSD). This front-end reduces both the frequency of OSD invocations and the dimensionality of the residual problems, yielding orders-of-magnitude speedups in the practically relevant low-error-rate regime with minimal accuracy loss. Because the residual after pre-decoding is typically much smaller, the amortized cost of the linear-algebra core in OSD (e.g., Gaussian elimination) is also reduced.

Our benchmarks further show a clear accuracy–runtime trade-off governed by tunable parameters. In particular, increasing the component-width/exploration parameter u_W consistently accelerates decoding; moderate settings usually deliver the best balance, while overly aggressive choices can induce small accuracy degradations on rotated-surface codes under circuit-level noise. Across other BP refinements (e.g., randomized scheduling/priors and light decimation), we observed benefits that are code- and noise-model dependent rather than universal. We also examined degeneracy-reducing matrix transformations as a preprocessing step for highly degenerate instances; these transformations can improve BP’s ability to correct low-weight errors without changing maximum-likelihood performance.

In practice, the most immediate implication is straightforward: inserting the proposed pre-decoder as a lightweight first pass in BP+OSD pipelines lowers latency substantially with negligible implementation overhead. Selecting conservative default parameters (with moderate u_W) preserves accuracy across the evaluated codes and noise models, while still capturing a large fraction of the attainable speedup. Taken together, these results sharpen the understanding of when BP refinements are effective, when OSD remains indispensable, and how locality-aware preprocessing can act as an efficient catalyst for low-latency decoding under realistic noise.

Bibliography

- [1] Michael E Peskin. *An Introduction to quantum field theory*. CRC press, 2018.
- [2] Michael B Green, John H Schwarz, Edward Witten, and Paul H Frampton. Superstring theory volume 1: Introduction volume 2: Loop amplitudes, anomalies and phenomenology, 1987.
- [3] Stephen Wolfram. New kind of science, 1997.
- [4] Seth Lloyd. The universe as quantum computer, 2013.
- [5] Yuri Manin. Computable and uncomputable. *Sovetskoye Radio, Moscow*, 128:15, 1980.
- [6] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.
- [7] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.
- [8] Dilshod Nematov and Mirabbos Hojamberdiev. Machine learning - driven materials discovery: Unlocking next-generation functional materials – a minireview, 2025.
- [9] Zhixun Li, Bin Cao, Rui Jiao, Liang Wang, Ding Wang, Yang Liu, Dingshuo Chen, Jia Li, Qiang Liu, Yu Rong, Liang Wang, Tong yi Zhang, and Jeffrey Xu Yu. Materials generation in the era of artificial intelligence: A comprehensive survey, 2025.
- [10] Xiao-Qi Han , Xin-De Wang , Meng-Yuan Xu , Zhen Feng , Bo-Wen Yao , Peng-Jie Guo , Ze-Feng Gao , and Zhong-Yi Lu . Ai-driven inverse design of materials: Past, present, and future. *Chinese Physics Letters*, 42(2):027403, March 2025.

- [11] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, February 2017.
- [12] Dian Wu, Riccardo Rossi, Filippo Vicentini, Nikita Astrakhantsev, Federico Becca, Xiaodong Cao, Juan Carrasquilla, Francesco Ferrari, Antoine Georges, Mohamed Hibat-Allah, et al. Variational benchmarks for quantum many-body problems. *Science*, 386(6719):296–301, 2024.
- [13] Google AI Quantum et al. Hartree-fock on a superconducting quantum computer. *Science*, 2020.
- [14] Abhinav Kandala et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 2017.
- [15] Immanuel Bloch, Jean Dalibard, and Sylvain Nascimbene. Quantum simulations with ultracold quantum gases. *Nature Physics*, 2012.
- [16] Christopher Monroe et al. Programmable quantum simulations of spin systems with trapped ions. *Reviews of Modern Physics*, 2021.
- [17] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
- [18] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994.
- [19] Muhammad AbuGhanem. Ibm quantum computers: evolution, performance, and future directions. *The Journal of Supercomputing*, 81(5), April 2025.
- [20] Jwo-Sy Chen, Erik Nielsen, Matthew Ebert, Volkan Inlek, Kenneth Wright, Vandiver Chaplin, Andrii Maksymov, Eduardo Pérez, Amrit Poudel, Peter Maunz, and John Gamble. Benchmarking a trapped-ion quantum computer with 30 qubits. *Quantum*, 8:1516, November 2024.
- [21] Lars Pause, Lukas Sturm, Marcel Mittenbühler, Stephan Amann, Tilman Preuschoff, Dominik Schäffner, Malte Schlosser, and Gerhard Birkel. Supercharged two-dimensional tweezer array with more than 1000 atomic qubits. *Optica*, 11(2):222, February 2024.
- [22] Karen Wintersperger, Florian Dommert, Thomas Ehmer, Andrey Hoursanov, Johannes Klepsch, Wolfgang Maurer, Georg Reuber, Thomas Strohm, Ming Yin, and

- Sebastian Luber. Neutral atom quantum computing hardware: performance and end-user perspective. *EPJ Quantum Technology*, 10(1), August 2023.
- [23] H Aghaee Rad, T Ainsworth, RN Alexander, B Altieri, MF Askarani, R Baby, L Banchi, BQ Baragiola, JE Bourassa, RS Chadwick, et al. Scaling and networking a modular photonic quantum computer. *Nature*, pages 1–8, 2025.
 - [24] A Yu Kitaev. Quantum error correction with imperfect gates. In *Quantum communication, computing, and measurement*, pages 181–188. Springer, 1997.
 - [25] Emanuel Knill, Raymond Laflamme, and Wojciech Zurek. Threshold accuracy for quantum computation. *arXiv preprint quant-ph/9610011*, 1996.
 - [26] D. Aharonov and M. Ben-Or. Fault-tolerant quantum computation with constant error. In *STOC '97 Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, El Paso, Texas, USA – May 04-06, 1997*, pages 176–188. ACM New York, NY USA, 1997.
 - [27] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill. Topological quantum memory. *J. Math. Phys.*, 43:4452, 2002.
 - [28] S. B. Bravyi and A. Yu. Kitaev. Quantum codes on a lattice with boundary. unpublished, 1998.
 - [29] H. Bombin and M. A. Martin-Delgado. Topological quantum distillation. *Phys. Rev. Lett.*, 97:180501, Oct 2006.
 - [30] MACKAY David. Sparse graph codes for quantum error-correction. *IEEE Transactions on Information Theory*.
 - [31] E. Knill and R. Laflamme. Concatenated quantum codes. Unpublished, arXiv:quant-ph/9608012, 1996.
 - [32] Dave Bacon. Operator quantum error-correcting subsystems for self-correcting quantum memories. *Phys. Rev. A*, 73:012340, Jan 2006.
 - [33] D. Gottesman. Fault-tolerant quantum computation with constant overhead. *Quant. Information and Computation*, 14:1338–1371, 2014.
 - [34] Alexey A Kovalev and Leonid P Pryadko. Fault tolerance of quantum low-density parity check codes with sublinear distance scaling. *Physical Review A*, 87(2):020304, 2013.

- [35] Qian Xu, J. Pablo Bonilla Ataides, Christopher A. Pattison, Nithin Raveendran, Dolev Bluvstein, Jonathan Wurtz, Bane Vasic, Mikhail D. Lukin, Liang Jiang, and Hengyun Zhou. Constant-overhead fault-tolerant quantum computation with reconfigurable atom arrays, 2023.
- [36] Sergey Bravyi, Andrew W. Cross, Jay M. Gambetta, Dmitri Maslov, Patrick Rall, and Theodore J. Yoder. High-threshold and low-overhead fault-tolerant quantum memory. *Nature*, 627(8005):778–782, March 2024.
- [37] Ke Wang, Zhide Lu, Chuanyu Zhang, Gongyu Liu, Jiachen Chen, Yanzhe Wang, Yaozu Wu, Shibo Xu, Xuhao Zhu, Feitong Jin, Yu Gao, Ziqi Tan, Zhengyi Cui, Ning Wang, Yiren Zou, Aosai Zhang, Tingting Li, Fanhao Shen, Jiarun Zhong, Zehang Bao, Zitian Zhu, Yihang Han, Yiyang He, Jiayuan Shen, Han Wang, Jia-Nan Yang, Zixuan Song, Jinfeng Deng, Hang Dong, Zheng-Zhi Sun, Weikang Li, Qi Ye, Si Jiang, Yixuan Ma, Pei-Xin Shen, Pengfei Zhang, Hekang Li, Qiujiang Guo, Zhen Wang, Chao Song, H. Wang, and Dong-Ling Deng. Demonstration of low-overhead quantum error correction codes, 2025.
- [38] Oscar Higgott and Craig Gidney. Sparse blossom: correcting a million errors per core second with minimum-weight matching. *Quantum*, 9:1600, 2025.
- [39] Nicolas Delfosse and Naomi H. Nickerson. Almost-linear time decoding algorithm for topological codes. *Quantum*, 5:595, 2021.
- [40] Pavel Panteleev and Gleb Kalachev. Degenerate quantum LDPC codes with good finite length performance. *Quantum*, 5:585, 2021.
- [41] Laleh Aghababaie Beni, Oscar Higgott, and Noah Shutty. Tesseract: A search-based decoder for quantum error correction. *arXiv preprint arXiv:2503.10988*, 2025.
- [42] Kai R. Ott, Bence Hetényi, and Michael E. Beverland. Decision-tree decoders for general quantum ldpc codes, 2025.
- [43] Joschka Roffe, David R White, Simon Burton, and Earl Campbell. Decoding across the quantum low-density parity-check code landscape. *Physical Review Research*, 2(4):043423, 2020.
- [44] Stasiu Wolanski and Ben Barber. Ambiguity clustering: an accurate and efficient decoder for qldpc codes, 2025.
- [45] Timo Hillmann, Lucas Berent, Armanda O. Quintavalle, Jens Eisert, Robert Wille, and Joschka Roffe. Localized statistics decoding: A parallel decoding algorithm for quantum low-density parity-check codes, 2024.

- [46] Antonio deMarti iOlius, Imanol Etxezarreta Martinez, Joschka Roffe, and Josu Etxezarreta Martinez. An almost-linear time decoding algorithm for quantum ldpc codes under circuit-level noise, 2024.
- [47] Antonio deMarti iOlius and Josu Etxezarreta Martinez. The closed-branch decoder for quantum ldpc codes, 2024.
- [48] D. Poulin and Y. Chung. On the iterative decoding of sparse quantum codes. *Quant. Info. and Comp.*, 8:987, 2008.
- [49] Julien Du Crest, Francisco Garcia-Herrero, Mehdi Mhalla, Valentin Savin, and Javier Valls. Layered decoding of quantum ldpc codes. *arXiv e-prints*, pages arXiv–2308, 2023.
- [50] Hanwen Yao, Waleed Abu Laban, Christian Häger, Alexandre Graell i Amat, and Henry D. Pfister. Belief propagation decoding of quantum ldpc codes with guided decimation, 2024.
- [51] Anqi Gong, Sebastian Cammerer, and Joseph M Renes. Toward low-latency iterative decoding of qldpc codes under circuit-level noise. *arXiv preprint arXiv:2403.18901*, 2024.
- [52] Kao-Yueh Kuo and Ching-Yi Lai. Exploiting degeneracy in belief propagation decoding of quantum codes. *npj Quantum Information*, 8(1):111, 2022.
- [53] Kao-Yueh Kuo and Ching-Yi Lai. Fault-tolerant belief propagation for practical quantum memory, 2024.
- [54] Tristan Müller, Thomas Alexander, Michael E. Beverland, Markus Bühler, Blake R. Johnson, Thilo Maurer, and Drew Vandeth. Improved belief propagation is sufficient for real-time decoding of quantum memory, 2025.
- [55] Keyi Yin, Xiang Fang, Jixuan Ruan, Hezi Zhang, Dean Tullsen, Andrew Sornborger, Chenxu Liu, Ang Li, Travis Humble, and Yufei Ding. Symbreak: Mitigating quantum degeneracy issues in qldpc code decoders by breaking symmetry, 2024.
- [56] Leonid P Pryadko. On maximum-likelihood decoding with circuit-level errors. *Quantum*, 4:304, 2020.
- [57] David J. C. MacKay and Radford M. Neal. Good codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45(2):399–431, 1999.
- [58] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

- [59] Erdal Arıkan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Transactions on Information Theory*, 55(7):3051–3073, 2009.
- [60] Emanuel Knill and Raymond Laflamme. Theory of quantum error-correcting codes. *Phys. Rev. A*, 55:900–911, 1997.
- [61] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, Caltech, 1997.
- [62] A Robert Calderbank and Peter W Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996.
- [63] A. M. Steane. Simple quantum error-correcting codes. *Phys. Rev. A*, 54:4741–4751, 1996.
- [64] Bin Dai, Shilin Ding, and Grace Wahba. Multivariate bernoulli distribution. 2013.
- [65] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962.
- [66] S. B. Bravyi and A. Yu. Kitaev. Quantum codes on a lattice with boundary, 1998.
- [67] A. Yu. Kitaev. Fault-tolerant quantum computation by anyons. *Ann. Phys.*, 303:2, 2003.
- [68] S. Bravyi and B. Terhal. A no-go theorem for a two-dimensional self-correcting quantum memory based on stabilizer codes. *New Journal of Physics*, 11(4):043029, 2009.
- [69] J-P Tillich and G Zemor. Quantum ldpc codes with positive rate and minimum distance proportional to $n^{1/2}$. In *2009 IEEE International Symposium on Information Theory*.
- [70] Alexey A Kovalev and Leonid P Pryadko. Quantum kronecker sum-product low-density parity-check codes with finite rate. *Physical Review A*, 88(1):012311, 2013.
- [71] Renyu Wang and Leonid P Pryadko. Distance bounds for generalized bicycle codes. *Symmetry*, 14(7):1348, 2022.
- [72] Pavithran Iyer and David Poulin. Hardness of decoding quantum stabilizer codes, 2013.
- [73] Oscar Higgott and Craig Gidney. Sparse blossom: correcting a million errors per core second with minimum-weight matching. *arXiv preprint arXiv:2303.15933*, 2023.

- [74] Oscar Higgott. Pymatching: A python package for decoding quantum codes with minimum-weight perfect matching, 2021.
- [75] Yue Wu and Lin Zhong. Fusion blossom: Fast mwpm decoders for qec, 2023.
- [76] Yue Wu, Namitha Liyanage, and Lin Zhong. Micro blossom: Accelerated minimum-weight perfect matching decoding for quantum error correction. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '25, page 639–654. ACM, March 2025.
- [77] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), September 2012.
- [78] Nicolas Delfosse and Gilles Zémor. Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel. *Physical Review Research*, 2(3), July 2020.
- [79] Yue Wu, Namitha Liyanage, and Lin Zhong. An interpretation of union-find decoder on weighted graphs. *arXiv preprint arXiv:2211.03288*, 2022.
- [80] Ashley M Stephens. Fault-tolerant thresholds for quantum error correction with the surface code. *Physical Review A*, 89(2):022321, 2014.
- [81] Craig Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021.
- [82] Tom Richardson and Ruediger Urbanke. *Modern coding theory*. Cambridge university press, 2008.
- [83] Kao-Yueh Kuo and Ching-Yi Lai. Refined belief propagation decoding of sparse-graph quantum codes. *IEEE Journal on Selected Areas in Information Theory*, 1(2):487–498, August 2020.
- [84] Jacob Goldberger and Haggai Kfir. Serial schedules for belief-propagation: analysis of convergence time. *IEEE Transactions on Information Theory*, 54(3):1316–1319, 2008.
- [85] Juntan Zhang and Marc PC Fossorier. Shuffled iterative decoding. *IEEE Transactions on Communications*, 53(2):209–213, 2005.
- [86] Eran Sharon, Simon Litsyn, and Jacob Goldberger. An efficient message-passing schedule for ldpc decoding. In *2004 23rd IEEE Convention of Electrical and Electronics Engineers in Israel*, pages 223–226. IEEE, 2004.

- [87] Tom Richardson. Error floors of ldpc codes. In *Proceedings of the annual Allerton conference on communication control and computing*, volume 41, pages 1426–1435. The University; 1998, 2003.
- [88] Changyan Di, David Proietti, I Emre Telatar, Thomas J Richardson, and Rüdiger L Urbanke. Finite-length analysis of low-density parity-check codes on the binary erasure channel. *IEEE Transactions on Information theory*, 48(6):1570–1579, 2002.
- [89] Lara Dolecek, Zhengya Zhang, Venkat Anantharam, Martin J Wainwright, and Borivoje Nikolic. Analysis of absorbing sets and fully absorbing sets of array-based ldpc codes. *IEEE Transactions on Information Theory*, 56(1):181–201, 2009.
- [90] David JC MacKay and Michael S Postol. Weaknesses of margulis and ramanujan-margulis low-density parity-check codes. *Electronic Notes in Theoretical Computer Science*, 74:97–104, 2003.
- [91] Pascal O Vontobel and Ralf Koetter. Graph-cover decoding and finite-length analysis of message-passing iterative decoding of ldpc codes. *arXiv preprint cs/0512078*, 2005.
- [92] Nithin Raveendran and Bane Vasić. Trapping sets of quantum ldpc codes. *Quantum*, 5:562, 2021.
- [93] Marc PC Fossorier and Shu Lin. Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on information Theory*, 41(5):1379–1396, 2002.
- [94] Yun-Jiang Wang, Barry C. Sanders, Bao-Ming Bai, and Xin-Mei Wang. Enhanced feedback iterative decoding of sparse quantum codes. *IEEE Transactions on Information Theory*, 58(2):1231–1241, 2012.
- [95] David Declercq, Bane Vasic, Shiva Kumar Planjery, and Erbao Li. Finite alphabet iterative decoders—part ii: Towards guaranteed error correction of ldpc codes via iterative decoder diversity. *IEEE transactions on communications*, 61(10):4046–4057, 2013.
- [96] Asit Kumar Pradhan, Nithin Raveendran, Narayanan Rengaswamy, Xin Xiao, and Bane Vasić. Learning to decode trapping sets in qldpc codes. In *2023 12th International Symposium on Topics in Coding (ISTC)*, pages 1–5. IEEE, 2023.
- [97] Julien Du Crest, Mehdi Mhalla, and Valentin Savin. Stabilizer inactivation for message-passing decoding of quantum ldpc codes. In *2022 IEEE Information Theory Workshop (ITW)*, pages 488–493. IEEE, 2022.

- [98] L. P. Pryadko. vecdec - vectorized decoder and ler estimator. <https://github.com/QEC-pages/vecdec>, 2025.
- [99] Hsiang-Ku Lin, Xingrui Liu, Pak Kau Lim, and Leonid P. Pryadko. Single-shot and two-shot decoding with generalized bicycle codes, 2025.
- [100] Hsiang-Ku Lin, Pak Kau Lim, Alexey A. Kovalev, and Leonid P. Pryadko. Abelian multi-cycle codes for single-shot error correction, 2025.