Towards reliable organisms: fault-tolerance in unconventional models of computation

by

Andrew K. Tan

BASc Engineering Science, University of Toronto, 2019

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN PHYSICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2024

Authored by:	Andrew K. Tan Department of Physics January 12, 2024
Certified by:	Isaac L. Chuang Professor of Physics, Thesis Supervisor
Accepted by:	Lindley Winslow Professor of Physics Associate Department Head of Physics

Towards reliable organisms: fault-tolerance in unconventional models of computation

by

Andrew K. Tan

Submitted to the Department of Physics
on January 12, 2024 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN PHYSICS

ABSTRACT

Computation is often described in abstract and idealized terms. In this setting, the details of the computer can be neglected: different models of computation are interchangeable for a cost which is at most polynomial in the size of the task at hand. The situation is more complicated for computers composed of imperfect components which are liable to introduce errors into the computation. Early work by John von Neumann showed that computers composed of imperfect components could be used to simulate perfect computation with high probability through the introduction of redundancy at the hardware level, i.e. they could be made *fault-tolerant* if certain conditions were met. These conditions, however, depend crucially on the details of noisy computational primitives; while the principles of fault-tolerance remain—store and manipulate data redundantly and perform error correction throughout—the constructions must be designed bespoke to the primitives at hand. In this thesis, we present examples of fault-tolerance in less conventional models of computation and investigate cases in which hardware-level fault-tolerant design may be more efficient in spite of the requisite redundancy.

First, we develop new fault-tolerance results in unconventional models of classical and quantum computation. In particular, we study a model of formula-based computation over larger alphabets subject to symmetric noise; and show that performing computation with large alphabet majority gates results in strictly larger nominal thresholds than achievable using Boolean majority gates. We then move away from a formula-based architecture to study large-alphabet computation based on feed-forward artificial neural networks subject to analog noise. Using the biologically-inspired grid code, we show that fault-tolerant neural network-based computation can be achieved. Then, we envision quantum computation composed of subroutines—rather than gates—developing a method for the error correction of noisy quantum subroutines based on the quantum singular value transform.

Second, we seek a precise understanding when fault-tolerance is not only possible but preferable. In certain cases, the choice between non-fault-tolerant and fault-tolerant designs is clear: for classical computation, hardware-based fault-tolerance is rarely used due to existence of cheap and reliable transistors; in quantum computation, fault-tolerance appears to be the most economical route towards large-scale computation owing to the difficulty of reliably manipulating quantum information. The trade-off may not be as clear in less conventional models of computation. We make a detailed accounting of the resources required to achieve fault-tolerance in a simple setting which allows us to precisely delineate a regime in which hardware-level fault-tolerance is preferred despite the requisite redundancy.

Lastly, we discuss some connections between the study of fault-tolerance and information theory. In particular, we argue that an algorithm we developed for noisy compression can be understood to be finding optimal encodings over noisy channel and is therefore helpful for designing good encodings of data for fault-tolerant computation. This is followed by a review of upper bounds on the fault-tolerant threshold based on information theoretic arguments. We suggest that new measures of information are needed if existing upper bounds are to be improved for circuits, and propose a high-level template for this work.

We end with a reflection on the term "organism" and a discussion of the implications of this collection of results.

Thesis supervisor: Isaac L. Chuang
Title: Professor of Physics

Acknowledgments

It has been my goal to complete this dissertation before a large language model is capable of performing an all-around better job. If I have succeeded at this goal, it is perhaps only just so. That said, it seems to me that the ultimate end of prompt engineering looks very much like graduate advising and I therefore owe a huge debt of gratitude to my prompt engineer, Isaac Chuang. Ike has a singularly broad expertise that spans the gamut from machine learning to quantum information to atomic physics and beyond. It goes without saying that I've learned a lot from Ike, but it's his contagious enthusiasm and ability to put deeply technical topics into a broader context that have pushed me to discover new connections in a large corpus of scholarly knowledge. In addition to his ability to prompt engineer a graduate dissertation, I am grateful for his generosity with his limited time and his support academically and otherwise. Though we are still grappling with how to interact with increasingly powerful artificial intelligence, it is clear that we can all benefit from being more like Ike.

I would like to thank Max Tegmark who introduced me to the MIT community and showed me that it was the place to be. My first couple years are filled with memories of being in a dorm room spending countless hours talking and writing papers with Max and his ability to communicate complex concepts is something to which I aspire. I also would like to thank the final member of my thesis committee, Daniel Harlow, whose comments have improved this thesis. I would also like to thank Sydney Miller and Cathy Modica for their administrative support throughout the years.

At MIT, I have had the pleasure of working with a number of wonderful people, many of whom have had a hand in the work presented herein: Jiahai Feng, Ila Fiete, Matthew Ho, Yuan Liu, John Martyn, Oris Neto, Zane Rossi, Minh Tran, Silviu Udrescu, Tailin Wu, and Alexander Zlokapa. I would also like to thank the friends and colleagues that have been part of this journey: Peter, Kyle, Arko, Beili, Ouail, Felix, Brendan, Ziming, Bhairav, Adrian, Eric, Gabe, Matt, Nicolas, David, Jasmine, Tamara, and countless friends at home. I have learned immensely from all of you and am excited to see what the future holds in store.

I am also grateful to my teachers and mentors throughout the years who taught me to predict the next token in various domains: Juan Carrasquilla, Kyros Kutulakos, Hoi-Kwong Lo, Michael McMaster, Sandra Rothauser, David Schwab, and Henri van Bemmel.

Finally, I am grateful for the support of my partner Emily, and my family back home. It is their consistent support that has enabled me to pursue this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computation is often described in abstract terms, as a mathematical operation accomplished through a sequence of idealized operations and subroutines. In reality, this abstract description must first be encoded into physical degrees of freedom, made to undergo a dynamical process, and subsequently decoded. Often, these carefully choreographed dynamics occur in systems composed of large numbers of imperfect components. For modern computers, this complex dynamical process occurs in a substrate comprised of tens of billions of transistors. In quantum computers, the subject of much active research, mathematical procedures are mapped onto dynamical systems comprised of tens to thousands of quantum mechanical objects. More abstractly, our bodies, which continually make decisions based upon a stream of external stimuli are reliant on the coordination of tens of trillions of cells. More abstractly still, the value of financial instruments is determined collectively through the emergent computation effected by hundreds of millions of market participants. In these last two examples, while the information processing is not being performed by computers of any conventional description, they are undoubtedly performing computation of a less conventional sort. Three questions motivate the scope of this thesis:

(Q1) When can arbitrarily complicated mathematical procedures be natively and fault-tolerantly encoded in the dynamics of less conventional models of computation?

(Q2) When is fault-tolerance not merely possible but desirable?

(Q3) What connections exist between fault-tolerance and information theory?

While there is a large body of work addressing these questions in more standard circuit-based models of computation, relatively little is known about fault-tolerance in other systems that perform noisy information processing. Often, these less conventional models of information processing—such as those emergent in biological systems—are precisely those that would benefit from fault-tolerant design due to high native error rates; and as models of noisy information processing are rarely perfectly interchangeable, it is necessary to develop an understanding of fault-tolerance tailored to the computational primitives at hand, i.e. at the *hardware-level.* In our quest to explore fault-tolerance as a property of systems that perform information processing more generally, some of which bear little resemblance to what one might conventionally consider to be a computer, we refer to general information processing systems composed of noisy components as *reliable organisms*[1].

1.1 On the synthesis of reliable classical organisms

The first exposition on this topic dates back to a lecture series by John von Neumann titled "Probabalistic logics and the synthesis of reliable organisms from unreliable components" [Neu56], in which von Neumann remarked at the robustness of information processing in biological systems subject to noise. As a proxy for biological systems, von Neumann chose to study noisy automata which functioned in a manner similar to the mathematical model of the neuron developed by McCulloch and Pitts [MP43]. In more modern parlance, the noisy automaton of von Neumann is best described as a *noisy Boolean circuit.* von Neumann

[1]It is from von Neumann that we borrow the term "reliable organism" [Neu56].

considered building reliable computations from different basic gate sets, arriving at a key observation: the probability of error in the circuit can be bounded by a constant in a manner independent of computation depth by encoding data in an error correcting code and interleaving computation with error correction as long as the basic components erred with a probability below some *threshold*. While techniques and thresholds depend on the specifics of the basic components chosen, this observation is the essence of all *positive* fault-tolerance results to date.

Since von Neumann's initial work, positive results have been tightened for the special case of the Boolean *formula*, i.e. circuits with fan-out 1 for different gate sets. The first tight results for 3-input gates are due to Hajek and Weller in 1991 [HW91]. This was followed by tight results for 2-input gates by Evans and Pippenger in 1998 [EP98], which proved more difficult to obtain due to the lack of an error correcting 2-input majority gate. Evans and Schulman extended the positive results of Hajek and Weller to all gates sets of odd fan-in in 2003 [ES03]. Furthermore, these formula-based positive thresholds are conjectured to hold for the more general class of fault-tolerant circuits. While these works paint a nearly definitive picture of fault-tolerance in the Boolean circuit model of computation, fault-tolerance in larger alphabet models of computation and those not based on circuits have received little attention.

In Chapter 2, we expand on the body of positive results through the study fault-tolerance in less conventional models of classical computation (Q1). In an attempt to remain true to von Neumann's initial conception of a reliable organism, we do this with an eye towards the study of reliable biologically-inspired models. Noting that there are few naturally occurring two-state, i.e. Boolean, systems, we focus on fault-tolerant computation over larger alphabets. We study two models of fault-tolerance over larger alphabets: the first is a generalization of past fault-tolerance literature to computation based on larger alphabet circuits, and the second assumes a neural network model of computation. Despite the relatively straightforward substitutions, we encounter qualitatively interesting differences in both cases.

1.2 On the synthesis of reliable quantum organisms

As progress was being made on the theory of classical fault-tolerance in the 1980s, a new frontier for computation began to emerge with the suggestion that some mathematical problems may admit more efficient solutions if mapped to *quantum*, rather than classical, dynamics [Deu85; Fey85]. Interest in quantum computers was further heightened in the following decade as more concrete examples of this quantum advantage were discovered [Sho94; Gro96]. At the same time, it became clear that if these algorithms were to be realized experimentally, methods for the practical control and manipulation of quantum degrees of freedom needed to be improved substantially.

The requirement for precision was relaxed significantly by Shor in 1996 with the proposal of a scheme for quantum fault-tolerance [Sho96] and subsequent improvements by Aharonov et al. [AB97]. Today, quantum fault-tolerance is an active area of research, with experimental research striving for more perfect quantum control, and theoretical research focused on developing fault-tolerant constructions with higher tolerance for imperfection. The convergence of the experimental and theoretical approaches appear vital to the promise of large-scale quantum computation [Pre98; KLZ98; Ter15; CTV17]. In contrast, hardware-level fault-tolerance has largely been eschewed in favor of error correction and error detection at the software-level in classical computers. This is largely a result of the overwhelming reliability of modern transistor technology, which by some estimates have soft error rates of the order of 10^{-21} errors per operation [WA08]. It is for these reasons that recent fault-tolerance results have focused on quantum models of computation.

More specifically, the focus has been on fault-tolerance in the *quantum circuit model* of computation [Deu89]. As with its classical counterpart, quantum circuit-based fault-tolerance works through the construction of a more perfect effective logical circuit using noisy physical gates and degrees of freedom [CTV17]. However, provably equivalent alternatives to the quantum circuit model have been proposed, including those based on measurement [RBB03; Bri+09], the quantum adiabatic theorem [Far+00; Far+01; AL18], and quantum walks [Lov+10; CGW13] among others. One problem for many of these other models of quantum computation is that it is less clear how they can be made fault-tolerant. Of course, one can embed these models in a fault-tolerant circuit-based quantum computer, but there are advantages to studying error correction that is native to these models. For one, native error correction schemes may prove easier to implement in the near term. Additionally, native error correction may eventually be used in conjunction with quantum circuit-based error correction.

In Chapter 3, we study a model of quantum computation composed from noisy subroutines which departs significantly from the traditional circuit-based fault-tolerance, with redundancy in time rather than (Hilbert) space (Q1). This is motivated by recent work on the quantum signal processing (QSP) subroutine and its generalization, the quantum singular value transform [Gil+19], which has been observed to capture many of the known speedups attributable to quantum processing [Mar+21]. In this way, the QSVT may be seen as the basis for a new model of quantum computation with favorable semantic properties [RC23]: instead of a quantum computation being composed of simple noisy gates, we can imagine the computation composed of quantum subroutines. In keeping with our program of studying error correction and fault-tolerance in less standard models of computation, we develop a deeper understanding of the propagation of gate-level errors to the level of the subroutine; in doing so, we develop an error reduction procedure that is native to QSP.

1.3 On the efficiency of reliable organisms

We have so far addressed the first question posed, that of when fault-tolerance is possible in less conventional computational models, with many examples in the affirmative. We now turn to the second question: the cost of fault-tolerance (Q2). In other words, we would like to study when fault-tolerance is not only possible, but desirable.

It was noticed by von Neumann [Neu56] that an essential feature of the interleaved construction was that a fault-tolerant circuit required a larger number of basic units and necessarily proceeded slower than a non-fault-tolerant circuit. Both of these factors contribute to what we term the *overhead* required to maintain fault-tolerance and have been features of all subsequent fault-tolerant constructions: classical [HW91; EP98; ES03] and quantum [Sho96; AB97; CTV17]. This overhead has been shown to be an essential feature of fault-tolerance itself: first, for noisy classical formulas by Pippenger [Pip88], later by Feder for noisy classical circuits [Fed89], and later by Kempe et al. for noisy quantum circuits [Kem+08].

To sketch the origin of this overhead, consider the goal of building a reliable organism with some bounded overall error rate. Without fault-tolerance, the accuracy of each component of the computation must scale in inverse proportion to the total number of components—for simplicity here, we assume that errors occur independently and the failure of any single component results in an erroneous final result. This comes at the cost of some *resource*, as a price often has to be paid to increase the reliability of each basic component. Fault-tolerance lifts the accuracy requirement from one that is inversely proportional to the number of basic components to one that is constant as long as it falls below a fixed threshold. Within this fault-tolerant regime, we may reduce the rate of logical error through increased use of redundancy. However, as we have seen, maintaining the system in this fault-tolerant regime comes at its own cost: this is the overhead with which we are concerned. In both approaches—the fault-tolerant and the non-fault-tolerant—reliability comes at a cost. The question of which approach is preferred can be answered with a careful analysis of the fault-tolerance overhead.

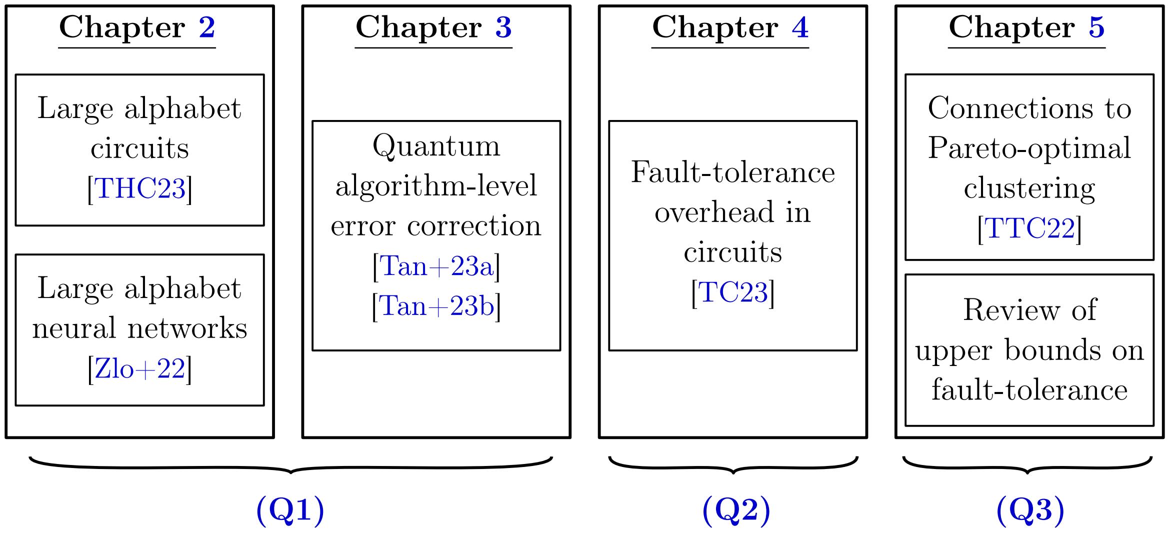
The question of when hardware-level fault-tolerance is preferred to non-fault-tolerant ones has already been implicitly answered in the case of classical and quantum computation; while modern classical computers do not use hardware-level fault-tolerance, proposed quantum computers rely heavily on fault-tolerant techniques. This is a result of the relative difficulty in constructing a reliable classical gate compared with the difficulty in constructing a reliable quantum gate. We can attempt to navigate this trade-off in cases where the choice is less clear. Previous attempts to perform a fine-grained comparison of non-fault-tolerant to fault-tolerant constructions by Thaker et al. [Tha+05; Tha+08] and Impens [Imp04] used a recursive fault-tolerant design in which overhead scaling is polylogarithmic and constants are difficult to obtain owing to the fact that the size of the resulting fault-tolerant circuits grow exponentially with the level of recursion.

In Chapter 4, we perform a precise accounting of the cost of fault-tolerance in a non-recursive fault-tolerance scheme allowing us to more precisely determine scenarios where the fault-tolerant construction is more efficient (Q2). Notably, in our model which uses an error correction circuit based on a locally correctable error correction code, our overhead cost is strictly logarithmic. This work can be seen both as a prescription of when fault-tolerant design may be preferable, and taking a broader view, may be understood to describe when fault-tolerance may naturally emerge, i.e. as the result of a preference for efficiency.

1.4 On the relationship between information theory and fault-tolerance

Though von Neumann sought a thermodynamical treatment of noisy information processing in the manner developed by Shannon for noisy communication in the years prior [Sha48], the original lecture series fell short of this goal by his own admission [Neu56]. The connections to Shannon's information theory have been strengthened in the ensuing decades. Perhaps the most straightforward extension can be found in a 1963 monograph by Winograd and Cowan [WC63], who defined a "computation capacity," in analogy with Shannon's channel capacity. More recently, a line of argument using information-theoretic quantities lead to *negative* fault-tolerance results which provided non-constructive upper bounds on fault-tolerance thresholds. Broadly, these non-constructive arguments focus on tracking a quantity that is provably monotonic in depth in a noisy formula or circuit. This line of argument can be traced back to Pippenger in 1988 who showed the monotonicity of an information-theoretic quantity to provide non-trivial upper bounds on fault-tolerant formulas [Pip88]; this was soon extended for more general circuits by Feder in 1989 [Fed89], and further improved by Evans and Schulman in 1999 [ES99]. Due to their generality, such information-theoretic monotones have since found applicability in a large number of settings including information flow in biological networks, probabilistic cellular automata, and in the analysis of statistical mechanical phase transitions among others—a recent thesis by Makur provides a comprehensive review of such techniques [Mak19].

In Chapter 5, we study another connection between noisy computation and information theory (Q3). While in previous chapters we have considered simple error correcting codes subject to symmetric noise, in this work, we consider the problem of efficiently encoding a random variable through an arbitrary noisy channel. Motivated by recent effective information-theoretic descriptions of neural networks [Sax+19], we consider a particular quantification of efficiency and noise known as the Deterministic Information Bottleneck [SS17]. We develop a tool for finding these optimal encodings, i.e. Pareto-optimal encodings, which may be used to design error correcting codes tailored to arbitrary noise channels. We conclude Chapter 5 with a review of existing negative fault-tolerance results which are based on information theoretic bounds. While we do not present any new results, by distilling existing upper bounds on fault-tolerance thresholds into their key conceptual components, we suggest a path towards new tight fault-tolerance upper bounds on circuits.


Chapter 2
Large alphabet circuits [THC23]
Large alphabet neural networks [Zlo+22]
Chapter 3
Quantum algorithm-level error correction [Tan+23a] [Tan+23b]
Chapter 4
Fault-tolerance overhead in circuits [TC23]
Chapter 5
Connections to Pareto-optimal clustering [TTC22]
Review of upper bounds on fault-tolerance
(Q1)
(Q2)
(Q3)
Figure 1.1: Roadmap of the thesis and correspondence between Chapters and (Q1–3).

1.5 Roadmap

This thesis is composed of a collection of results motivated by the guiding philosophy that noisy computation, broadly construed, can be made more perfect through hardware-level fault-tolerant design, and is guided by the three questions (Q1–3) set out at the beginning of the introduction. The overall structure of the thesis is sketched in Fig. 1.1.

In Chapter 2, we describe fault-tolerance results for two new large alphabet models of computation.

The first of which considers formula-based computation over larger alphabets of size $q > 2$. We show that for logical alphabets of size $\ell = q$ the threshold for denoising with q-ary majority gates subject to q-ary symmetric noise with error probability ε is strictly larger than that possible in the Boolean case, and is possible as long as signals remain distinguishable, i.e. $\epsilon < (q-1)/q$, in the limit of large fan-in $k \to \infty$. Interestingly, we observe the existence of multiple 'phase transitions' in the error correction procedure which are not present in the previously considered Boolean models. We also provide an example where $\ell < q$, showing that reliable Boolean computation can be performed using 2-input ternary logic gates subject to symmetric ternary noise of strength $\varepsilon < 1/6$ by using the additional alphabet element for error signaling. The existence of this denoising threshold requires the use of a proof technique based on a novel monotone which is satisfied in the fault-tolerant regime.

The second demonstrates an alternate path towards large alphabet fault-tolerance in a neural network based architecture. In this demonstration, we numerically study analog error correction code known as the grid code which has been observed in the mammalian cortex. We show that it is possible to both perform error correction and perform computation on data encoded in this grid code assuming access to primitives resembling neurons in artificial neural networks.

In Chapter 3, we take a brief detour from the more straightforward application of von Neumann's construction to describe a new method for correcting systematic errors in the recent quantum singular value transformation subroutine for quantum algorithms. Here, we consider a model of QSP with generic perturbative noise in the signal processing basis, and present a novel diagrammatic notation useful for analyzing such errors. To demonstrate our technique, we study a specific coherent error, that of under- or over-rotation of the signal processing operator parameterized by $\epsilon \ll 1$. For this coherent error model, it is shown that while Pauli Z-errors are not recoverable without additional resources, Pauli X and Y errors can be arbitrarily suppressed by coherently appending a noisy 'recovery QSP' without the use of additional resources or ancillas. Furthermore, through a careful accounting of errors using our diagrammatic tools, we provide an upper- and lower-bound on the length of this recovery QSP operator. We anticipate that the perturbative technique and the diagrammatic notation proposed here will facilitate future study of generic noise in QSP and quantum algorithms.

In Chapter 4, we address the question of when fault-tolerance may be preferred, i.e. as it provides a more economical route to reliability when compared with a naive construction. We present a general framework to account for this overhead cost in fault-tolerance schemes based on constant-depth error correction circuits; this accounting is used to compare fault-tolerant to non-fault-tolerant approaches for computation in the limit of small logical error rates. We determine explicit boundaries at which fault-tolerant designs become more efficient than designs that achieve comparable reliability through direct consumption of resources. We find that the fault-tolerant construction is always preferred when high reliability is required in cases where the resources required to construct a basic unit grows faster than $\log(1/\epsilon)$ asymptotically for small ϵ. In addition to suggesting when hardware-level fault-tolerance may be preferred, our result suggests that certain selection pressures may favor principles of fault-tolerance in systems that perform emergent information processing.

Finally, in Chapter 5, we discuss connection between fault-tolerance results and information theory in two parts.

In the first part, we use an information theoretic objective to move beyond simple symmetric error channels and develop a tool for finding optimal error correcting codes tailored to arbitrary noise channels. We define optimality information-theoretically in the sense of the Deterministic Information Bottleneck, which can equivalently be understood as a clustering algorithm. At the heart of the problem is a trade-off between the fidelity and size of the learned representation. Our goal is to map out and study the Pareto frontier that quantifies this trade-off. We focus on the optimization of the DIB objective over the space of hard clusterings. We present an algorithm for mapping out the Pareto frontier of the DIB trade-off that is also applicable to other two-objective clustering problems and use our algorithm to map the DIB frontier of a number of compression tasks. For example, we consider the channel that maps each character of the Latin alphabet to the next in a body of natural English text. This is clearly a noisy channel, as the next letter cannot in general be predicted with certainty. Our algorithm addresses the problem of how to best group these letters in order to preserve as much information as possible when acted on by this channel, and can be used to design error correcting codes over the channel.

In the second part, we review existing negative fault-tolerance results and place them in context. We point towards point towards new promising directions for improved fault-tolerance upper bounds on circuits based on novel measures of information. In addition, we review connections between fault-tolerance and related problems such as the problem of maintaining a signal in a graph of noisy relays, and connections to statistical physics.

Finally we provide some concluding remarks in Chapter 6. Though the work in this thesis merely provides glimpses into what may be considered a theory of fault-tolerant information processing, it is the author's belief that a such a theory would be a worthwhile endeavor. This may be especially true in the case of less conventional models of computations; for example in systems that perform information processing in an emergent manner where the noise of individual components cannot be inexpensively reduced.

1.5.1 Bibliographical note

The work described in Chapters 2 to 5 have been included nearly verbatim from published or otherwise submitted papers. Since these papers were written in collaboration with coauthors, in the interest of transparency, we detail the specific contributions of the author.

Chapter 2 draws from two published or submitted works. The first, based on [THC23], was on the fault-tolerance in larger alphabet formula-based computation. This paper was initially proposed by the author who wrote and proved the theorems for the $\ell = q$ case. Matthew Ho proposed the error correction gate for the case of $\ell = 2$, $q = 3$ and wrote the majority of that section in discussions with the author. All of this was done in collaboration and with helpful feedback from Isaac Chuang. The second, based on [Zlo+22], was on fault-tolerance in larger alphabet neural network based computation. This paper was initially proposed by the author who suggested the use of grid codes to combat analog noise. Alexander Zlokapa designed the fault-tolerant computation scheme and decoder circuit, performed the theoretical and numerical analysis of the model, and performed the majority of the writing related to the fault-tolerant neural network. John Martyn performed the analysis and the majority of the writing related to the model synaptic error. The overall project was performed with feedback and suggestions from the senior authors: Ila Fiete, Max Tegmark, and Isaac Chuang.

Chapter 3 draws from two published or submitted works. The first, based on [Tan+23a], was on error correction at the algorithm level. This paper was written primarily by the author in constant discussion with the coauthors Yuan Liu, Minh Tran, and Isaac Chuang who initially suggested the term “algorithm level error correction (ALEC)”. The second, based on [Tan+23b], provided a more detailed analysis of the first result. The author proved the main technical theorems, performed numerical analysis, and performed the majority of the writing. Again, all of these tasks were done through constant discussion with the same coauthors.

Chapter 4 draws from [TC23]. This was initially conceptualized by the author who performed the technical analysis and performed the majority of the writing. All of this was done in collaboration and with helpful feedback from Isaac Chuang.

Chapter 5 draws from [TTC22]. The project was initially proposed by Max Tegmark as a generalization of [TW19]. The author designed the algorithms, proved theorems, performed numerics, and performed the majority of the writing. All of this was done in collaboration and with helpful feedback from both Max Tegmark and Isaac Chuang.

Chapter 2

Fault-tolerance in classical models of computation

While the study of fault-tolerance was initially motivated by the understanding of robustness of information processing in biological systems [Neu56], the work up until now has focused on the study of noisy Boolean circuit-based models of computation. In keeping with the theme of studying fault-tolerance in unconventional models of computation (Q1), this Chapter focuses on fault-tolerance results in classical models of computation that go beyond Boolean circuits. As idealized binary systems rarely exist in nature, we choose to study examples of noisy models of computation over larger alphabets. In addition to the choice of alphabet size, we must also make choices in regard to the basic components, the method of composing these basic components, and the model of noise. We note that while the larger alphabet does not confer any additional computation power over Boolean logic in the noiseless setting, with one being easily simulated by the other, the introduction of noise makes this reduction less straightforward.

In Section 2.1[1], in an attempt to isolate the effect of increasing alphabet size such that we can nominally compare threshold rates with prior work, we maintain the formula-based

[1]This has appeared as "On reliable computation over larger alphabets," by Andrew K. Tan, Matthew Ho, and Isaac L. Chuang in arXiv preprint arXiv:2306.13262 (2023) [THC23].

model of composition and consider the analogous model of symmetric noise. This work illustrates that nominal error rates can be strictly improved for computation over larger alphabets using large alphabet majority gates. We also investigate the alternative, that of reducing a large alphabet system into one that is Boolean; in this case, we use elements of the physical alphabet for error signaling—again showing that noisy large alphabet computation cannot be trivially reduced to noisy Boolean computation. In Section 2.2[2], we adopt a more biologically-inspired model of noisy computation based on noisy artificial neural networks. In this case the basic components are artificial neurons with a standard ReLU non-linearity composed in feed-forward neural networks. Neurons are subject to analog additive Gaussian noise which motivates the encoding of data in a biologically-inspired error correcting code. Using the von Neumann construction we again demonstrate fault-tolerance native to this less conventional model of computation.

2.1 Positive results for formula-based computation over larger alphabets

2.1.1 Introduction

The problem of performing computation with noisy components was first studied by von Neumann in 1952 [Neu56]. Motivated by understanding the robustness of information processing in biological systems, he proposed a toy model of Boolean computation using noisy circuits in which Boolean-valued functions were computed through composition of basic gates with bounded fan-in. Interestingly, von Neumann showed that this model admitted a fault-tolerant regime: for circuits with noise parameterized by strength ε, there exists a threshold β for which noisy circuits composed of gates subject to error $\varepsilon < \beta$ could simulate a noiseless

[2]This has appeared as "Biological error correction codes generate fault-tolerant neural networks," by Alexander Zlokapa, Andrew K. Tan, John M. Martyn, Ila R. Fiete, Max Tegmark, and Isaac L. Chuang in arXiv preprint arXiv:2202.12887 (2022) [Zlo+22].

circuit of size N to precision ε_{L} with a modest overhead of $\mathrm{polylog}(N/\varepsilon_{\mathrm{L}})$ in the number of gates [NC10]. The key observation made by von Neumann was that by encoding data in an error correcting code and interleaving computation with error correction, the error of the Boolean circuit's output could be bounded away from $1/2$ in a manner independent of circuit depth; such a circuit computes a function *reliably*, in a way which can be made mathematically precise. The positive result of [Neu56] has since been elegantly tightened for formulas (i.e. circuits of fan-out one) [HW91; EP98; ES03], as summarized in Table 2.1. Despite the use of formulas, a qualitatively different setting than circuits, these positive results for formulas nonetheless follow the essence of von Neumann's original construction, whose hallmark is the interleaving of distinct computation and error correction stages.

Given a model of noisy computation, i.e. a set of elementary operations over a finite alphabet along with a specification of the error process, we distill the *von Neumann construction* for noisy formula-based fault-tolerance into the following two-step process:

1. Show that a given denoising operation (e.g. majority) has $\ell \geq 2$ stable fixed-points in the probability simplex over the physical alphabet up to $\varepsilon < \beta'$ where ε parameterizes the error model: each fixed-point along with its basin of attraction under this denoising operation, is associated with a logical state.

2. Exhibit a set of gates capable of performing operations on the logical states, maintaining their outputs in the correct basins of attraction for $\varepsilon < \beta \leq \beta'$.

We call β' the *denoising threshold* of this particular construction, and β its *computation threshold*.

Though von Neumann's original analysis was limited to the toy model of noisy Boolean circuits, inspired by biological systems, he left open the possibility for reliable computation over non-binary alphabets, which have found renewed interest in certain modern computational models. Specifically, multivalued logic [Smi81; Smi88] has recently been studied both for providing more natural embeddings for certain computational problems [EFR74; SKR95; Dub99; Rin14], and for admitting efficient implementations in new materials [MON89; Dub99; JKC21]. More recently still, multi-level quantum systems, i.e. qudits, have been shown to offer promising avenues (beyond qubits) for fault-tolerant quantum information processing [BRS17; Wan+20].

Motivated by moving beyond Boolean alphabets, we make initial steps towards extending formal positive fault-tolerance results using the von Neumann construction for computation over larger alphabets. Formally, we define an *alphabet* of size $q > 2$ to be the set $[q] \equiv \{0, 1, \ldots, q-1\}$; and we refer to elements of the alphabet set, either physical or logical, as *characters*. A number of qualitative differences emerge in the study of larger alphabet fault-tolerance, with two specific opportunities (and matching challenges) arising.

First is the opportunity allowed by larger alphabets to recover from higher error rates, compared with the case for a binary alphabet. This is due to the simple fact that any particular error is increasingly less likely to overwhelm the signal as q grows. However, it is not manifest that such error correctability extends to allow reliable *computation* over larger alphabets for gates with bounded fan-in.

Second, when $q > 2$, logical computation may be performed without using the entire alphabet — in other words, the logical alphabet size ℓ can be strictly smaller than the physical alphabet size q. As a result, members of the physical alphabet that are not part of the logical alphabet can be used to signal that an error has occurred. Whether this opportunity for "error signaling" can improve fault-tolerance thresholds, however, has never before been rigorously established.

Here, we perform an analysis of the thresholds for reliable computation using gates subject to q-ary symmetric noise of strength ε, i.e. where an error occurs with probability ε, leaving the output in any of the $q-1$ erroneous states with equal probability. We study computation over physical alphabets of size $q > 2$ using the two step von Neumann construction. We

Table 2.1: Overview of positive fault-tolerance results for gates subject to q-ary symmetric noise. The first positive results were shown for circuits by von Neumann, who analyzed computation with both {MAJ-[2, 3], NAND} and {NAND} gate sets [Neu56]. These results were subsequently tightened: first for the {MAJ-[2, 3], NAND} gate set by Hajek and Weller [HW91], and for the more challenging fan-in 2 gate set {NAND} by Evans and Pippenger [EP98]. Evans and Schulman demonstrated tight thresholds for {MAJ-[2, k], XNAND}, k odd [ES03].

Denoising Gate	Denoising Threshold	Computation Gate(s)	Computation Threshold	Reference
MAJ-$[2,3]$	$1/6$	NAND	≈ 0.0073	[Neu56]
NAND	-	NAND[1]	≈ 0.0107	[Neu56]
MAJ-$[2,3]$	$1/6$	XNAND	$1/6$	[HW91][2]
NAND	$(3-\sqrt{7})/4$	NAND	$(3-\sqrt{7})/4$	[EP98][2]
MAJ-$[2,k]$, k odd	$\frac{1}{2} - 2^{k-2} \Big/ k\binom{k-1}{\frac{k-1}{2}}$	XNAND	$\frac{1}{2} - 2^{k-2} \Big/ k\binom{k-1}{\frac{k-1}{2}}$	[ES03][2]
MAJ-$[q,k]$, k odd	$\frac{q-1}{q}\frac{C^{[q,k]}-1}{C^{[q,k]}}$	ADD^q, MUL^q	$\frac{q-1}{q}\frac{C^{[q,k]}-1}{C^{[q,k]}}$	Section 2.1.3[3]
MAJ-$[q,3]$, q prime	$\frac{q-1}{q(q+1)}$	ADD^q, MUL^q	$\frac{q-1}{q(q+1)}$	Section 2.1.3
DEN	$1/6$	ENAND[2]	$1/6$	Section 2.1.4

[1] Result is for circuits.
[2] Universal for binary computation.
[3] Computation threshold is asymptotic for $k \to \infty$ but conjectured to hold generally.

consider two main settings: one where $\ell = q$, and one for which $\ell < q$. For the first setting, where computation is performed with the logical alphabet being the same size as the physical alphabet:

- We generalize the results of Evans and Schulman [ES03] to $q > 2$ alphabets. In particular, we find that the denoising threshold with q-ary majority gates subject to q-ary symmetric noise of strength ε is improved over larger physical alphabet sizes (Lemma 8, in Section 2.1.3).

- We show that the set of functions generalizing the Boolean XOR over larger alphabets can be reliably computed up to the previously shown denoising threshold (Theorem 1, in Section 2.1.3).

- We show that reliable universal computation is possible up to $(q-1)/q$ asymptotically as $k \to \infty$, and is possible up to the previously shown denoising threshold for finite fan-in $k = 3$ and q prime in Section 2.1.3).

For the second setting, in which computation is performed with a subset of the available physical alphabet such that some of the physical alphabet can be employed to signal errors and help with denoising:

- We define error signaling protocols for $\ell < q$ and employ these to demonstrate that denoising of two logical states over a physical ternary alphabet using 2-input gates subject to ternary symmetric noise is possible for $\varepsilon < 1/6$ (Lemma 14, in Section 2.1.4.).
- Using error signaling we prove that universal Boolean ($\ell = 2$) computation using 2-input gates subject to ternary symmetric noise is possible up to the denoising threshold when using $q = 3$ (Theorem 2, in Section 2.1.5).

We present these results beginning with definitions and framework in Section 2.1.2, followed by the large-alphabet computation $\ell = q > 2$ scenario in Section 2.1.3 and the embedded Boolean computation $\ell < q$ scenario in Section 2.1.4. Possible extensions and concluding observations are discussed in Section 2.1.6.

2.1.2 Definitions and framework

First, we make precise the set of functions that can be computed. Recall from the theory of universal algebras:

Definition 1 (Clone)**.** A *clone* is a set C of finitary operations such that

i) C is closed under composition.

ii) C contains all projection functions, i.e. $f_i \in C$ such that $f_i(X_1, \ldots, X_n) = X_i$.

The clones of Boolean functions have been completely classified by Post [Pos41], and classifications exist for larger alphabets [Lau06]. Since clones are closed under composition, we are often interested in the clone's generating set, i.e. a minimal set of gates capable of computing all functions in the clone through composition. At one limit, we have the complete clone:

Definition 2 (Universal q-ary computation)**.** We say a particular set of gates is universal for q-ary computation if the clone that it generates contains all n input functions $f : [q]^n \to [q]$.

In the previous literature on Boolean computation, the 2-input NAND gate [Neu56; EP98], being itself universal, has been used both to perform computation as well as construct the denoising operation. This becomes more complicated over larger alphabets, as certain non-universal clones may admit larger thresholds (e.g. Section 2.1.4).

Additionally, in the setting of larger alphabets, a distinction can be made between the *physical* and *logical* alphabets. Specifically, operating over a physical alphabet of size q, we may in general choose a smaller number of logical states over which to perform reliable computation through the use of error correction and error detection. We study an example where the logical alphabet is strictly smaller than the physical alphabet in Section 2.1.4.

Next, we formalize the notion of reliable computation in the presence of noise.

Definition 3 (δ-reliability)**.** Let $f : [q]^n \to [q]$ be a q-ary function. We say a noisy circuit F_ε computes f *δ-reliably* if the probability of error over all inputs is at most δ, or equivalently

$$\min_{x_1,\ldots,x_n \in [q]} \Pr[F_\varepsilon(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)] \geq 1 - \delta. \tag{2.1}$$

For large alphabet computation (i.e. $q > 2$), a single parameter is no longer sufficient to characterize the noise in general; however, we elect to focus on gates subject to q-ary symmetric noise which is characterized by a single parameter:

Definition 4 (q-ary symmetric noise)**.** We say a k-input gate over alphabet size q, $g : [q]^k \to [q]$ is *ε-noisy*, denoted g_ε, if for all inputs $x_1, \ldots, x_k \in [q]$, we have

$$\forall y \in [q], y \neq g(x_1, \ldots, x_k) \implies \Pr[g(x_1, \ldots, x_k) = y] = \frac{\varepsilon}{q-1}. \tag{2.2}$$

There are several reasons for this choice of error model. Firstly, it is both a natural generalization of the binary symmetric noise studied in [Neu56; EP98; ES99; ES03], and the polar opposite of binary symmetric noise which concentrates all errors on a single erroneous element of the alphabet. On one hand, the property of maximally spreading out errors across the alphabet is the best case noise in the sense that it results in the highest nominal denoising and computation thresholds. On the other hand, symmetric noise is also the worst case noise in the sense that, conditioned on an error having occurred, it provides no information about the correct signal.

2.1.3 Large alphabet computation

In this Section, we analyze the threshold for reliable computation over larger alphabets i.e. $q > 2$. Following the von Neumann construction, we first derive, in Section 2.1.3, the denoising thresholds of a generalized majority gate, and then in Section 2.1.3 determine the set of functions which can be reliably computed given these denoised inputs for large k. Finally, in Section 2.1.3, we provide lower-bounds on the computation threshold for the special case of $k = 3$ and q prime.

First, let us only consider inputs subject to q-ary symmetric noise:

Definition 5 (q-ary symmetric a-noisy encoding)**.** We say a random variable X is an *a-noisy encoding* of $\hat{x}$ if

$$\Pr[X_i = x] = \begin{cases} 1 - a, & \text{if } x = \hat{x}_i \\ \frac{a}{q-1}, & \text{otherwise} \end{cases} . \tag{2.3}$$

Recovery threshold

For computation with ε-noisy gates, we find that the k-input majority gate over alphabet of size q is the optimal denoising gate (in the maximum-likelihood sense) in the case where the logical alphabet size $\ell = q$. This generalized majority gate, MAJ-[q, k]: $[q]^k \to [q]$, outputs the mode of its inputs; ties are broken by choosing the value of the mode with the lowest input index, resulting in a deterministic balanced gate.

Suppose that input X is subject to symmetric noise of strength a, then applying a noiseless restoring gate to k independent copies, $\{X_i\}_{i=1}^{k}$, of the noisy signal yields an output MAJ-[q, k]$_0(X_1, \ldots, X_k)$ with error

$$m_0^{[q,k]}(a) \equiv \sum_{\ell=0}^{k} \binom{k}{\ell} c_\ell^{[q,k]} (1-a)^\ell a^{k-\ell}, \tag{2.4}$$

where $c_\ell^{[q,k]} \in [0,1]$ denotes the fraction of assignments over an alphabet of size q to the restoring gate taking k inputs, exactly ℓ of which are correct, resulting in an incorrect output. Consider the limiting example of a binary alphabet: all assignments with $\ell < \lfloor k/2 \rfloor$ (assuming k odd) and therefore

$$c_\ell^{[2,k]} = \begin{cases} 1, & \text{for } \ell < \lfloor \frac{k}{2} \rfloor \\ 0, & \text{otherwise} \end{cases}. \tag{2.5}$$

Over larger alphabets, a strict majority is not required for the plurality operation to succeed, therefore allowing stronger denoising (i.e. $c_\ell^{[q,k]} < 1$ for some $\ell < \lfloor k/2 \rfloor$).

For an ε-noisy restoring gate, we have

$$m_\varepsilon^{[q,k]}(a) = \left(1 - \frac{\varepsilon}{q-1}\right) m_0^{[q,k]}(a) + \varepsilon\left(1 - m_0^{[q,k]}(a)\right). \tag{2.6}$$

Remark 6 (Properties of $c_\ell^{[q,k]}$)**.** Here, we take note of a few properties of $c_\ell^{[q,k]}$.

1. When a majority of the inputs are correct, no assignment of the remaining inputs results in an error and therefore (assuming k odd) $c_\ell^{[q,k]} = 0$ for all $\ell > \lfloor k/2 \rfloor$.

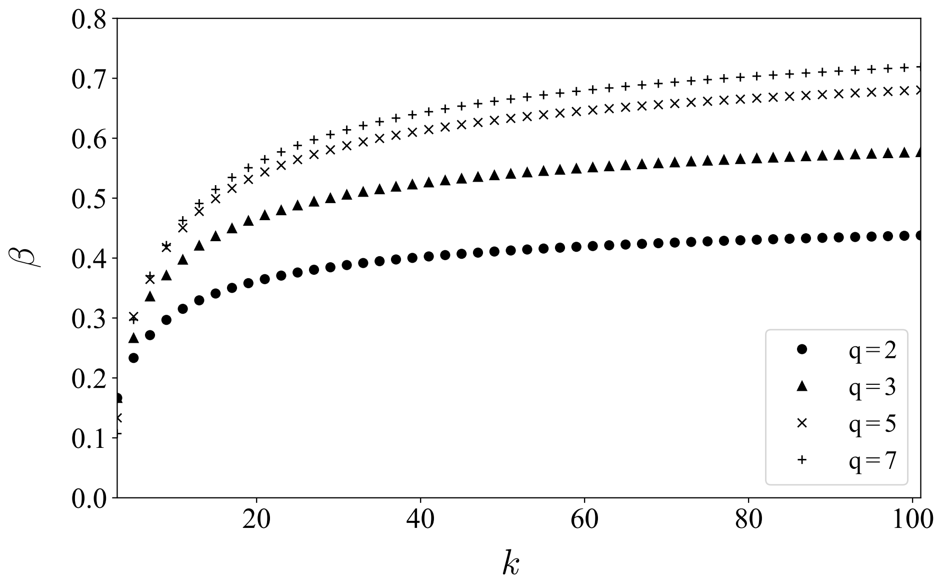


Figure 2.1: Plot of denoising threshold, β, using the MAJ-$[q,k]$ gate as a function of fan-in for different alphabet sizes. The plotted threshold values are achieved for symmetrically noisy inputs and are known to be tight for $q = 2$ [ES03]. In the large-k limit, denoising thresholds approach $(q-1)/q$.

2. Furthermore, if fewer than a $1/q$ fraction of inputs are correct, all assignments of the remaining inputs results in error, therefore $c_\ell^{[q,k]} = 1$ for $\ell < k/q$.

3. Finally, $c_{\ell+1}^{[q,k]} \leq c_\ell^{[q,k]}$.

Remark 7 (Maximally mixed fixed-point)**.** For all ε, the error rate has a fixed-point at $a = (q-1)/q$.

$$m_0^{[q,k]}\left(\frac{q-1}{q}\right) = \frac{1}{q^k}\sum_{\ell=0}^{\lfloor k/2\rfloor}\binom{k}{\ell}c_\ell^{[q,k]}, \tag{2.7}$$

$$= \frac{q-1}{q}. \tag{2.8}$$

The final equality is a consequence of the restoring gate being balanced and therefore must output each value for exactly $1/q$ of the q^k possible assignments of inputs. Examining Eq. (2.6), we see that the fixed-point of $a = (q-1)/q$ is preserved for all ε.

Lemma 8 (Lower-bound on MAJ-$[q,k]$ denoising threshold)**.** For $q \geq 2$ and odd k, let

$$C^{[q,k]} \equiv \frac{k}{q^{k-1}} \sum_{\ell=0}^{\lfloor k/2 \rfloor} \binom{k-1}{\ell} (c_\ell^{[q,k]} - c_{\ell+1}^{[q,k]})(q-1)^{k-\ell-1}. \tag{2.9}$$

If $\varepsilon < \frac{q-1}{q}\frac{C^{[q,k]}-1}{C^{[q,k]}}$, the following hold:

i) $a \in [0, \frac{q-1}{q}) \implies \exists \mu_\varepsilon^{[q,k]} \in [0, \frac{q-1}{q})$ such that $\lim_{n\to\infty} m_\varepsilon^{[q,k]n}(a) = \mu_\varepsilon^{[q,k]}$; and

ii) $a \in (\frac{q-1}{q}, 1] \implies \exists \nu_\varepsilon^{[q,k]} \in (\frac{q-1}{q}, 1]$ such that $\lim_{n\to\infty} m_\varepsilon^{[q,k]n}(a) = \nu_\varepsilon^{[q,k]}$.

Proof. The function $m_\varepsilon^{[q,k]}(a)$ has at most three fixed-points. We show that for all $\varepsilon < \frac{q-1}{q}\frac{C^{[q,k]}-1}{C^{[q,k]}}$, the fixed-point at $a = (q-1)/q$ given by Remark 7 is unstable.

Taking the derivative,

$$\begin{aligned}
\frac{dm_0^{[q,k]}}{da} &= \frac{d}{da} \sum_{\ell=0}^{\lfloor k/2 \rfloor} \binom{k}{\ell} c_\ell^{[q,k]} (1-a)^\ell a^{k-\ell}, && (2.10)\\
&= \sum_{\ell=0}^{\lfloor k/2 \rfloor} (k-\ell)\binom{k}{\ell} c_\ell^{[q,k]} (1-a)^\ell a^{k-\ell-1} - \sum_{\ell=0}^{\lfloor k/2 \rfloor} \ell \binom{k}{\ell} c_\ell^{[q,k]} (1-a)^{\ell-1} a^{k-\ell}, && (2.11)\\
&= k \sum_{\ell=0}^{\lfloor k/2 \rfloor} \binom{k-1}{\ell} c_\ell^{[q,k]} (1-a)^\ell a^{k-\ell-1} - k \sum_{\ell=0}^{\lfloor k/2 \rfloor -1} \binom{k-1}{\ell} c_{\ell+1}^{[q,k]} (1-a)^\ell a^{k-\ell-1}, && (2.12)\\
&= k \sum_{\ell=0}^{\lfloor k/2 \rfloor} \binom{k-1}{\ell} (c_\ell^{[q,k]} - c_{\ell+1}^{[q,k]})(1-a)^\ell a^{k-\ell-1}. && (2.13)
\end{aligned}$$

Taking another derivative,

$$\begin{aligned}
\frac{d^2 m_0^{[q,k]}}{da^2} &= k \frac{d}{da} \sum_{\ell=0}^{\lfloor k/2 \rfloor} \binom{k-1}{\ell} (c_\ell^{[q,k]} - c_{\ell+1}^{[q,k]})(1-a)^\ell a^{k-\ell-1}, && (2.14)\\
&= k(k-1) \sum_{\ell=0}^{\lfloor k/2 \rfloor} \binom{k-2}{\ell} (c_\ell^{[q,k]} - 2c_{\ell+1}^{[q,k]} + c_{\ell+2}^{[q,k]})(1-a)^\ell a^{k-\ell-2}. && (2.15)
\end{aligned}$$

As a result, $m_\varepsilon^{[q,k]}(a) = a$ has three distinct solutions so long as

$$\left.\frac{dm_\varepsilon^{[q,k]}}{da}\right|_{a=(q-1)/q} = 1. \tag{2.16}$$

From Eq. (2.13),

$$\begin{aligned}\left.\frac{dm_0^{[q,k]}}{da}\right|_{a=(q-1)/q} &= \frac{k}{q^{k-1}} \sum_{\ell=0}^{\lfloor k/2 \rfloor} \binom{k-1}{\ell} (c_\ell^{[q,k]} - c_{\ell+1}^{[q,k]})(q-1)^{k-\ell-1}, \\ &= C^{[q,k]}.\end{aligned} \tag{2.17}$$

Therefore,

$$\left.\frac{dm_\varepsilon^{[q,k]}}{da}\right|_{a=(q-1)/q} = \left(1 - \frac{\varepsilon}{q-1}\right) \left.\frac{dm_0^{[q,k]}}{da}\right|_{a=(q-1)/q} - \varepsilon \left.\frac{dm_0^{[q,k]}}{da}\right|_{a=(q-1)/q}, \tag{2.18}$$

which shows that

$$\varepsilon < \frac{q-1}{q} \frac{C^{[q,k]} - 1}{C^{[q,k]}} \implies \left.\frac{dm_0^{[q,k]}}{da}\right|_{a=(q-1)/q} > 1, \tag{2.19}$$

and thus the Lemma holds. □

By setting $q = 2$ in Lemma 8 and using Eq. (2.5), we recover the Boolean denoising result of [ES03]. Furthermore, in the case of computation over the Boolean alphabet, this bound is tight; we note however that this is not necessarily true for $q > 2$, in which case stable fixed-points exist above the threshold of Lemma 8, but denoising is no longer possible for the full range $a \in [0, (q-1)/q)$ as the fixed-point corresponding to a maximally mixed $((q-1)/q)$-noisy fixed-point becomes stable. We will refer to the threshold of Lemma 8 as the denoising threshold, or more specifically, the point of transcritical bifurcation; this is in contrast with the ultimate saddle-node bifurcation after which point only one stable fixed-point remains (see Fig. 2.2). The positive results for various alphabet sizes q and fan-ins k are shown in Fig. 2.1.

We now generalize our denoising result to all input distributions (i.e. beyond q-ary

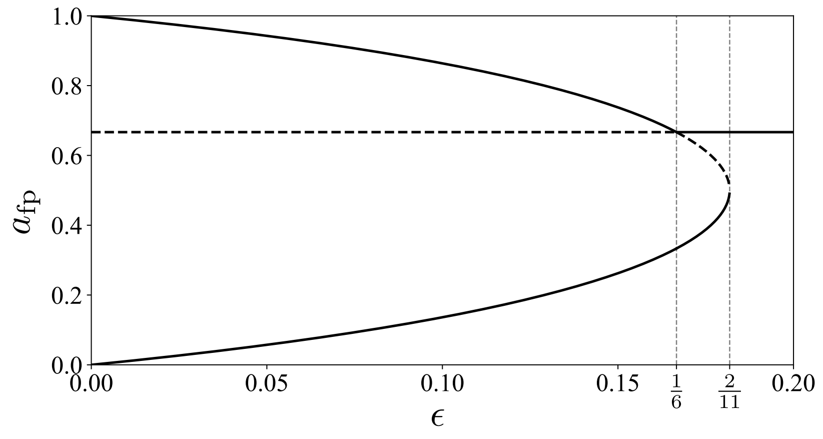


Figure 2.2: Diagram showing stable (solid lines) and unstable (dashed lines) fixed-points of the ε-noisy MAJ-$[3,3]$ gate for symmetrically a-noisy inputs. Note that at the denoising lower-bound of $\varepsilon = 1/6$ (Lemma 8) corresponds to a transcritical bifurcation, at which point the qualitative structure of the fixed-points changes, with the central $(2/3)$-noisy fixed-point becoming stable. Two stable fixed-points persist until the ultimate saddle-node bifurcation at $\varepsilon = 2/11$, corresponding to a discontinuous phase transition.

symmetric noise). Our approach is to analyze denoising as a discrete-time dynamical system.

First, we specify a categorical distribution over a q-ary alphabet as a point in the simplex

$$\Delta_q = \left\{ \vec{p} \in \mathbb{R}^q \middle| p_i \geq 0, \sum_{i=1}^{q} p_i = 1 \right\} \subset \mathbb{R}^q. \tag{2.20}$$

For independent and identically distributed (i.i.d.) q-ary input random variables X with distribution parameterized by vector $\vec{p} \in \Delta_q$, denote its output $Y_\varepsilon = \text{MAJ-[q, k]}(X_1, \ldots, X_k)$ with distribution parameterized by $\vec{q} \in \Delta_q$. Denote the map from input distributions to output distributions $\mathcal{M}^{[q,k]} : \Delta^q \to \Delta^q$ such that $\mathcal{M}(\vec{p}) = \vec{q}$, where we have omitted superscript $[q, k]$ for brevity.

In this picture, the fixed-points of Lemma 8 correspond to points $\vec{\chi}^{(i)} \in \Delta_q$ such that $\vec{\chi}^{(i)}_i = 1 - \nu_\varepsilon$ and $\vec{\chi}^{(i)}_j = \nu_\varepsilon/(q-1)$ for $j \neq i$, where again superscripts of $[q, k]$ and subscripts of ε have been omitted.

Further, let us denote by $\mathcal{R}^{(i)} \subset \Delta_q$, the regions in the probability simplex decoding to logical character i. Denoising using the MAJ-$[q, k]$ gate, we will show that it makes sense to define the regions as follows:

$$\mathcal{R}^{(i)} = \{\vec{p} \in \Delta_q \mid \forall j \in [q] \setminus \{i\},\ p_i - p_j > 0\}\,. \tag{2.21}$$

That is, $\mathcal{R}^{(i)}$ corresponds to the region in the probability simplex over q elements, Δ_q, where symbol $i \in [q]$ is the most likely character.

Our goal is to show that below the denoising threshold, all points in $\mathcal{R}_0^{(i)}$ flow towards their respective fixed-points $\vec{\chi}^{(i)}$ under repeated iteration of the map $\mathcal{M}$.

Lemma 9 (All points in $\mathcal{R}^{(i)}$ approach $\vec{\chi}^{(i)}$ upon repeated iteration of $\mathcal{M}$)**.** For $q \geq 2$, $k \geq 3$, and ε below threshold, all points $\vec{p} \in \mathcal{R}^{(i)}$ satisfy

$$\lim_{n\to\infty} \mathcal{M}^n(\vec{p}) = \vec{\chi}^{(i)}. \tag{2.22}$$

Proof. Without loss of generality, consider a q-ary random variable with distribution $\vec{p} \in \mathcal{R}_0^{(0)}$ with elements in sorted order (i.e. $i > j \implies p_i \geq p_j$). After one application of the map $\mathcal{M}$, we have $\vec{p'} = \mathcal{M}(\vec{p})$.

Note that p_0' is a convex combination of convex functions of the form

$$p_0^{k_0} p_1^{k_1} \dots p_{q-1}^{k_{q-1}}, \tag{2.23}$$

and is therefore itself jointly convex in $p_1, \dots, p_q$ and for fixed p_0 is maximized at the boundary $p_i = (1-p_0)/(q-1)$ for all $i > 0$. Intuitively, for fixed p_0, the minimum chance of confusion, and therefore maximum amplification, is achieved when all erroneous inputs are equally likely. As a result, the probability of error after denoising $\alpha' \equiv 1 - p_0'$ is minimized for the case of symmetric noise studied in Lemma 8,

$$\alpha' \geq m_\varepsilon^{[q,k]}(1-p_0). \tag{2.24}$$

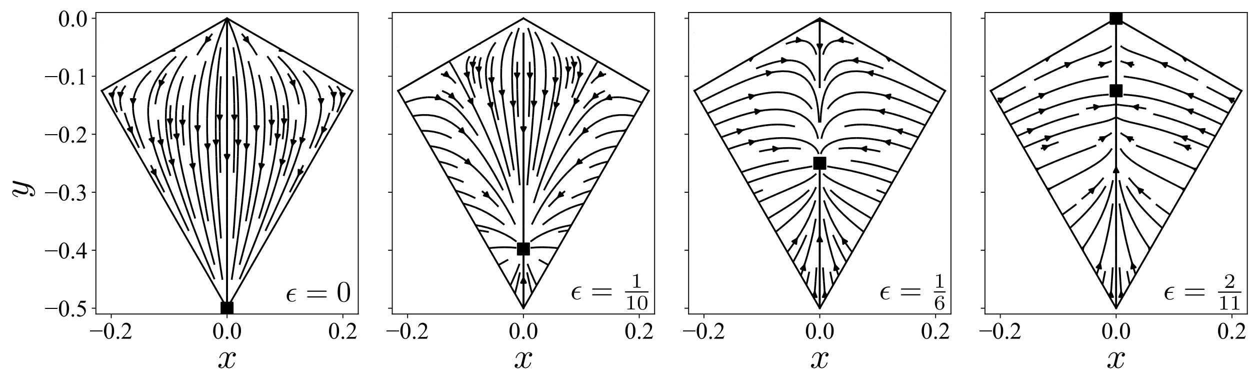


Figure 2.3: Streamlines of the vector field $\bar{\mathcal{M}}_\varepsilon(x, y) - (x, y)$ for various values of ε in the region $\bar{\mathcal{R}}^{(2)}$ — result is symmetric for $\bar{\mathcal{R}}^{(0)}$ and $\bar{\mathcal{R}}^{(1)}$ — along with stable fixed-points (square markers). Note that the fixed-point at the transcritical bifurcation (Lemma 8) $\varepsilon = 1/6$ is (1/3)-noisy (third plot). Between the transcritical bifurcation and saddle-node bifurcation (i.e. $1/6 < \varepsilon < 2/11$), denoising is still possible for a strict subset of the original region $\bar{\mathcal{R}}^{(2)}$ (fourth plot) owing to the emergence of a fourth stable fixed-point at the center of the simplex.

We can devise a complementary bound by noticing that, for fixed gap $\delta_1 \equiv p_0 - p_1$, the gap δ_1 is also convex in the region $\mathcal{R}^{(0)}$ since $p_0 > p_1$. Here, note that δ_1' is minimized for symmetrically noisy inputs $p_i = (1 - p_0)/(q - 1)$ for all $i > 0$. As a result, the gap after denoising is minimized for

$$\delta_1' = p_0' - p_1' \geq (1 - m_\varepsilon^{[q,k]}(1 - p_0)) - \frac{m_\varepsilon^{[q,k]}(1 - p_0)}{q - 1}, \tag{2.25}$$

$$\implies \alpha' = 1 - p_0' \leq m_\varepsilon^{[q,k]}(1 - p_0) - \left(p_1' - \frac{m_\varepsilon^{[q,k]}(1 - p_0)}{q - 1}\right). \tag{2.26}$$

Together, the bounds Eq. (2.26) and Eq. (2.24) along with the result for symmetrically noisy inputs of Lemma 8, gives the desired result. □

For visualization purposes, and for consistency with Section 2.1.4, we introduce a transformed set of coordinates for Δ_3, defined by

$$x = \frac{\sqrt{3}}{4}(p_1 - p_0), \qquad \text{and} \qquad y = \frac{3}{4}\left(p_0 + p_1 - \frac{2}{3}\right). \tag{2.27}$$

We will denote the action of the transformation using barred variables $\bar{\mathcal{M}}_\varepsilon$, and the corresponding logical regions $\bar{\mathcal{R}}^{(i)}$. Note that, in these coordinates, the origin $x = y = 0$ corresponds to the maximally mixed probability distribution $(1/3, 1/3, 1/3)$, and the level curves of the coordinate p_2 are horizontal lines. A visualization of the denoising dynamics is shown for $q = k = 3$ in Fig. 2.3.

Computation threshold

First, we consider the class of operations that may be performed using only gates that preserve the symmetric noise of Definition 4 and show that such gates have the property that they can be computed up to the ultimate threshold of $(q-1)/q$ for all $k \geq 2$. While this class of operations is not itself universal, it may be augmented to a universal set using gates which can be computed reliably up to threshold $(q-1)/q$ asymptotically as $k \to \infty$.

Definition 10 (Symmetric noise preserving gates)**.** Let $\{X_i\}_{i=1}^k$ be independent, but not necessarily identically distributed, a_i-noisy encodings of $\{\hat{x}_i\}_{i=q}^k$.

A gate $g : [q]^k \to [q]$ is said to be *symmetric noise preserving (SNP)*, if the output

$$Y = g(X_1, \ldots, X_n), \tag{2.28}$$

is a-noisy where a is a function only of $\{a_i\}_{i=1}^k$.

Note that since the matrix for q-ary symmetric channel of Definition 5 is invertible for $\varepsilon < (q-1)/q$, this property holds independent of ε and therefore nothing is lost by considering noiseless gates.

Lemma 11 (SNP gates are not universal for $q > 2$)**.** The clone of symmetric noise preserving gates is not universal for alphabet of size $q > 2$.

Proof. Let g be a k-input SNP gate. Then for all indices $1 \leq i \leq k$, and all choices $\vec{c} = (c_j)_{j \neq i}$, the restricted function $g_{\vec{c},i} : [q] \to [q]$,

$$g_{\vec{c},i}(x) \equiv g(c_1, \ldots, c_{i-1}, x, c_{i+1}, c_k), \tag{2.29}$$

must either be a constant function or a bijection.

To see this, suppose there is a function g which does not satisfy Eq. (2.29) for some i and $\vec{c}$. Consider inputs X_j which are 0-noisy encodings of $\hat{x}_j = c_j$ for all $j \neq i$; and X_i an $(a_i > 0)$-noisy encoding of $\hat{x}_i$. Further let $\hat{g} = g_{\vec{c},i}(\hat{x}_i)$. Since $g_{\vec{c},i}$ is neither constant nor a bijection, for $q > 2$ there are $y, y' \in [q] \setminus \hat{g}$ such that $\Pr[G = y] = 0$ and $\Pr[G = y'] > 0$. Therefore $G = g(X_1, \ldots, X_k)$ is not a a-noisy.

Furthermore, the set of SNP functions is closed under composition and therefore forms a clone. Therefore any functions computed using only SNP gates must satisfy Eq. (2.29), and therefore SNP gates are not universal for q-ary computation. □

If we place additional constraints on the inputs, there may exist larger sets of gates that preserve symmetric noise. The denoising gate of Section 2.1.3, is a notable example which does not satisfy the conditions of Lemma 13 but nevertheless preserves symmetric noise for identically distributed inputs.

We now consider the following class of functions that generalize the Boolean XOR over larger alphabets:

Definition 12 (Pseudo-additive functions)**.** A k-input q-ary function g is called *pseudo-additive (PA)* if it can be written in the following form:

$$g(x_1, \ldots, x_k) = \sum_{i=1}^{k} \sigma_i(x_i) \pmod q, \tag{2.30}$$

where each $\sigma_i : [q] \to [q]$ is either constant function or a bijection.

Note that the set of pseudo-additive functions is closed under composition and therefore forms a clone. Additionally, we show that they preserve symmetric noise.

Lemma 13 (PA $\subset$ SNP)**.** A k-input q-ary pseudo-additive function is symmetric noise preserving.

Proof. First, we show that this condition is sufficient. Let $G_r \equiv \sum_{i=1}^{r} \sigma_i(X_i) \pmod q$ and $\hat{g}_r \equiv \sum_{i=1}^{r} \sigma_i(\hat{x}_i) \pmod q$. First we show that this condition is sufficient for unary gates g. If the function is constant, then $\Pr[g(X_1) = y] = \delta_{y,\hat{g}_1}$ and is trivially b-noisy with $b = 0$. If the function is a surjection, then

$$\Pr[g(X_1) = y] = \begin{cases} 1 - a_1, & \text{if } y = \hat{g}_1 \\ \frac{a_1}{q-1}, & \text{otherwise} \end{cases}, \tag{2.31}$$

and is b-noisy with $b = a_1$.

We proceed by induction on k. Suppose the Lemma holds for k-input functions giving a b-noisy output.

$$\begin{aligned} &\Pr[G_{k+1} = y] = \Pr[G_k + \sigma_{k+1}(X_{k+1}) = y \pmod q] = \\ &\quad \Pr[G_k + \sigma_{k+1}(X_{k+1}) = y \pmod q | G_k = \hat{g}_k, X_{k+1} = \hat{x}_{k+1}] \Pr[G_k = \hat{g}_k] \Pr[X_{k+1} = \hat{x}_{k+1}] + \\ &\quad \Pr[G_k + \sigma_{k+1}(X_{k+1}) = y \pmod q | G_k \neq \hat{g}_k, X_{k+1} = \hat{x}_{k+1}] \Pr[G_k \neq \hat{g}_k] \Pr[X_{k+1} = \hat{x}_{k+1}] + \\ &\quad \Pr[G_k + \sigma_{k+1}(X_{k+1}) = y \pmod q | G_k = \hat{g}_k, X_{k+1} \neq \hat{x}_{k+1}] \Pr[G_k = \hat{g}_k] \Pr[X_{k+1} \neq \hat{x}_{k+1}] + \\ &\quad \Pr[G_k + \sigma_{k+1}(X_{k+1}) = y \pmod q | G_k \neq \hat{g}_k, X_{k+1} \neq \hat{x}_{k+1}] \Pr[G_k \neq \hat{g}_k] \Pr[X_{k+1} \neq \hat{x}_{k+1}], \end{aligned} \tag{2.32}$$

$$= \delta_{y\hat{g}_{k+1}}(1 - a_{k+1})(1 - b) + \frac{1 - \delta_{y\hat{g}_{k+1}}}{q - 1}(a_{k+1} + b - 2ab) + \frac{q - 2 + \delta_{y\hat{g}_{k+1}}}{(q-1)^2} a_{k+1} b, \tag{2.33}$$

where the second equality is an application of the chain rule of probability; and the third equality makes use of assumptions that G_k and $\sigma_{k+1}(X_{k+1})$ are b-noisy and a_{k+1}-noisy (or 0-noisy if σ_{k+1} is constant) respectively, as well as the fact that modular addition randomizes equally among all elements of the alphabet.

We then have

$$\Pr[G_{k+1} = y] = \begin{cases} (1-a_{k+1})(1-b) + \frac{a_{k+1}b}{q-1}, & \text{if } y = \hat{g}_{k+1} \\ \frac{a_{k+1}+b-2a_{k+1}b}{q-1} + \frac{(q-2)a_{k+1}b}{(q-1)^2} & \text{otherwise} \end{cases}, \tag{2.34}$$

and is therefore symmetric noise preserving i.e. PA $\subseteq$ SNP.

Comparing with Definition 10, we see that for $q > 3$, the containment is strict. □

Theorem 1. *Pseudo-additive functions can be reliably computed up to the denoising threshold.*

Proof. First, we show that the pseudo-additive functions can be reliably for all error rates up to the maximally fixed-point $\varepsilon < (q-1)/q$.

Let $\sigma_\varepsilon : [q] \to [q]$ denote an ε-noisy permutation gate or constant gate and X be an a-noisy encoding of $\hat{x}$. If σ is a constant gate, then its output is always ε-noisy and therefore the output can be denoised up to the maximally mixed fixed-point. Therefore we focus on the case of a permutation σ, in which case $\sigma_\varepsilon(X)$ is a b-noisy encoding of $\sigma_0(\hat{x})$, where

$$b = \varepsilon + \left[1 - \left(\frac{q}{q-1}\right)\varepsilon\right]a. \tag{2.35}$$

Therefore for all $q \geq 2$,

$$a < \frac{q-1}{q} \implies b < \frac{q-1}{q}. \tag{2.36}$$

Therefore the permutation gate can reliably compute up to the denoising threshold.

Next, consider the ε-noisy modular addition gate $\text{ADD}_\varepsilon^{(q)}(x_1, x_2) = x_1 + x_2 \mod q$. If its inputs X_i are a_i-noisy encodings of $\hat{x}_i$ for $i \in \{1, 2\}$, then $\text{ADD}_\varepsilon^{(q)}(X_1, X_2)$ is b'-noisy with

$$b' = \varepsilon + \frac{(a_1(q-1) + (1-a_1)a_2 q - a_2)(q(1-\varepsilon)-1)}{(q-1)^2}. \tag{2.37}$$

Therefore for all $q \geq 2$,

$$a_1 < \frac{q-1}{q}, \text{ and } a_2 < \frac{q-1}{q} \implies b' < \frac{q-1}{q}. \tag{2.38}$$

The set of gates $\{\sigma, \textsc{add}^q\}$ generates the clone of PA functions, and therefore the clone of PA functions can be reliably computed to the maximally mixed fixed-point for all k. $\square$

Notably, the clone of PA functions can be reliably computed as long as its inputs are a-noisy for any $a \in [0, \frac{q-1}{q})$. This is in contrast to the case of universal computation over Boolean alphabets, in which reliable computation requires input noise to be fine-tuned to be near the denoiser fixed-points and making inputs less noisy adversarially can actually make reliable computing impossible [ES03; Ung10]. This fine-tuned input noise also happens to be required for our later result of Theorem 2.

While the set of SNP gates are not themselves universal (Lemma 11), they can be augmented into a universal set using a general 2-input operation over alphabet q [Lau06]. Naturally, all such operations can be reliably computed up to the ultimate threshold of $(q-1)/q$ as $k \to \infty$; for example, by using redundant inputs to perform native error correction (in the spirit of the XNAND for $q = 2$). Thus reliable universal computation can be achieved up to threshold nearing $(q-1)/q$ for large k.

Example $k = 3$ and q prime

In addition to the asymptotic result argued in Section 2.1.3, we show that the denoising threshold may be achieved for finite k. In fact, we demonstrate that for $k = 3$ and q prime, reliable universal computation can be achieved up to the denoising lower-bound of Lemma 8.

First, we find

$$c_0^{[q,3]} = 1, \qquad c_1^{[q,3]} = 1 - \frac{q-2}{3(q-1)}, \qquad c_2^{[q,3]} = 0, \qquad \text{and} \qquad c_3^{[q,3]} = 0. \tag{2.39}$$

This gives a denoising threshold lower-bound of

$$C^{[q,3]} = \frac{q+1}{q} \implies \beta^{[q,3]} \geq \frac{q-1}{q}\frac{C^{[q,3]}-1}{C^{[q,3]}} = \frac{q-1}{q(q+1)}. \tag{2.40}$$

Substituting the coefficients from Eq. (2.39) into Eq. (2.6), we find that at the transcritical bifurcation, the outputs are $(1/q)$-noisy. As an aside, a similar calculation shows that the saddle-node bifurcation occurs for $k = 3$ at

$$\varepsilon = \frac{q-1}{5q-4}, \tag{2.41}$$

with $(1/2)$-noisy outputs, which can be seen as the ultimate denoising threshold.

To obtain a lower-bound for the computation threshold, consider augmenting the PA gates with the modular multiplication operation operating on the first two inputs:

$$\text{MUL}^q_\varepsilon(X_1, \ldots, X_k) = X_1 \times X_2 \pmod q. \tag{2.42}$$

To see this, note that the most likely error is that the output of two non-zero elements is mistaken for '0.' For symmetrically noisy a-noisy inputs and prime q,

$$p_{\text{correct}} = (1-a)^2 + \frac{q-2}{(q-1)^2}a^2, \tag{2.43}$$

$$p_0 = \left(\frac{a}{q-1}\right)^2 + 2\left(\frac{a}{q-1}\right)\left(1 - \frac{a}{q-1}\right), \tag{2.44}$$

where, in calculating these probabilities, we have used the fact that all elements in $[q]$ have inverses for q prime. We find that $p_{\text{correct}} > p_0$ requires $a < 1 - 1/\sqrt{q}$.

Taken together with the error rate of the denoising operation from above, we find that for prime q and $k \geq 3$, reliable universal computation is achievable up to the denoising transition of Lemma 8. Notably, this is a consequence of the fact that the denoiser fixed-points do not approach the maximally mixed point at the point of the transcritical bifurcation, as in Fig. 2.2

for the $q = 3$ case. This is in stark contrast with the previously studied Boolean case ($q = 2$). Practically, this offers some advantages: instead of requiring a uniquely designed gate such as the XNAND to achieve the optimal computation threshold, for $k = 3$ and q prime, the standard modular multiplication operation can be reliably computed up to the transcritical bifurcation.

2.1.4 Universal Boolean computation using error signaling

In this Section, we design a set of gates for performing universal Boolean computation, showing that we can improve on the fault-tolerance threshold given in [EP98] if we add a third alphabet character to perform error signaling. We show that over a ternary alphabet under symmetric noise, we can achieve fault-tolerant computation whenever $\varepsilon < 1/6$. Again we proceed using the von Neumann construction by analyzing a denoising gate (Section 2.1.4) followed by a computation gate (Section 2.1.5) that is sufficient for universal Boolean computation over the logical alphabet. Throughout this section, we assume that $\varepsilon < 1/6$.

Denoising threshold

Consider a denoising gate DEN : $[3]^2 \to [3]$ with truth table given in Table 2.2, which performs denoising for two logical states over a ternary alphabet, encoded using the physical alphabet characters '0' and '1'. The third physical alphabet character, '2', can be understood as an error flag, while the first two physical alphabet characters constitute the logical alphabet. This DEN gate is designed to be used to combine multiple redundant computations to correct for any possible errors. Therefore, we can think of this gate as having realizations from two independent and identically distributed random variables as input, where the mode of the random variable is the value the inputs would take if no errors occurred in the computation. We would like the denoising gate to output this "correct" value as often as possible. If both inputs to the gate are the same member of the logical alphabet, then it is most likely that both inputs are correct and therefore the gate should output the value of the inputs, while if the inputs are differing members of the logical alphabet we output a '2' as it is equally likely that an error has occurred for each of the inputs. If one input is a '2' while the other is a member of the logical alphabet, we output the input that is a member of the logical alphabet, as it is more likely than not that the latter input is correct. Finally, if both inputs are '2's, we output a '2' as we do not have any information about which member of the logical alphabet is correct.

x	y	DEN(x,y)
0	0	0
0	1	2
0	2	0
1	1	1
1	2	1
2	2	2

Table 2.2: Truth table for a balanced denoising gate DEN for binary computation over a ternary alphabet. Note that DEN is symmetric, i.e. $\text{DEN}(x, y) = \text{DEN}(y, x)$.

We characterize the behavior of repeatedly applying DEN to a random variable with probability distribution p over the alphabet $\{0, 1, 2\}$. We parameterize the probability distribution over a ternary-valued random variable X with the tuple (p_0, p_1) as follows:

$$\begin{pmatrix} \Pr[X = 0] \\ \Pr[X = 1] \\ \Pr[X = 2] \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ 1 - p_0 - p_1 \end{pmatrix}. \tag{2.45}$$

Let DEN_ε denote an ε-noisy version of DEN. Then for two identical and independently distributed random variables X_1 and X_2 with distribution given by (p_0, p_1), the distribution of $\text{DEN}_\varepsilon(X_1, X_2)$ is given by the vector-valued function

$$\mathcal{D}_\varepsilon(p_0, p_1) \equiv \begin{pmatrix} \Pr[D(X_1, X_2) = 0] \\ \Pr[D(X_1, X_2) = 1] \end{pmatrix} = \begin{pmatrix} \left(1 - \frac{3\varepsilon}{2}\right)(2 - p_0 - 2p_1)p_0 + \frac{\varepsilon}{2} \\ \left(1 - \frac{3\varepsilon}{2}\right)(2 - 2p_0 - p_1)p_1 + \frac{\varepsilon}{2} \end{pmatrix}. \tag{2.46}$$

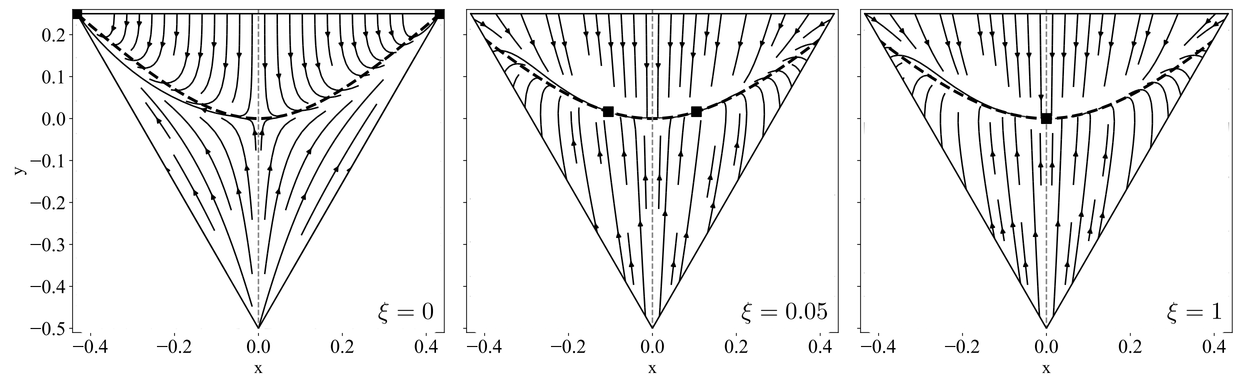


Figure 2.4: Streamlines of the vector field $\bar{\mathcal{D}}_\varepsilon(x,y) - (x,y)$ for various values of ξ showing the y nullcline (dashed black line), stable fixed-points $(x_\pm, y_\pm)$ (square markers), and line separating logical states (dashed grey line).

Lemma 14. Given a random variable with an initial probability distribution $(p_0, p_1, 1-p_0-p_1)$ over $(0,1,2)$, we observe that

$$\lim_{n\to\infty} \mathcal{D}^n_\varepsilon(p_0,p_1) = \begin{cases} \left(\frac{1}{3}, \frac{1}{3}\right) & \text{if } p_0 = p_1, \\ \left(\frac{(1-3\varepsilon)+\Delta}{1-\frac{3\varepsilon}{2}}, \frac{(1-3\varepsilon)-\Delta}{1-\frac{3\varepsilon}{2}}\right) & \text{if } p_0 > p_1, \\ \left(\frac{(1-3\varepsilon)-\Delta}{1-\frac{3\varepsilon}{2}}, \frac{(1-3\varepsilon)+\Delta}{1-\frac{3\varepsilon}{2}}\right) & \text{if } p_0 < p_1, \end{cases} \tag{2.47}$$

where $\Delta = \sqrt{(1-6\varepsilon)(1-2\varepsilon)}$.

Proof. It is straightforward to verify that the limit distributions described above correspond to fixed-points of $\mathcal{D}_\varepsilon$, and that they are the only fixed-points within the allowable region.

We first consider the case when $p_0 = p_1 = p$. By induction, any number of applications of DEN will maintain equality in the first and second arguments as

$$\mathcal{D}_\varepsilon(p,p) = \begin{pmatrix} p' \\ p' \end{pmatrix}, \tag{2.48}$$

where $p' = \left(1 - \frac{3\varepsilon}{2}\right)(2-3p)p + \frac{\varepsilon}{2}$.

Now note that for $0 < \varepsilon \leq 1/6$ and $p \in [0, 1/2)$,

$$\begin{aligned}\left|p' - \frac{1}{3}\right| &= \left|\left(1 - \frac{3\varepsilon}{2}\right)(2-3p)p + \frac{\varepsilon}{2} - \frac{1}{3}\right|,\\ &= \left|3\left(1 - \frac{3\varepsilon}{2}\right)\left(p - \frac{1}{3}\right)^2\right|,\\ &= \left|3\left(1 - \frac{3\varepsilon}{2}\right)\left(p - \frac{1}{3}\right)\right|\left|p - \frac{1}{3}\right|,\\ &< \left|p - \frac{1}{3}\right|.\end{aligned} \tag{2.49}$$

Therefore, repeated iteration results in convergence to 1/3. We conclude that $\lim_{n\to\infty} \mathcal{D}_\varepsilon^n(p,p) = (1/3, 1/3)$ for all $p \in [0, 1/2]$.

Next, we make use of the parameterization of Eq. (2.27). In these coordinates, the action of the denoising gate is

$$\bar{\mathcal{D}}_\varepsilon(x,y) \equiv (\xi + 3)\begin{pmatrix} \frac{x(1-y)}{3} \\ \frac{x^2-y^2}{2} \end{pmatrix}, \tag{2.50}$$

where $\xi = 1 - 6\varepsilon$. We consider the dynamics of probability distributions over the ternary alphabet under repeated application of Eq. (2.50) with sequences $\{(x_i, y_i)\}_{i=0}^{\infty}$, such that $(x_{i+1}, y_{i+1}) = \bar{\mathcal{D}}_\varepsilon(x_i, y_i)$. These discrete dynamics can be more easily visualized using a continuous approximation by plotting the streamlines of the vector field $\bar{\mathcal{D}}_\varepsilon(x,y) - (x,y)$ as in Fig. 2.4.

Our original problem is equivalent to showing that all points (x_0, y_0) in the regions

$$\mathcal{R}^{(0)} = \left\{(x,y)\middle| -\frac{\sqrt{3}}{4} \leq x < 0, -\frac{1}{2} \leq y \leq \frac{1}{4}, 0 \leq 1 + 2\sqrt{3}x + 2y\right\}, \tag{2.51}$$

$$\mathcal{R}^{(1)} = \left\{(x,y)\middle| 0 < x \leq \frac{\sqrt{3}}{4}, -\frac{1}{2} \leq y \leq \frac{1}{4}, 0 \leq 1 - 2\sqrt{3}x + 2y\right\}, \tag{2.52}$$

converge to their respective fixed-points,

$$\left(x^{(0)}, y^{(0)}\right) = \left(-\frac{\sqrt{\xi(\xi+2)}}{\xi+3}, \frac{\xi}{\xi+3}\right), \qquad \text{and} \qquad \left(x^{(1)}, y^{(1)}\right) = \left(\frac{\sqrt{\xi(\xi+2)}}{\xi+3}, \frac{\xi}{\xi+3}\right), \tag{2.53}$$

for $0 \leq \xi < 1$.

We focus on the case $p_0 < p_1 \implies (x_0, y_0) \in \mathcal{R}^{(1)}$, with $(x_0, y_0) \in \mathcal{R}^{(0)}$ following analogously. First, note that $x > 0$ is preserved under iteration of Eq. (2.50), and therefore $(x_0, y_0) \in \mathcal{R}^{(1)}$ implies that for all $i > 0$, $(x_i, y_i) \in \mathcal{R}^{(1)}$. One may verify using cylindrical algebraic decomposition that the following is a Lyapunov function for the dynamics:

$$V_\xi(x, y) = \left((\xi+3)^2x^2 - (\xi+2)(\xi+3)y\right)^2 + \left((\xi+3)^2x^2 - \xi(\xi+2)\right)^2. \tag{2.54}$$

That is, for all $i \geq 0$, $V_\xi(x_{i+1}, y_{i+1}) \leq V_\xi(x_i, y_i)$, and the function is strictly minimized at $(x^{(1)}, y^{(1)})$ for all points in $\mathcal{R}^{(1)}$. □

Lemma 14 shows that denoising is possible for $\varepsilon < 1/6$, and allow us to associate probability distributions over the physical alphabet in the regions $\mathcal{R}^{(0)}$, $\mathcal{R}^{(1)}$ with the logical 0 and 1 state respectively.

2.1.5 Computation threshold

Next, we show that universal computation is possible up to this threshold of $\varepsilon < 1/6$ by providing a universal computation gate that is compatible with the denoising gate DEN. We need to design a computation gate that is universal for binary computation over a ternary alphabet. Because the NAND gate is universal over a binary alphabet, we will design a balanced ternary gate whose behavior when applied to the fixed-points given in Lemma 14 is analogous to the behavior of a NAND gate.

To design this gate, we first note that the restriction of the gate to the logical alphabet $\{0, 1\}$ should be the NAND gate. It remains to assign outputs for cases when one or both inputs are 2. In order to keep the gate balanced, we therefore must assign the output 2 when both inputs are 2, and need to determine whether to assign an output of 0 or 2 when one input is a 0 or a 1 and the other is a 2.

Consider the ENAND gate, a generalization of the binary NAND gate to a ternary alphabet where one element is used for error signaling. This gate has a truth table given in Table 2.3.

x	y	ENAND(x,y)
0	0	1
0	1	1
0	2	2
1	1	0
1	2	0
2	2	2

Table 2.3: Truth table for a universal ENAND for binary computation over a ternary alphabet. Note that ENAND is symmetric, i.e. ENAND$(x, y) =$ ENAND(y, x).

We verify that this gate applied to the fixed-points given in Lemma 14 behaves analogously to a NAND gate acting on the corresponding elements of the logical alphabet.

Theorem 2. *For $(k = 2)$-input gates over an alphabet of size $q = 3$, reliable universal computation subject to ε-noisy symmetric noise is possible for $\varepsilon < 1/6$.*

Proof. We associate the logical values $\{0, 1\}$ to the distributions

$$P_0 = \begin{pmatrix} p_+ \\ p_- \\ 1 - p_+ - p_- \end{pmatrix}, \qquad \text{and} \qquad P_1 = \begin{pmatrix} p_- \\ p_+ \\ 1 - p_+ - p_- \end{pmatrix}, \tag{2.55}$$

where

$$p_\pm = \frac{(1 - 3\varepsilon) \pm \Delta}{1 - \frac{3\varepsilon}{2}}, \qquad \Delta = \sqrt{(1 - 6\varepsilon)(1 - 2\varepsilon)}. \tag{2.56}$$

Note that we are able to prepare states arbitrarily close to P_0 and P_1 through repeated denoising via Lemma 14.

Once again, we parameterize distributions over the ternary alphabet by using the probabilities corresponding to the two members of the logical alphabet, and let $\mathcal{G}_\varepsilon : (P_{\text{in},0}, P_{\text{in},1}) \mapsto P_{\text{out}}$ denote the function mapping input probability distributions to output probability distributions under the ε-noisy ENAND gate; further, we let $\mathcal{G}_\varepsilon^{(0)}$ and $\mathcal{G}_\varepsilon^{(1)}$ denote the respective probabilities of the output being in the 0 or 1 state. For the computation under the ENAND gate to be successful, given inputs from $\{P_0, P_1\}$, the output must land in the appropriate half of the simplex, that is, for inputs $u, v \in \{0,1\}$, $\mathcal{G}_\varepsilon(P_u, P_v) \in \mathcal{R}^{(\textsc{nand}(u,v))}$, or more specifically

$$\mathcal{G}_\varepsilon^{(0)}(P_0, P_0) < \mathcal{G}_\varepsilon^{(1)}(P_0, P_0), \tag{2.57a}$$

$$\mathcal{G}_\varepsilon^{(0)}(P_0, P_1) < \mathcal{G}_\varepsilon^{(1)}(P_0, P_1), \tag{2.57b}$$

$$\mathcal{G}_\varepsilon^{(0)}(P_1, P_0) < \mathcal{G}_\varepsilon^{(1)}(P_1, P_0), \tag{2.57c}$$

$$\mathcal{G}_\varepsilon^{(0)}(P_1, P_1) > \mathcal{G}_\varepsilon^{(1)}(P_1, P_1), \tag{2.57d}$$

thus implementing the binary NAND gate. We find that

$$\mathcal{G}_\varepsilon(P_0, P_0) = \left(\frac{1 - 3\varepsilon - \Delta}{2 - 3\varepsilon}, \frac{1 + \Delta + \varepsilon(-4 + 6\varepsilon - 3\Delta)}{2 - 3\varepsilon}\right), \tag{2.58a}$$

$$\mathcal{G}_\varepsilon(P_0, P_1) = \left(1 + 4\varepsilon - \frac{1}{1 - \frac{3\varepsilon}{2}}, \frac{1}{1 - \frac{3\varepsilon}{2}} - 6\varepsilon\right), \tag{2.58b}$$

$$\mathcal{G}_\varepsilon(P_1, P_0) = \left(1 + 4\varepsilon - \frac{1}{1 - \frac{3\varepsilon}{2}}, \frac{1}{1 - \frac{3\varepsilon}{2}} - 6\varepsilon\right), \tag{2.58c}$$

$$\mathcal{G}_\varepsilon(P_1, P_1) = \left(\frac{1 - 3\varepsilon + \Delta}{2 - 3\varepsilon}, \frac{1 - \Delta + \varepsilon(-4 + 6\varepsilon + 3\Delta)}{2 - 3\varepsilon}\right). \tag{2.58d}$$

We observe that the second and third inequalities of Eq. (2.57) hold since

$$\begin{aligned} \frac{1}{1 - \frac{3\varepsilon}{2}} - 6\varepsilon - \left(1 + 4\varepsilon - \frac{1}{1 - \frac{3\varepsilon}{2}}\right) &= \frac{1 - \frac{17\varepsilon}{2} + 15\varepsilon^2}{1 - \frac{3\varepsilon}{2}}, \\ &= \frac{(1 - 6\varepsilon)(1 - \frac{5\varepsilon}{2})}{1 - \frac{3\varepsilon}{2}} > 0. \end{aligned} \tag{2.59}$$

For the first inequality of Eq. (2.57),

$$\begin{aligned}\frac{1+\Delta+\varepsilon(-4+6\varepsilon-3\Delta)}{2-3\varepsilon}-\frac{1-3\varepsilon-\Delta}{2-3\varepsilon}&=\frac{\varepsilon(6\varepsilon-1)+\Delta-\frac{3\varepsilon\Delta}{2}}{1-\frac{3\varepsilon}{2}},\\&=\Delta-\frac{\varepsilon(1-6\varepsilon)}{1-\frac{3\varepsilon}{2}},\end{aligned}\tag{2.60}$$

and so the inequality is equivalent to

$$\Delta^2-\left(\frac{\varepsilon(1-6\varepsilon)}{1-\frac{3\varepsilon}{2}}\right)^2=\frac{(1-6\varepsilon)\left(1-5\varepsilon+16\varepsilon^2-3\varepsilon^3\right)}{\left(1-\frac{3\varepsilon}{2}\right)^2}>0,\tag{2.61}$$

which is true on the interval $\left[0,\frac{1}{6}\right)$. The final inequality follows by a similar argument.

Since $\mathcal{G}_\varepsilon$ is a continuous function of the input probability distributions, we have that for any $\varepsilon<1/6$ there exists some sufficiently small ball around the fixed-points of $\mathcal{D}_\varepsilon$ such that ENAND can be used to perform computation. Alternating ENAND gates and sufficiently many DEN gates therefore allows arbitrary computation to be performed for $\varepsilon<1/6$. $\square$

2.1.6 Concluding remarks

In this paper, we extended a number of positive results for reliable Boolean ($q=2$) computation to the setting of alphabets of size $q>2$.

In Section 2.1.3, we provided a generalization of the results of Evans and Schulman [ES03] for alphabets of size $q>2$, showing a positive result for the reliable computation. Notably, we find that reliable universal computation is possible up to $\varepsilon<(q-1)/q$ as $k\to\infty$ using q-ary majority gates. While this result applies only in the case of q-ary symmetric noise, we note that similar asymptotic arguments apply to generic models of i.i.d. gate noise over alphabets of size q. For example, suppose that the probability of maintaining the correct logical state is $1-\epsilon$ and the probability of the most likely erroneous state is $c\epsilon$ for some $c>0$. Then the asymptotic $k\to\infty$ threshold occurs at the point where the two equal, i.e. $\epsilon=1/(1+c)$, which is maximized in the case of symmetric noise $c=1/(q-1)$. General results for finite k depend on the existence of a properly adapted majority gate and computation gate. In the case of q-ary symmetric noise, we showed positive computation results for finite fan-in $k = 3$ and q prime up to the transcritical bifurcation (Lemma 8). We conjecture that reliable computation is possible up to the transcritical bifurcation for all k odd and q prime using q-ary majority gates. Interestingly, and in contrast with the Boolean case [ES03], distinguishable stable fixed-points exist past the transcritical bifurcation (see the $q = k = 3$ example in Figs. 2.2 and 2.3), leading to the possibility of reliable computation beyond the lower-bound of Lemma 8, though a redefinition of the logical regions in Δ_q is required.

In Section 2.1.4, we extended the work of Evans and Pippenger [EP98] for computation using 2-input gates, showing that universal computation is possible for $\varepsilon < 1/6$ using error signaling. We would also like to highlight a conceptual difference in the way fault-tolerance comes about for the model of Boolean computation over a ternary alphabet in Section 2.1.4. The DEN (Table 2.2) and ENAND (Table 2.3) gates in effect use the third element of the physical alphabet as an error flag, signaling uncertainty to subsequent gates, resulting in a nearly two-fold increase compared to the analogous result over Boolean alphabets [EP98]. The use of this third alphabet character is reminiscent of the usage of flag qubits [CR20] and space-time codes [Got22], where similar error signaling mechanisms are used to achieve quantum fault-tolerance. There are many extensions to the error signaling construction developed in this paper that would be interesting to explore. It is also interesting to study the optimal logical alphabet size for a given error model over the physical alphabet, and how to design error signaling gates to achieve optimal threshold error rates—it is likely that such gates using a logical Boolean alphabet would allow a higher fault-tolerance threshold than the q-ary majority gates under the same noise model.

Finally, while increasing the physical alphabet size under the q-ary symmetric noise model increases the threshold error rate—for example, the result of Section 2.1.4 shows that reliable Boolean computation is possible for $\varepsilon < \beta = 1/6$, nearly double the nominal threshold for standard Boolean computation [HW91; EP98]. However, we are not advocating generically for performing computation over larger alphabets as nominal error rates are not strictly comparable between different alphabet sizes. Since all errors are equally likely by definition under the symmetric noise model, signals are distinguishable for $\varepsilon < (q-1)/q$; and since distinguishability is sufficient for reliable computation as $k \to \infty$, computation over larger physical alphabets is afforded a natural advantage under the assumption of symmetric noise. In practice, error rates need to grounded in the utilization of some resource [Imp04; Tha+05; Tha+08; TC23]; for example, one could compare the energy required to implement an ε-noisy binary circuit with that required to implement an ε-noisy ternary circuit. The limit of large q and large k, for which our results yield the most favorable nominal thresholds, comes at the cost of significant resource requirements which need to be taken into account in practice. The comparison of error rates across different physical alphabet sizes presents an interesting opportunity for future work.

2.2 Biological computation and neural networks

2.2.1 Introduction

Early in the development of computer science, it was unknown if unreliable hardware would make the construction of reliable computers impossible. Whenever a component failed, the resulting error had to be corrected by additional components that were themselves likely to fail. Inspired by ideas from error correction, the notion of *fault-tolerant computation* resolved this issue in standard frameworks of classical and quantum computation [Neu16; Pip85; HW91; EP98; ES99; GQF05; Sho96]. In these settings, every computation is evaluated by a sequence of faulty components such as Boolean gates (e.g., AND, OR, NOT). If each component's probability of failure falls below a sharp threshold, a strict criterion defining fault-tolerant computation is provably satisfied: computations of any length can be performed with arbitrarily low error. It is also worth noting that the distinction between error correction and fault-tolerance is vital here: while error correction uses noiseless gates

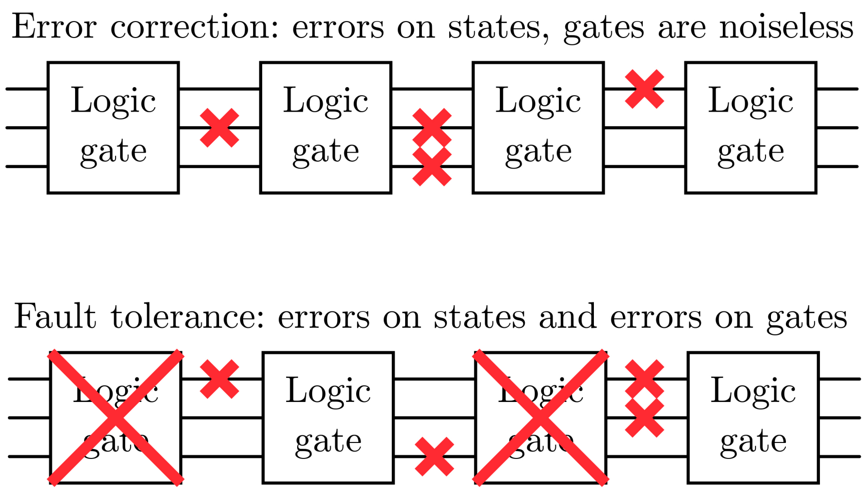

Error correction: errors on states, gates are noiseless
Logic gate
Logic gate
Logic gate
Logic gate
Fault tolerance: errors on states and errors on gates
Logic gate
Logic gate
Logic gate
Logic gate

Figure 2.5: Schematic comparison of error correction and fault-tolerance. While error correction uses noiseless gates to correct errors (red crosses), fault-tolerance must use faulty gates to generate reliable output. Note that errors on states in the fault-tolerant setup can be rewritten as errors on gates, i.e., faulty wires do not have to be directly considered.

to correct errors on a state, fault-tolerant computation only has access to faulty gates. We depict this distinction in Fig. 2.5.

In artificial intelligence, it is unresolved [TG17] if neural networks exposed to noise can satisfy an analogous criterion of fault-tolerance. That is, taking a noisy neuron as the fundamental component of computation, can any neural network be executed to arbitrary accuracy when the noise strength falls below a threshold? A similar question crops up in neuroscience, where observations of the mammalian brain have shown that neural *representations* are protected against noise by error correction codes [Haf+05; FBB08; SF11], yet it is unknown if such codes are powerful enough to protect *computations* to achieve arbitrarily small error.

We resolve both open questions in artificial intelligence and neuroscience by demonstrating fault-tolerant neural computation via carefully constructed error correction codes. This success hinges on generalizations of traditional fault-tolerance in Boolean formulas, as well as a modification of a biologically-observed error correction code known as *the grid code*. Beyond the analytic results proven here, we also provide a numerical estimate of the fault-tolerance threshold and show that naturally existing noisy biological neurons lie within the fault-tolerant regime.

Making the notion of fault-tolerance more precise, we begin by examining von Neumann's original result for fault-tolerant Boolean formulas, which perform universal computation using formulas of Boolean gates [Neu16]. In this setting, one considers access to *physical* gates, which are erroneous and fail (i.e. output the incorrect bit) with some fixed probability p. In a fault-tolerant construction, each gate in the original error-free formula is replaced by a *logical* gate composed of many physical gates. The logical gate is built with error correction such as the repetition code: data is repeated in bundles of three and majority voting determines the outcome. Despite the voting itself being performed by faulty physical gates, von Neumann showed via a recursive repetition code that a fault-tolerant Boolean formula can be constructed if the failure probability p falls below some threshold p_0. As Boolean gates suffer discrete errors, we will refer to the fault-tolerance of Boolean formulas as *digital fault-tolerance*, which is formally defined as follows:

Digital Fault-Tolerance. A Boolean formula containing N (error-free) gates can be simulated with probability of error at most ϵ using $O(N\text{polylog}(N/\epsilon))$ faulty gates. Each gate may fail with probability p for $p < p_0$, where p_0 is independent of N and independent of the noiseless formula depth.

In general, the value of p_0 depends on the model of computation under study [Pip85; HW91; EP98; ES99; GQF05]; for example, Ref. [EP98] demonstrated a noise threshold for reliable computation of $p_0 = (3 - \sqrt{7})/4 \approx 0.09$ for Boolean formulas constructed from 2-input NAND formulas, which are sufficient for universal computation.

The digital setting of traditional fault-tolerance strongly contrasts the *analog* computation paradigm of neuroscience and machine learning, where neurons operate using continuous rather than discrete values. Here, we consider two biologically-motivated sources of error. The first is (1) synaptic failure, where a connection between neurons is dropped [SW94; HSM93]; this may be modelled by having the neuron output 0 with some fixed probability p. This is in essence a discrete error (the connection is either present or it is not) and may be satisfactorily treated by an extension of von Neumann's construction. The second source of error is (2) analog noise afflicting the output of a neuron [SK93]; this may be modelled as additive Gaussian noise with standard deviation σ. This second type of error is more pernicious and will require specialized treatment via the grid code mentioned above.

To formalize the analog setting of computation, we adopt the framework of artificial neural networks [MP43], which are universal approximators of continuous functions [HSW89] and have experienced wide success in applications resembling cognitive tasks [LBH15]. The resilience of artificial neural networks to errors has been limited primarily to demonstrations of robustness to weight perturbations or other noise, and hardware fault-tolerance in neuromorphic computing [NSY92; EG17; Liu+19; SC90; NSY92; Ruc+89], without considering biologically-motivated noise nor addressing the formal notion of fault-tolerance analogous to digital fault-tolerance defined above.

We will ultimately prove the following result by using grid-code-based error-correcting mechanisms to achieve fault-tolerant neural computation:

> **Neural Network Fault-Tolerance.** A Boolean formula of N (error-free) gates can be simulated by a neural network with probability of error at most ϵ using only faulty neurons. Each synapse entering a neuron fails with probability p; the output of each neuron is subject to additive Gaussian noise with mean zero and standard deviation σ; a neuron admits at most a fixed number of synapses. There exist nonzero thresholds p_0 and σ_0 such that if $p < p_0$ and $\sigma < \sigma_0$, simulating the formula requires $O(N\text{polylog}(N/\epsilon))$ faulty neurons.

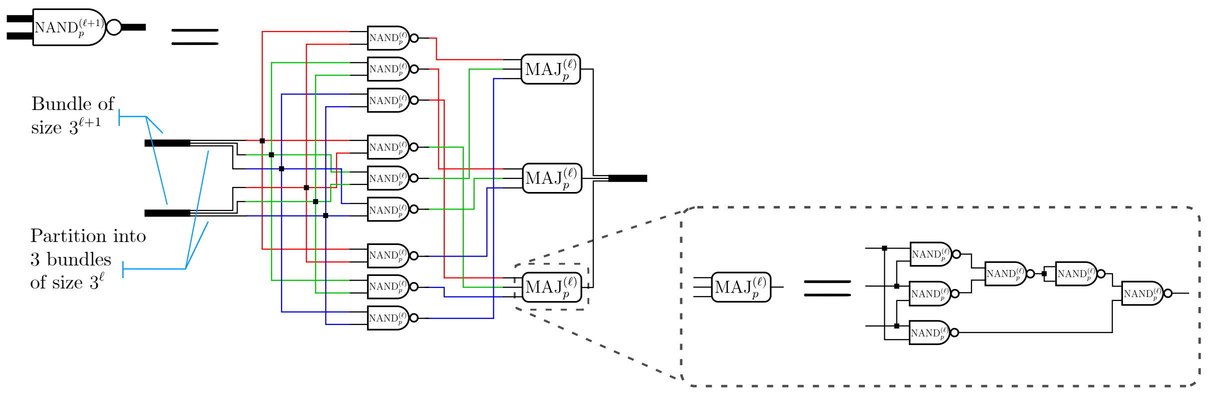
In the spirit of previous fault-tolerance results [Neu56; EP98; ES03], the core of our proof is the construction of a logical neuron from a configuration of noisy physical neurons.

An outline of this work is as follows. In Section 2.2.2, we first provide a brief review of digital fault-tolerance and then demonstrate how this construction may be adapted to design neural networks that are robust against synaptic failure. This is followed by the design of a neuron that is robust to additive Gaussian noise by encoding data in the grid code in Section 2.2.3; here we also demonstrate how error correction and computation may be achieved using a noisy neural network. We then showcase our fault-tolerant construction by designing a reliable circuit using our logical neuron subject to both modes of noise in Section 2.2.4, thereby proving our statement of neural network fault-tolerance. Finally, we provide some concluding remarks including a discussion of the biological plausibility of our assumptions in Section 2.2.5.

2.2.2 Fault-tolerance against digital errors

First, we provide a review of concatenated fault-tolerance results in digital circuits (Section 2.2.2). This is followed by a demonstration of an analogous technique for constructing neural networks that are robust against synaptic failure (Section 2.2.2).

Fault-tolerant Boolean circuits



$\text{NAND}_p^{(\ell+1)}$
$=$
Bundle of size $3^{\ell+1}$
Partition into 3 bundles of size 3^{ℓ}
$\text{NAND}_p^{(\ell)}$
$\text{MAJ}_p^{(\ell)}$

Figure 2.6: The recursive concatenation scheme, based on a ternary repetition code, used to construct a logical NAND gate at concatenation level $\ell + 1$ (denoted $\text{NAND}_p^{(\ell+1)}$) from logical NAND gates at concatenation level ℓ, with the base case $\text{NAND}_p^{(0)} = \text{NAND}_p$. The gates denoted $\text{MAJ}_p^{(\ell)}$ indicate a majority voting operation built from $\text{NAND}_p^{(\ell)}$ gates, whose explicit construction is illustrated in the inset.

The original construction of a fault-tolerant Boolean gate was initially proposed in Ref. [Neu16] and more rigorously discussed in Ref. [WC63]. We begin by presenting an adaptation of this construction via a recursive concatenation of repetition codes. To best explain this scheme, let us consider a Boolean gate B, with associated function $B(x)$ that accepts as input a string of bits x and outputs a single bit (for instance, $B = \text{NAND}$ and $B(x_0, x_1) = \text{NAND}(x_0, x_1) = \neg(x_0 \vee x_1)$). Let us also consider its faulty counterpart B_p that fails (i.e. outputs the incorrect bit) with probability p. We would like to construct a fault-tolerant version of B_p whose error can be decreased arbitrarily for $p < p_0$ for some threhsold p_0.

This is achieved by devising a recursive concatenation scheme wherein a *logical* B gate is constructed from *physical* B gates, these being the faulty B_p gates. In particular, a logical B gate at concatenation level-ℓ, which we denote by $B_p^{(\ell)}$, is recursively defined by a mapping of logical B gates at concatenation level $\ell - 1$ (i.e. $B_p^{(\ell-1)}$), with the base case $B_p^{(0)} = B_p$. In this mapping, $B_p^{(\ell)}$ is defined as a repetition code acting on multiple outputs of $B_p^{(\ell-1)}$, such that the error suffered by $B_p^{(\ell)}$ is less than that of $B_p^{(\ell-1)}$ for $p < p_0$. Thus, increasing ℓ decreases the error arbitrarily.

In his seminal work on fault-tolerance [Neu16], von Neumann employed a ternary repetition code, in which a logical bit is encoded as a bundle of physical bits. At concatenation level ℓ, each bundle consists of 3^ℓ physical bits, and its corresponding logical bit may be decoded as the majority of its physical bits. For instance, the bundle 110 encodes the logical bit 1 at concatenation level $\ell = 1$. In this manner, the inputs and outputs to $B_p^{(\ell)}(x)$ are bundles of size 3^ℓ, and the output is correct if its physical bits decode to the correct logical bit.

The recursive mapping from $B_p^{(\ell)}$ to $B_p^{(\ell+1)}$ is defined by this ternary repetition code: the inputs to $B_p^{(\ell+1)}$ are each linearly partitioned into three smaller bundles, which are then copied and sent through nine $B_p^{(\ell)}$ gates in parallel to generate nine independent outputs. To correct errors in these nine outputs, they are then split into three groups of threes, each of which is passed through a (faulty) majority voting gate, and the three resulting outputs are recombined to represent the final output of $B_p^{(\ell+1)}$. The majority voting gate is constructed from $B_p^{(\ell)}$ gates, and hence is also imperfect; its explicit construction depends on the Boolean gate of interest and influences the fault-tolerance threshold. In general, the fewer the gates in the majority gate, the larger the threshold.

For clarity, we depict this fault-tolerance construction applied to a NAND gate in Fig. 2.6. The specific arrangement of the wires fed into the majority voting gates is chosen is to prevent error propagation and produce a nonzero threshold. Not all arrangements will yield a nonzero threshold in the limit $\ell \to \infty$; von Neumann's original presentation even suggests randomly permuting these wires. Numerics indicate that this particular construction produces a threshold $p_0 \approx 2.36\%$.

As the NAND gate is universal for Boolean computation, this construction enables arbitrarily accurate computation of any Boolean function from faulty NAND gates if the failure probability p lies below the threshold p_0. Moreover, for $p < p_0$, the logical error suffered decreases doubly-exponentially with increasing ℓ, while the circuit size grows only exponentially with ℓ. Hence, achieving a desired error ϵ requires overhead $O(\text{polylog}(1/\epsilon))$ by the usual arguments for concatenation codes (see e.g. the fault-tolerance threshold theorem of Ref. [NC10]).

Moreover, for a circuit of N gates, an overall error of ϵ could be achieved by demanding individual gate errors ϵ/N as per the union bound. Inserting this desired error rate into the above polylogarithmic overhead, we find a total gate count $O(N\text{polylog}(N/\epsilon))$, in accordance with the digital fault-tolerance theorem discussed in Section 2.2.1.

Fault-tolerant neural networks for synaptic failure

The above fault tolerant construction may be adapted to devise a fault-tolerant neural network that is robust against synaptic failure, as this is in essence a discrete error. To illustrate this, let us consider a neural network constructed from rectified linear unit (ReLU) activation functions, where $\text{ReLU}(x) = \max(0, x)$ on real inputs x. In this case, synaptic failure may be modeled by replacing each ReLU with a faulty ReLU that fails with probability p, i.e.,

$$\mathrm{ReLU}_p(x) \equiv \begin{cases} \mathrm{ReLU}(x) & \text{with probability } 1-p \\ 0 & \text{with probability } p. \end{cases} \tag{2.62}$$

Like von Neumann's error model for Boolean gates, the output of this faulty ReLU is incorrect with some probability, and thus its errors may be corrected by employing a concatenated repetition code.

The aim is to construct a fault-tolerant ReLU activation function, which is equivalent to a fault-tolerant neuron. We will employ a concatenated ternary repetition code analogous to that presented above, replacing the logical Boolean gates with logical ReLU's. However, there is one important distinction in our construction: as inputs and outputs are now analog instead of binary, we will interpret the logical value carried by a bundle as the *median* of its values rather than the majority. Accordingly, the majority voting gate in the original repetition code is replaced by a median gate, which will appropriately correct errors that occur in a bundle. With this modification noted, we illustrate the complete recursive scheme in Fig. 2.7a; here, it is shown how to construct a logical ReLU at concatenation level $\ell + 1$ (denoted $\mathrm{ReLU}_p^{(\ell+1)}$) from logical ReLU's at concatenation level ℓ (denoted $\mathrm{ReLU}_p^{(\ell)}$), with the usual base case $\mathrm{ReLU}_p^{(0)} = \mathrm{ReLU}_p$.

What remains is to construct the median operation out of ReLU's. At concatenation level ℓ, we are interested in computing the median of three bundles, each of size 3^ℓ. Denoting this quantity as $m = \mathrm{Med}_p^{(\ell)}(a, b, c)$, where a, b, c represent each bundle, it may be computed with the following network of depth three:

$$x = \mathrm{ReLU}_p^{(\ell)}(a-b), \tag{2.63a}$$

$$y = \mathrm{ReLU}_p^{(\ell)}(-a+c+x), \tag{2.63b}$$

$$z = \mathrm{ReLU}_p^{(\ell)}(b-c+x), \tag{2.63c}$$

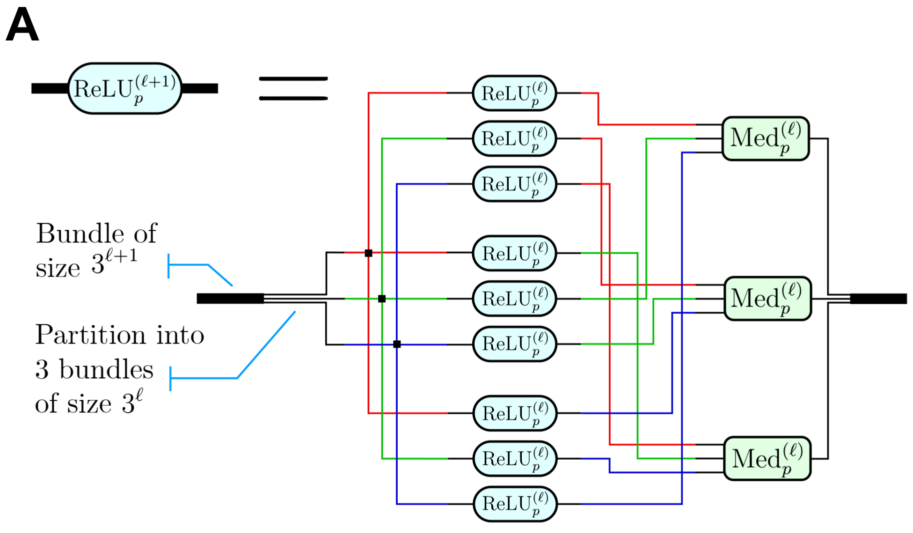
$$m = \mathrm{ReLU}_p^{(\ell)}(a+b-c+y-z). \tag{2.63d}$$

While the final ReLU is not strictly necessary for the computation of the median, it is included to prevent error propagation and achieve fault-tolerance. As a result, this median works only on positive inputs, but this is admissible as the output of ReLU_p^ℓ (which is input into the median) is necessarily non-negative. We also note that expressing this median construction as a neural network requires skip connections to perform its computation.

We visualize the performance of this fault-tolerance construction in Fig. 2.7 by plotting the *pseudothresholds*: where the error probability at concatenation level ℓ intersects that of $\ell = 0$. Plotting these for increasing levels of concatenation indicates a convergence to the threshold $p_0 \approx 3.72\%$. Therefore, this construction ultimately produces a fault-tolerant ReLU neuron, protected against synaptic failure for $p < p_0$. And by an argument analogous to the digital fault-tolerance of Boolean circuits, achieving a desired error ϵ requires overhead $O(\mathrm{polylog}(1/\epsilon))$. Using the argument presented at the end of Section 2.2.2, this translates to an overhead $O(N\mathrm{polylog}(N/\epsilon))$ for a circuit of N gates, thus achieving neural-network fault-tolerance (excluding Gaussian noise) as presented in Section 2.2.1.

2.2.3 Fault-tolerance against analog errors

While a simple adaptation of fault-tolerant constructions on noisy Boolean circuits yields a neural network that is robust to synaptic failure, the treatment of additive Gaussian noise proves more difficult. In particular, the repetition-based scheme of von Neumann fails for Gaussian noise with nonzero standard deviation σ: unlike the exponential suppression found



A
$\mathrm{ReLU}_p^{(\ell+1)}$
$\mathrm{ReLU}_p^{(\ell)}$
$\mathrm{Med}_p^{(\ell)}$
Bundle of size $3^{\ell+1}$
Partition into 3 bundles of size 3^{ℓ}

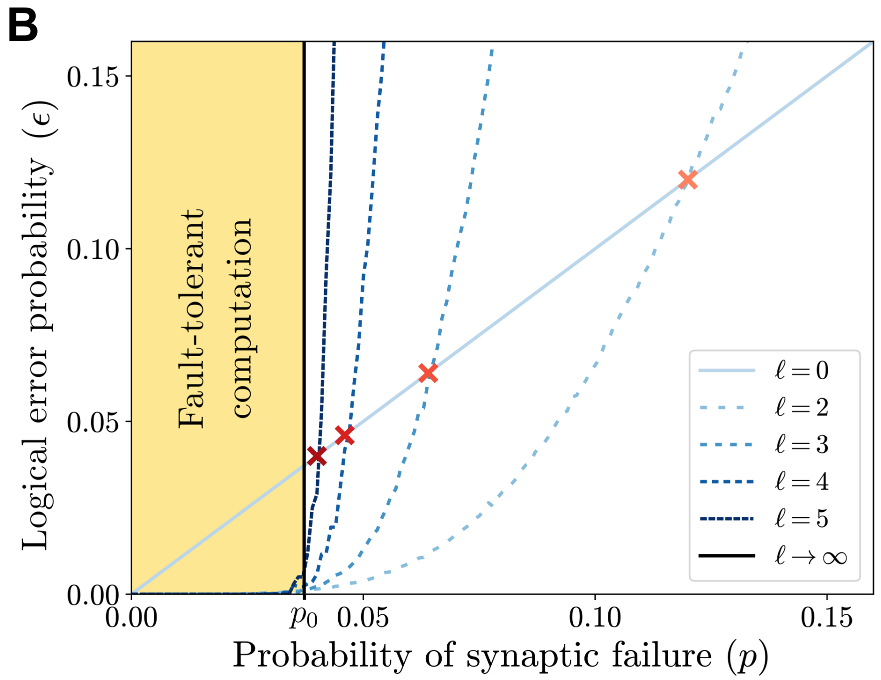


Figure 2.7: (A) The recursive concatenation scheme of digital fault-tolerance is extended to construct a logical ReLU at concatenation level $\ell+1$ from logical ReLUs at level ℓ. Note how the fault-tolerant ReLU is a generalization of the fault tolerant NAND gate in Fig. 2.6. The gates denoted $\mathrm{Med}_p^{(\ell)}$ indicate a median operation that is composed of $\mathrm{ReLU}_p^{(\ell)}$'s and used to correct errors; its explicit construction is presented in Eq. (2.63). This construction ultimately generates a logical neuron for a fault-tolerant neural network in the presence of synaptic failure. (B) The logical error probability of $\mathrm{ReLU}_p^{(\ell)}(x)$ on random inputs $x \in [-1, 1]$ as determined by numerical simulation. The pseudothresholds (red crosses) occur when the error probability intersects that of $\ell = 0$; they converge exponentially to the threshold $p_0 \approx 3.72\%$ (vertical black line) with increasing ℓ.

for digital errors, repeating N neurons in the presence of analog noise only reduces analog noise to $\sigma/\sqrt{N}$. Hence the requisite circuit size scales as $O(1/\epsilon^2)$, which does not achieve the $O(\text{polylog}(1/\epsilon))$ performance desired by the neural network fault-tolerance theorem.

Instead, we turn to an analog error correction code: the grid code. Unlike the repetition code, the grid code achieves exponentially small error at asymptotically finite information rates, saturating the Shannon bound [Gob65] and allowing effective error correction against Gaussian neural spiking noise [SF11] (see Appendix A.1 for a more detailed discussion).

We start with a brief overview of the grid code and its properties in Section 2.2.3. Next, we detail the construction of an error correcting procedure using noisy neurons in Section 2.2.3. Finally, we describe in Section 2.2.3 how the logical signal may be manipulated in a manner that allows for universal approximation and analyze its error threshold assuming a distribution of logical neural weights.

Overview of the grid code

We first provide a brief, self-contained exposition of the original grid code results of Refs. [Haf+05; FBB08; SF11]. These works study the entorhinal cortex in mammals and show that lattice neural firing patterns may correspond to a special encoding of the mammal's position (in 2D space), known as the *grid code*. In the grid code, a particular coordinate (say x or y in 2D space) takes values from a discrete set $\{x_k\}$ of S possible values that lie within a fixed interval $[0, X)$.

The encoding of each possible value x_k is modeled by a set of phases

$$\mathrm{Enc}\,[x_k] \equiv \left\{ \frac{e(x_k)}{\lambda_j} \bmod 1 \right\}_{j=1}^{M}, \tag{2.64}$$

which is defined over M relatively prime integers $\{\lambda_j\}_{j=1}^{M}$, referred to as *moduli* [Haf+05; FBB08], and a function $e(x)$ referred to as the *encoding function*. The choice of relatively prime moduli ensures, by consequence of the Chinese Remainder Theorem, that all $x \in$

$[0, \prod_{j=1}^{M} \lambda_j)$ are encoded into distinct codewords. Restricting our domain as above, with $X \ll \prod_{j=1}^{M} \lambda_j$, allows the remaining phase space to be used for error correction. Moreover, in the original grid code, the encoding function $e(x)$ is chosen to be the identity. Here, we will instead perform neural network computations by selecting $e(x)$ to implement an activation function; we will let $e(x)$ be an arbitrary function for now, and specify it later. In general, we denote the vector of M phases produced by the encoder as $\boldsymbol{\phi} \equiv \mathrm{Enc}\,[x_k] = \{\phi_j\}$. As a visual aid, an exemplary firing pattern of the grid code, as well as its moduli, is illustrated in Fig. 2.8a.

To maintain the favorable error-correcting properties of the grid code, the x_k's are chosen to satisfy $x_k \ll X$, and the minimum spacing between codewords $\Delta x \equiv \min_{k \neq j} |x_k - x_j|$ is chosen such that $\max_j \lambda_j \ll \Delta x$. More generally, when the encoding function $e(x)$ is not the identity function, the same condition must be upheld for Δx given by $\min_{i \neq j} |e(x_i) - e(x_j)|$ such that $e(x_i) \neq e(x_j)$.

In the limit $\max_j \lambda_j \ll X$ for $X \ll \prod_{j=1}^{M} \lambda_j$, the codeword $\boldsymbol{\phi}$ encoding a randomly sampled $x_{k^*} \in \{x_k\}$ is well-approximated as being drawn from a uniform distribution ($\phi_j \sim \mathcal{U}(0,1)$) [SF11]. We visualize this fact in Fig. 2.8b by plotting the phases of an example grid code. This property provides a sensitive encoding that changes significantly if the input is slightly perturbed. Since each codeword consists of a vector of phases $\{\phi_j\}$ with the period of each ϕ_j determined by λ_j, decoding corresponds to the constructive interference of summed phases to yield the correct decoded position, as depicted in Fig. 2.8c.

An ideal decoder $\mathrm{Dec}\,[\boldsymbol{\phi}]$ would perform maximum likelihood estimation (MLE) to recover the most probable value x_{k^*} given a codeword $\boldsymbol{\phi}$. For ease of presentation, we modify the original biologically inspired neural decoder that approximates MLE [SF11] to a simpler but functionally equivalent form; this form will be more easily implemented by a neural network later in this work. Given phases $\boldsymbol{\phi} = \{\phi_j\}$, we will recover the true position x_{k^*} by the MLE decoder

$$\mathrm{Dec}\,[\boldsymbol{\phi}] \equiv \arg\max_{x_k} \sum_{j=1}^{M} \cos\left[2\pi\left(\frac{x_k}{\lambda_j} - \phi_j\right)\right]. \tag{2.65}$$

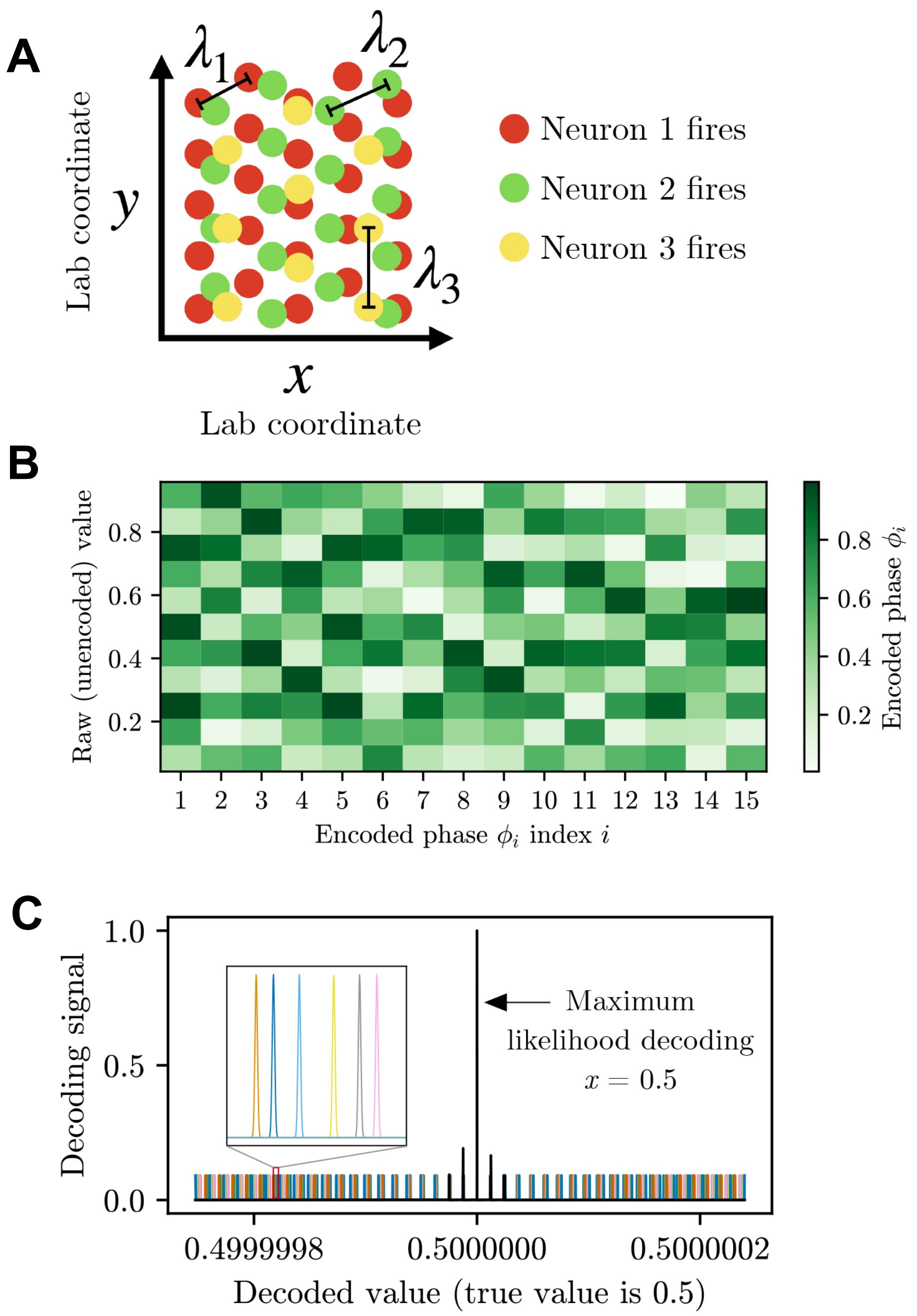


Figure 2.8: (A) Biological setting of the grid code. Neuron firings form a hexagonal lattice with different spacings λ_j, with lattice sites corresponding to physical locations of an animal in the lab. (B) Example encoding performed by the grid code over $M = 15$ moduli $\{\lambda_1, \ldots, \lambda_{15}\}$. Observe that these phases are well-approximated as being drawn uniformly at random, in accordance with the formalism of the grid code. (C) Example decoding of phases representing $x = 0.5$. The possible decodings allowed by a given phase (indicated by a unique color for each λ_j) are periodic. Each decoded phase is subject to Gaussian noise (inset). Since the phases constructively add at the true decoded value, maximum likelihood estimation selects the value with the highest signal.

To see that this procedure is indeed performing maximum likelihood estimation, observe that if x_{k^*} is known to belong to a discrete set of values $\{x_k\}$, then the estimated decoding $\hat{x}$ is given by maximizing the conditional probability

$$\hat{x} = \arg\max_{x_k} \Pr(\boldsymbol{\phi}|x_k). \tag{2.66}$$

Assuming that the encoding $\mathrm{Enc}\left[x_k\right]$ is distributed in the codespace according to a spherical Gaussian with variance s^2, the likelihood function is a wrapped normal distribution

$$\Pr(\boldsymbol{\phi}|x_k) \propto \prod_{j=1}^{M} \exp\left(-\frac{1}{2s^2}\|\mathrm{Enc}\left[x_k\right]_j - \phi_j\|^2\right), \tag{2.67}$$

where $\|\phi\| \equiv \min\{|\phi|, 1-|\phi|\}$ denotes the distance between phases. In the limit of $s \ll 1$, the likelihood function is well approximated by the more tractable circular normal function

$$\Pr(\boldsymbol{\phi}|x_k) \propto \prod_{j=1}^{M} \exp\left(\frac{1}{2\pi s^2}\cos\left[2\pi\left(\frac{x_k}{\lambda_j} - \phi_j\right)\right]\right). \tag{2.68}$$

Comparing Eq. (2.68) to Eq. (2.65), we see that the decoding scheme of Eq. (2.65) is indeed maximizing the likelihood.

Lastly, to more intuitively understand the grid code, note that because the M phases ϕ_j fall between 0 and 1, the coding space is the unit hypercube $[0,1]^M$; due to the unit modulo, the coding space satisfies periodic boundary conditions and thus corresponds to the M-torus. The coding line $[0, X)$ is thus a set of parallel line segments in the hypercube. In general, error correction codes may be described as a hypersphere packing problem: each codeword corresponds to an origin of a sphere in a high-dimensional space, and errors that fall within the radius of the sphere are correctable to the true codeword. Here, the grid code is a hypersphere packing problem in the $M-1$ dimensional hyperplane perpendicular to the coding line segments. Under this formalism, we arrive at a scaling of the minimum distance between line segments with the number of phases for fixed X of $d_{\min} = \Theta(\sqrt{M})$

[SF11], denoting an asymptotic bound on $d_{\min}$ from both above and below. Our choice of $\lambda \ll \Delta x$ ensures that each $\operatorname{Enc}[x_i]$ lies within a different line segment and is therefore also separated by at least $d_{\min}$, and consequently any perturbation in the phase space less than $d_{\min}/2$ is correctable using the maximum likelihood decoder.

The fault-tolerant logical neuron

Let us now use the grid code to present and analyze the construction of a fault-tolerant neuron. We assume an error model where Gaussian noise $\xi \sim \mathcal{N}(0, \sigma)$ is added to the output of each neuron, representing the noise associated with neural spikes in a biological setting. We note that we do not account synaptic failures at this stage, as the grid code is only tailored to analog noise; later in Section 2.2.4, we address both additive Gaussian noise and synaptic failure.

Focusing on a single neuron in a larger neural network, we take the number of neurons connected from the previous layer to be m_0. As in the presentation of the grid code, each neuron carries a value that is guaranteed to belong to a discrete set of S values, which we parameterize here as $\{x_k = k\Delta x\}$ for $k = 0, \ldots, S-1$, such that $(S-1)\Delta x < X$ for some X. The M relatively prime moduli must satisfy $\lambda_j \ll X \ll \prod_{j=1}^{M} \lambda_j$, and thus the codewords are uniformly distributed for random x_{k^*}.

We however introduce the following modification to the underlying grid code. While the typical grid code assumes a range of values $x_k \in [0, X)$, here we will take advantage of the periodicity of the grid code due to the periodicity of the phases (as these are evaluated modulo 1), and introduce a smaller range of values $[0, X')$ to which the encoding function $e(x)$ may output. That is, the encoding function $e(x)$ is chosen such that encoded values $e(x_k)$ exist in a condensed space $[0, X')$ for some $X' < X$, while fully decoded values x_k can still exist in the larger space $[0, X)$. A vanilla grid codes with identity encoding function $e(x_k) = x_k$ has $X' = X$; here, we will select $e(x_k)$ to be a non-identity function with $X' < X$, which will assist in building logical activation functions and thus performing neural network computation.

Moreover, to remain consistent with the usual requirement that $\lambda_j \ll X \ll \prod_{j=1}^{M} \lambda_j$, we will also demand $\lambda_j \ll X' \ll \prod_{j=1}^{M} \lambda_j$.

Turning now to the construction of our fault-tolerant neuron, we incorporate the traditional principles of fault tolerance: we perform computations in the codespace to protect against errors, and interleave each computation between encoding and decoding steps that correct errors and ensure that computation remains in the protected codespace. In the language of the grid code, this means performing computations on the phases $\boldsymbol{\phi}$, these computations corresponding to the application of weights and biases followed by an activation function.

The general construction of the logical neuron is presented in Fig. 2.9a. This depicts a logical neuron decomposed into physical neurons, with time advancing to the right. The number of inputs to the physical neurons is unrestricted, and hence this construction has an unbounded fan-in.

In the illustration, a previous layer of logical neurons passes to the logical neuron a set of encoded phase vectors, which we denote as $\boldsymbol{\theta}^{(i)} = [\theta_1^{(i)}, ..., \theta_M^{(i)}]$ for the M-dimensional phase vector of the ith logical input neuron. Assuming that inputs to the network are all encoded in the same grid code, each input phase vector $\boldsymbol{\theta}^{(i)}$ encodes a quantity that lies in $[0, X')$ in the decoded space.

The logical neuron itself consists of three stages: (1) the logical weights, (2) the decoder, and (3) the encoder. First, (1) the logical weights correspond to the weights of the error-free neuron that one seeks to apply; we denote these by $\{a_i\}_{i=1}^{m_0}$ for each of the m_0 (logical) neurons of the previous layer. As depicted in the figure, the logical weights are each repeated M times and then applied to the inputs $\boldsymbol{\theta}^{(i)}$, mapping directly from grid code phases to grid code phases.

Second, (2) the decoder performs error correction via maximum likelihood estimation (MLE) as described in Section 2.2.3. The key observation is that the structure of the grid code allows MLE to be approximated by a neural network. This is achieved using sine and cosine activation functions with appropriately chosen weights, the combination of which implements the MLE decoding scheme of Eq. (2.65) and also imposes the periodicity of the resulting phase encoding. The particular choice of weights, denoted $W_{ik}^{\sin}$ and $W_{ik}^{\cos}$, is explained and justified in the following section. Moreover, the decoder does not return to the original $[0, X')$ space; instead, it outputs a value in the larger space $[0, X)$, allowing the application of logical weights to decode to valid values. At the end of the decoding step, we are left with one-hot encoding representing the correct value $x_k \in [0, X)$ with high probability due to the robustness of the maximum likelihood estimate.

Lastly, (3) the encoder serves two roles: it re-encodes back into the codespace, and it also performs the computation via the application of a logical activation function (e.g., ReLU). Explicitly, the weights W_{ki}^{enc} are chosen such that this stage projects the one-hot representation of some x_k back to its appropriate codeword. The specific choice of weights also applies the logical activation function through a chosen encoding function $e(x_k)$, ultimately returning to the space $[0, X')$. The choice of weights W_{ki}^{enc} and encoding function $e(x)$ for various activation functions are presented in the following sections.

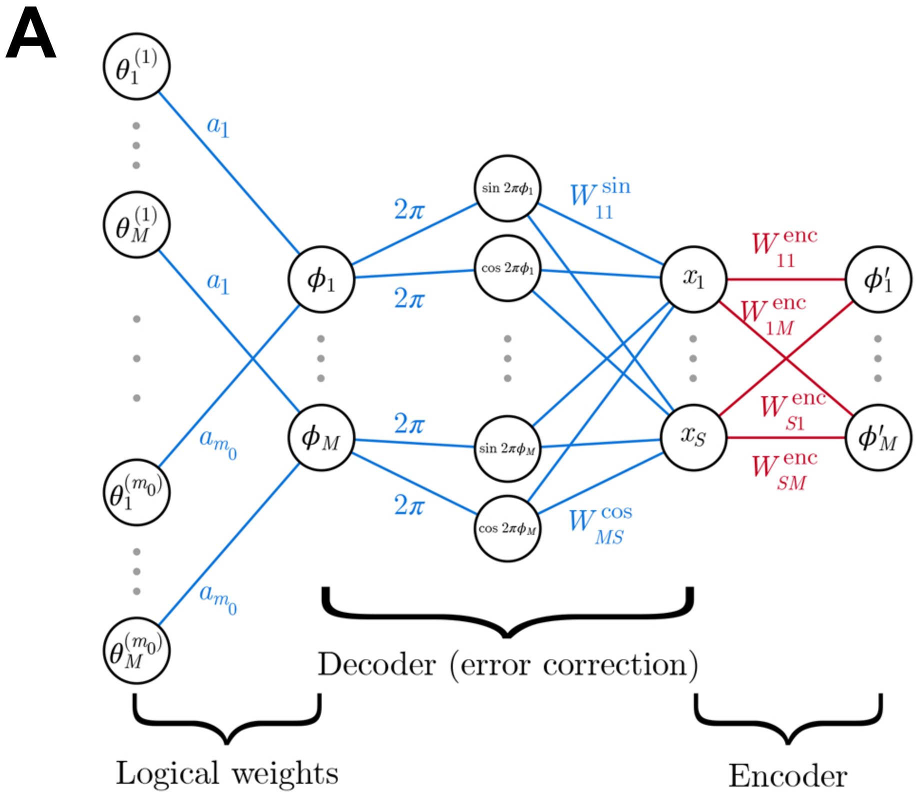
Neural network implementation of reliable computation

Let us now analyze the performance of the logical neuron in the fault-tolerant setting, where every physical neuron is subjected to noise. We will ultimately derive an analytical expression for the number of physical neurons needed to build a logical neuron with logical error at most ϵ.

To streamline our presentation, we begin by looking at the encoder stage. Accounting now for the additive Gaussian noise $\xi \sim \mathcal{N}(0, \sigma)$ suffered by the physical neurons, the encoder of Eq. (2.64) becomes

$$\widetilde{\text{Enc}}\,[x_k] = \left\{ \tilde{\phi}_j = \frac{e(x_k)}{\lambda_j} + \xi \bmod 1 \right\}, \tag{2.69}$$

for i.i.d. $\xi \sim \mathcal{N}(0, \sigma)$ sampled for each phase $\tilde{\phi}_j$. The decoder in the logical neuron uses only two layers (see Fig. 2.9a). The first layer multiplies each phase ϕ_j by a weight 2π; the



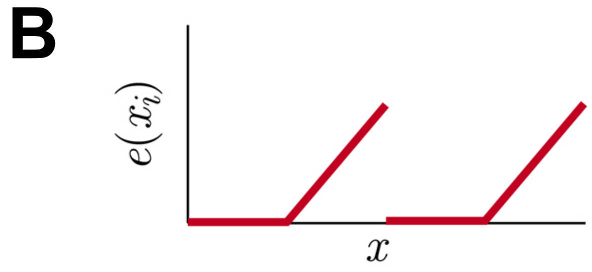


Figure 2.9: (A) Logical neuron decomposed into physical neurons to achieve fault-tolerance in the presence of analog noise. The neuron receives encoded neural outputs from the previous layer and performs a computation with time advancing to the right. The logical weights a_i are applied in the codespace, and the decoder recovers x_k by performing error correction. The encoder (red) performs a logical activation function (e.g., ReLU) using appropriate weights and encodes back to the codespace. (B) The logical ReLU encoding function $e(x_i)$ used in the encoder (Eq. (2.74)) for $X' = X/2$. This function is effectively a ReLU repeated over $X/X' = 2$ periods; the construction allows the fault-tolerant neural network to implement the standard ReLU activation function.

second layer uses sine and cosine activation functions to compute $\sin(2\pi\phi_j)$ and $\cos(2\pi\phi_j)$, and then multiplies them by weights $W_{jk}^{\sin}$ and $W_{jk}^{\cos}$, respectively. We select these weights to be $W_{jk}^{\sin} = \sin\left(2\pi\frac{x_k}{\lambda_j}\right)$ and $W_{jk}^{\cos} = \cos\left(2\pi\frac{x_k}{\lambda_j}\right)$.

Upon applying a decoding, any error is 'reset' if the decoding $\widetilde{\mathrm{Dec}}\left[\tilde{\boldsymbol{\phi}}\right]$ is successful, such that the logical neuron will not propagate any additional error into future computations. Evaluating all noise contributions, we have

$$\mathrm{Dec}\left[\tilde{\boldsymbol{\phi}}\right] = \arg\max_k f(k), \tag{2.70}$$

$$f(k) \equiv \xi + \sum_{j=1}^{M}\left[f_1(j,k) + f_2(j,k)\right], \tag{2.71}$$

where

$$f_1(j,k) = \sin\left(2\pi\frac{x_k}{\lambda_j}\right)\left[\sin\left(2\pi\tilde{\phi}_j\right) + \xi\right], \tag{2.72}$$

$$f_2(j,k) = \cos\left(2\pi\frac{x_k}{\lambda_j}\right)\left[\cos\left(2\pi\tilde{\phi}_j\right) + \xi\right], \tag{2.73}$$

and as usual each ξ is sampled i.i.d.

Suppose that the correct neuron value corresponds to $k = k^*$, i.e. the value x_{k^*} is encoded in the phases. For the decoder to identify the correct neuron via a threshold cutoff, we require $f(k = k^*) > f(k \neq k^*)$ for all k. If the mean of the correct neuron is greater than the mean of each incorrect neuron, a threshold will exist to distinguish the correct decoding from incorrect decodings in expectation. We use this insight to gain an analytical scaling for the number of physical neurons needed to construct a logical neuron with logical error ϵ.

The key observation is that in the noiseless limit, the phases ϕ_j are given by $\phi_j = x_{k^*}/\lambda_j \bmod 1$. At $k = k^*$, the elements in the sum of Eq. (2.70) constructively add as $f_1(j,k^*) + f_2(j,k^*) \approx 1$ for each j and thus $f(k = k^*)$ has a non-zero mean. On the other hand, for all $k \neq k^*$, the neural network weights are sine or cosine of a uniformly distributed random variable and the terms in the sum destructively interfere leaving $f(k \neq k^*) \approx 0$ on average.

To make the scaling argument precise for computation, we need to characterize the noise in Eq. (2.70), which requires assumptions to be made about the statistical properties of the noise and logical weights. With this in mind, we make the following assumptions. In order to maintain properties of modular arithmetic, we restrict the logical weights to integer values $a_i \in \mathbb{Z}$ such that $\sum_i |a_i| \leq X/X'$. We also assume that the logical weights a_i are approximately normally distributed from a Gaussian distribution with standard deviation α. Additionally, we take both the number of moduli M and the number of neurons m_0 connected from the previous layer to be much larger than one, allowing application of the central limit theorem.

For exemplary purposes, we will select ReLU as the logical activation function of the logical neuron; other activation functions may be implemented analogously. A ReLU activation function may be implemented by the encoding function

$$e(x_i) = \begin{cases} 0 & (x_i \bmod X') < X'/2 \\ (x_i - X'/2) \bmod X' & (x_i \bmod X') \geq X'/2, \end{cases} \tag{2.74}$$

which behaves like a periodic ReLU. We depict this encoding function in Fig. 2.9b. This function corresponds to choosing weights on the physical neurons $W_{ij}^{\text{enc}} = \frac{e(x_i)}{\lambda_j} \bmod 1$.

The above analysis can now be made explicit to demonstrate the fault-tolerant properties of the logical neuron. Looking at the logical weights stage, the m_0 logical neurons from the previous layer connected to the logical neuron are represented by codewords $\text{Enc}\left[x^{(1)}\right], \ldots, \text{Enc}\left[x^{(m_0)}\right]$, i.e. $\text{Enc}\left[x^{(i)}\right] = \boldsymbol{\theta}^{(i)}$. Therefore, the application of the logical weights must map from the $m_0 \times M$ neurons in $\{\text{Enc}\left[x_i\right]\}$ to a single set of phases $\{\phi_j\}$ such that our activation function (ReLU as we've chosen) is applied *after* decoding and re-encoding, as per the order of operations in the logical neuron. By assigning weights $W_{ij} = a_i$ from $\theta_j^{(i)}$ to ϕ_j (as illustrated in the 'Logical weights' layer in Fig. 2.9a) with a linear activation function and bias $-\frac{X'}{\lambda_j}\sum_{i:a_i<0} a_i$, we obtain the following phases in the absence of noise:

$$\phi_j = \left(\sum_{i=1}^{m_0} a_i \theta_j^{(i)}\right) - \left(\frac{X'}{\lambda_j}\sum_{i:a_i<0}^{m_0} a_i\right). \tag{2.75}$$

Factoring in noise now, let us denote a noisy encoding by phases $\tilde{\theta}_j^{(i)}$. Each of the S neurons over the discretized decoded space have noise ξ, and each is multiplied by approximately uniformly distributed weights due to the phases over the moduli. Applying the central limit theorem to $\sum_{i=1}^{S} u\xi$ for $u \sim \mathcal{U}(0,1)$, we find this equivalent to noise with mean zero noise and variance $\sigma^2 \cdot S/3$. Adding this noise of the codespace neuron to the noise acquired from the physical neuron, we find $\tilde{\theta}_j^{(i)} = \theta_j^{(i)} + \xi + \zeta$ for $\zeta \sim \mathcal{N}(0, \sigma\sqrt{S/3})$. Inserting noise in Eq. (2.75), we have

$$\begin{aligned} \tilde{\phi}_j &= \left[\sum_{i=1}^{m_0} a_i \tilde{\theta}_j^{(i)}\right] - \left(\frac{X'}{\lambda_j}\sum_{i:a_i<0} a_i\right) + \xi, \\ &= \phi_j + \xi + \sum_{i=1}^{m_0} a_i(\xi + \zeta). \end{aligned} \tag{2.76}$$

Applying the central limit theorem to the last term $\sum_{i=1}^{m_0} a_i(\xi+\zeta)$, we find that its mean vanishes while its variance is $\frac{1}{3}Sm_0\alpha^2\sigma^2$ in the large-S limit (having already applied the central limit theorem to S).

We can now formalize the above argument that correct decoding requires $f(k=k^*) > f(k \neq k^*)$. To simplify notation, we introduce the variable $\beta \equiv 4\pi^2(1 + Sm_0\alpha^2/3)$. By applying the error correction analysis (Eq. (2.70)) to the phases after a step of computation (Eq. (2.75)) again in large M regime, we find that the true decoding after application of the logical neuron is distributed as

$$f(k=k^*) \sim \mathcal{N}\left(Me^{-\beta\sigma^2/2}, \sqrt{M\left(\frac{1}{2} + \frac{1}{2}e^{-2\beta\sigma^2} - e^{-\beta\sigma^2} + \sigma^2\right) + \sigma^2}\right), \tag{2.77}$$

while the incorrect decoding is centered at zero:

$$f(k \neq k^*) \sim \mathcal{N}\left(0, \sqrt{M\left(\frac{1}{2}+\sigma^2\right)+\sigma^2}\right), \tag{2.78}$$

where both distributions are seen to have standard deviations $O(\sqrt{M})$. Upper-bounding the maximum element drawn from the distribution of $f(k \neq k^*)$ out of S draws using Jensen's inequality and a union bound, we find that

$$\begin{aligned} f_{\max}(k \neq k^*) &\equiv \mathbb{E}[\text{max draw of } f(k \neq k^*)], \\ &\leq \sqrt{[M(1+2\sigma^2)+2\sigma^2]\log S}. \end{aligned} \tag{2.79}$$

Finally, to determine if $\arg\max_k$ returns a value other than k^*, we compute the probability that this exceeds $f(k = k^*)$:

$$\begin{aligned} \Pr[\text{logical neuron fails}] &= \Pr[f(k=k^*) < f_{\max}(k \neq k^*)], \\ &\leq \frac{1}{2}\text{erfc}\left[\frac{e^{-\beta\sigma^2/2}M - \sqrt{[M(1+2\sigma^2)+2\sigma^2]\log S}}{\sqrt{M\left(1+e^{-2\beta\sigma^2}-2e^{-\beta\sigma^2}+2\sigma^2\right)+2\sigma^2}}\right]. \end{aligned} \tag{2.80}$$

This error probability is the logical error, which we seek to upper bound by ϵ. Expanding in small ϵ and taking $M, \beta\sigma^2 \gg 1$, we find that the number of moduli required to bound the logical error by ϵ scales is

$$\begin{aligned} M(\epsilon) &\approx \log(1/\epsilon)\left[e^{\beta\sigma^2}(1+2\sigma^2)+e^{-\beta\sigma^2}-2\right], \\ &\approx e^{\beta\sigma^2}(1+2\sigma^2)\log(1/\epsilon) = O(e^{\beta\sigma^2}\log(1/\epsilon)), \end{aligned} \tag{2.81}$$

where β is the aforementioned constant independent of the noise or error correction overhead. The $O(e^{\beta\sigma^2})$ dependence on σ originates from the constructive interference of the grid code: noisy phases for the true decoding contribute to a neuron with mean activation $Me^{-\beta\sigma^2/2}$, while the incorrect decoding yields a mean activation of zero. Although the noise produces neural activations of variance $O(\sigma\sqrt{M})$, there always exists sufficiently large M to identify the correct decoding.

Note that the number of physical neurons in the logical neuron scales linearly in the number of moduli, as per its structure in Fig. 2.9a). Hence, a fault-tolerant neural network can be constructed under the presence of arbitrarily large additive Gaussian noise using $O(e^{\beta\sigma^2}\log(1/\epsilon))$ physical neurons for a constant β. For a network of N neurons, this translates to $O(e^{\beta\sigma^2}N\text{polylog}(N/\epsilon))$ physical neurons as per the argument of Section 2.2.2. This result is in agreement with the neural network fault-tolerance theorem of Section 2.2.1 (excluding synaptic failure, which we address in Section 2.2.4), and also mirrors known results in digital fault-tolerance [ES03].

2.2.4 Reliable circuits from the fault-tolerant neuron

The fault-tolerant neural network presented above is a universal approximator of continuous functions due to the use of a ReLU activation function. In this Section, we demonstrate the flexibility of the fault-tolerant neural network construction by building Boolean circuits from fault-tolerant neural networks and providing evidence of their reliability. In Section 2.2.4, we numerically verify the predictions of Section 2.2.3 by simulating the code size requirement to implement a two-bit Boolean multiplication circuit constructed from neurons subject to additive Gaussian noise. In Section 2.2.4, we combine the constructions of Section 2.2.2 and Section 2.2.3 to provide analytic and numerical evidence of the robustness of our fault-tolerant neural network against both additive Gaussian noise and synaptic failure. Finally, in Section 2.2.4, we move towards more biological code parameters by studying the more biologically realistic scenario where moduli are encoded redundantly.

Reliability in the presence of Gaussian noise

Building on the fault-tolerant neural network of Section 2.2.3, a natural extension of this framework to Boolean gates emerges if additional encoding functions are introduced. In particular, as computations are done in the encoding step of the fault-tolerant neuron in

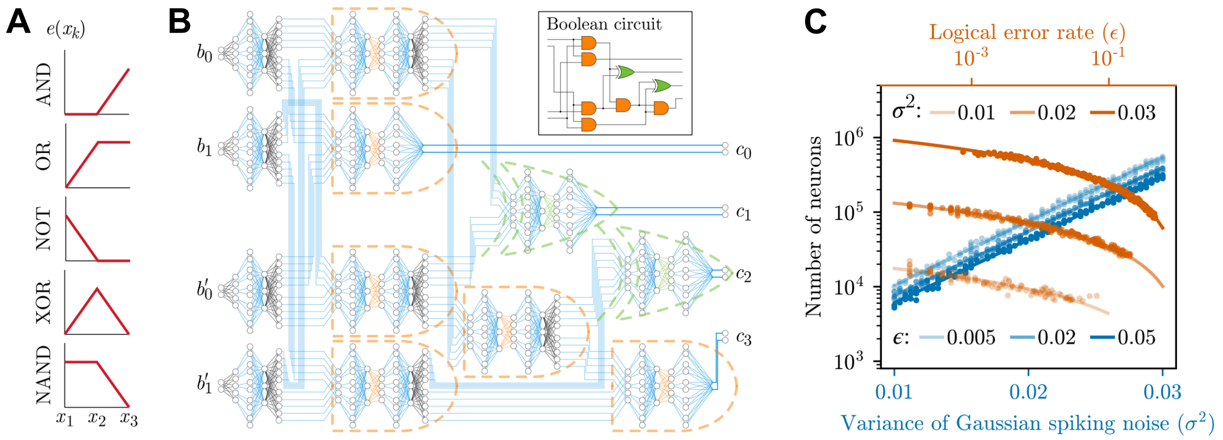


Figure 2.10: (A) Encoding functions $e(x_k)$ that induce appropriate logical activation functions to implement common Boolean gates. All logical weights $\{a_i\}$ are set to unity when implementing a Boolean gate. (B) Fault-tolerant neural network implementation of the two-bit multiplication circuit. In the inset, we illustrate the two-bit multiplication circuit decomposed into six AND gates (orange) and two XOR gates (green). In the neural network implementation thereof, two 2-bit binary numbers b_0b_1 and $b'_0b'_1$ (using the 0 index to denote the least significant bit) are one-hot encoded (as per Eqs. (2.82) and (2.83)) as input to the noisy neural network. The output $c_0c_1c_2c_3$ yields the product of the two numbers, and can achieve arbitrarily small error by increasing the number of moduli ($M = 5$ moduli illustrated). The grid code corrects Gaussian noise via the decoder (blue lines); neural encoders evaluate AND gates (orange outline) and XOR gates (green outline) to perform computation using appropriate encoder activation functions (shown in A); additional decoders and encoders are used to generate error-corrected copies of neural states (black). (C) Numerical simulation of the number of neurons required to perform two-bit multiplication with logical error probability ϵ in the presence of Gaussian noise of variance σ^2. The fit confirms the analytic scaling $O(e^{a\sigma^2}N\log(N/\epsilon))$ of Eq. (2.81).

Section 2.2.3, special encoding functions can be used to implement AND, OR, NOT, XOR, and NAND operations, among other Boolean gates. We illustrate these encoding functions in Fig. 2.10a, whose specific construction we discuss next. Afterwards, we will use these Boolean gate constructions to enable a fault-tolerant neural implementation of a multiplier circuit. Because these constructions use the fault-tolerant neuron of Section 2.2.3, they are robust against Gaussian noise only; we account both Gaussian noise and synaptic failure in Section 2.2.4.

To formalize this construction, define two logical input bits $A, B \in \{0, a\}$, interpreting 0 as False and a as True. Letting $\Delta x = a$, the decoder $\mathrm{Dec}[\boldsymbol{\phi}]$ will only decode to the set of variables $\{x_1 = 0, x_2 = a, x_3 = 2a\}$. For notational convenience, we define codeword vectors

$$\boldsymbol{\phi}_a \equiv \left\{\phi_j = \frac{a}{\lambda_j} \bmod 1\right\}, \tag{2.82}$$

$$\boldsymbol{\phi}_0 \equiv \left\{\phi_j = \frac{0}{\lambda_j} \bmod 1\right\}. \tag{2.83}$$

Beginning with a NOT gate, define the NOT encoder $\text{Enc}^{\neg}[x_1] = \boldsymbol{\phi}_a$ and $\text{Enc}^{\neg}[x_2] = \text{Enc}^{\neg}[x_3] = \boldsymbol{\phi}_0$. As before, this corresponds to a neural network with weights given by the codeword vectors. To compute $\neg A$, we simply compute $\text{Enc}^{\neg}[\text{Dec}[A]]$, which applies error correction and re-encode into the codespace with a NOTcomputation.

To implement AND and OR gates, we require an additional layer of unity weights, producing the value $\phi_i = \theta_i^A + \theta_i^B$ for input phases corresponding to bits A and B. Applying the decoder will give either $0, a$ or $2a$ based on the cases $(A, B) \in \{(0,0)\}$, $\{(a,0),(0,a)\}$ or $\{(a,a)\}$ respectively. The AND encoder is given by $\text{Enc}^{\wedge}[x_1] = \text{Enc}^{\wedge}[x_2] = \boldsymbol{\phi}_0$ and $\text{Enc}^{\wedge}[x_3] = \boldsymbol{\phi}_a$, and the OR encoder is given by $\text{Enc}^{\vee}[x_1] = \boldsymbol{\phi}_0$ and $\text{Enc}^{\vee}[x_2] = \text{Enc}^{\vee}[x_3] = \boldsymbol{\phi}_a$.

Likewise the XOR encoder is given by $\text{Enc}^{\oplus}[x_1] = \text{Enc}^{\oplus}[x_3] = \boldsymbol{\phi}_0$ and $\text{Enc}^{\oplus}[x_2] = \boldsymbol{\phi}_a$, and the NAND encoder by $\text{Enc}^{\bar{\wedge}}[x_1] = \text{Enc}^{\bar{\wedge}}[x_2] = \boldsymbol{\phi}_a$ and $\text{Enc}^{\bar{\wedge}}[x_3] = \boldsymbol{\phi}_0$. These Boolean gates furnish a universal gate set, from which arbitrary Boolean circuits, and thus arbitrary computations, may be achieved in a fault-tolerant manner. As per the results of Section 2.2.3, a fault-tolerant neural network assembled of these *neural-Boolean* gates satisfies the neural network fault-tolerance theorem (excluding synaptic failure) with $O\left(e^{O(\sigma^2)}\log(1/\epsilon)\right)$ physical neurons.

As an application of these neural-Boolean gates, we use them to implement a fault-tolerant two-bit multiplier. In this construction, the individual Boolean gates of the two-bit multiplier circuit are replaced with their corresponding neural-Boolean gates. We depict this circuit in Fig. 2.10b. Here, the neural network takes in two 2-bit binary numbers b_0b_1 and $b_0'b_1'$ and outputs their product, suffering an error that can be decreased arbitrarily error by increasing the number of moduli. For this neural two-bit multiplier, we numerically estimate the circuit size required to achieve a logical error rate ϵ with respect to the Gaussian noise strength σ^2. The results are shown in Fig. 2.10c and are in good agreement with the analytic prediction of Eq. (2.81).

Reliability in the presence of Gaussian noise and synaptic failure

Next, we study fault-tolerance with respect to both modes of noise: synaptic failure and additive Gaussian noise. Here, we consider a fault-tolerant neural NAND gate, which simplifies analysis as it is alone sufficient for universal Boolean computation. By comparing the NAND encoding function of Fig. 2.10a and the ReLU encoding function of Fig. 2.9b, we see that the NAND encoding function is the opposite of the ReLU encoding function. Hence, we can transfer over the Gaussian noise analysis of Section 2.2.3 to the setting of the neural NAND gate, with the modification that we choose 0 to correspond to the True state and a to the False state. This makes the ReLU encoding function equivalent to the direct implementation of a NAND gate.

Repeating the noisy logical neuron analysis of Eq. (2.77), but now with logical weights $a_i = 1$ and three decoder neurons, i.e. $S = 3$, as per the neural NAND gate construction, we find

$$f_{\text{NAND}}(k^*) \sim \mathcal{N}\left(M \cdot \frac{e^{-6\pi^2\sigma^2}\,\text{erf}^6(\sqrt{2}\pi\sigma)}{2^9\pi^3\sigma^6}, \sqrt{M\left(\frac{1}{2} + \sigma^2 - \zeta\right) + \sigma^2}\right), \tag{2.84}$$

for

$$\zeta = \frac{e^{-12\pi^2\sigma^2}\,\text{erf}^{12}\left(\sqrt{2}\pi\sigma\right) - 4\pi^3\sigma^6 e^{-24\pi^2\sigma^2}\,\text{erf}^6\left(2\sqrt{2}\pi\sigma\right)}{2^{18}\pi^6\sigma^{12}}. \tag{2.85}$$

However, Eq. (2.78) remains unchanged, i.e.

$$f_{\text{NAND}}(k \neq k^*) \sim \mathcal{N}\left(0, \sqrt{M\left(\frac{1}{2} + \sigma^2\right) + \sigma^2}\right). \tag{2.86}$$

Repeating a similar analysis to estimate $\Pr[f_{\text{NAND}}(k^*) < f_{\text{NAND}}(k \neq k^*)]$ yields the number of moduli

$$\begin{aligned} M(\epsilon) &\approx \frac{2^{18}\pi^6 e^{12\pi^2\sigma^2}\sigma^{12}\left(4\sigma^2+1\right)\log\left(\frac{3}{\epsilon}\right)}{\operatorname{erf}^{12}\left(\sqrt{2}\pi\sigma\right)}, \\ &= O\left(e^{\beta\sigma^2}\log(1/\epsilon)\right), \end{aligned} \tag{2.87}$$

consistent with the results of Section 2.2.3.

To also account for synaptic failure with probability p, we must modify Eqs. (2.84) and (2.86) to include the possibility of this discrete mode of noise. While a functional synapse with additive Gaussian noise returns value $y+\xi$, a synaptic failure returns value 0. A careful treatment of synaptic failure is provided in Appendix A.2, the result of which is a new set of distributions $f'_{\text{NAND}}(k \neq k^*)$ and $f'_{\text{NAND}}(k = k^*)$ which depend on both the strength of Gaussian errors σ and on the probability of synaptic failures p. As with Eqs. (2.84) and (2.86), $f'_{\text{NAND}}(k \neq k^*)$ is centered at zero with standard deviation $O(\sqrt{M})$, and $f'_{\text{NAND}}(k = k^*)$ is centered at $O(M)$ with standard deviation $O(\sqrt{M})$.

In order to proceed, we must take a more careful treatment of the activation function required for the error correction step of the logical neuron. In a biological discussion of the grid code, winner-take-all dynamics are often used to describe the decoding process [SF11], i.e. it is assumed that the only neuron activated is that representing the decoded value with the largest signal, as per maximum likelihood decoding approach discussed in Section 2.2.3. This decoding approach implicitly assumes communication between the decoding neurons, e.g. through an argmax-type non-linearity. However, for transparency in the treatment of noise, we demonstrate how a local step activation function, parameterized by a cutoff c, can replace winner-take-all dynamics with a simpler decoder.

Because the separation of means of the correct and incorrect decoding distributions scales as $O(M)$ compared to their standard deviations which scale as $O(\sqrt{M})$, an appropriate choice of threshold c is sufficient to distinguish between the two distributions with high probability (for large M). Since there are three decoding neurons, a correct decoding requires the correct neuron sampled from $f'_{\text{NAND}}(k = k^*)$ to exceed c and the two incorrect neurons sampled from $f'_{\text{NAND}}(k \neq k^*)$ to lie below c. Evaluating such probabilities is straightforward due to $f'_{\text{NAND}}(k)$ being normally distributed in both cases. The probability that the logical NAND neuron succeeds is given by $1 - \epsilon(\sigma, p)$, where $\epsilon(\sigma, p)$ is an error rate that depends on both the strength of Gaussian errors σ and the probability of synaptic failures p. A more detailed analysis of $\epsilon(\sigma, p)$, including its explicit expression, is provided in Appendix A.2.

To obtain a fault-tolerance threshold from this quantity, we apply the result of Evans and Pippenger [EP98] for fault-tolerant Boolean formulas built from NAND gates. Evans and Pippenger present a construction for that Boolean formulas built from NAND gates that achieves fault-tolerance if and only if the NAND probability of failure is below $\epsilon_0 = (3-\sqrt{7})/4$. We appeal to this bound to prove fault tolerance of neural NAND gates, which is equivalent to placing the grid code inside the code of Evans and Pippenger. While their NAND construction only considers errors as bit flips – i.e., an error is triggered if a gate that should return 0 returns a 1, and vice versa – errors in the neural NAND gate are biased. This occurs because synaptic failures bias neurons towards zero output; if all neurons fail, the neural NAND defaults to 0. However, biased errors are strictly easier to correct than unbiased errors, and thus the threshold of Evans and Pippenger serves as an appropriate lower bound. To ensure the lower bound is applied correctly, we report the error rate of the neural NAND in a manner that counts zero output forced by synaptic failure as an error.

We use both this bound and the expression for $\epsilon(\sigma, p)$ (see Appendix A.2) to analytically determine a fault-tolerance threshold for p and σ at $M = 10^5$ moduli. We analytically plot the neural NAND failure probability $\epsilon(\sigma, p)$ in Fig. 2.11a, as well as a contour (the dashed line) corresponding to the logical error being equal to the aforementioned threshold $\epsilon_0 = (3 - \sqrt{7})/4$. This plot indicates a region of σ, p with logical error $\epsilon(\sigma, p) < \epsilon_0$, within which fault-tolerant computation is achievable, and a sharp transition to a region with $\epsilon(\sigma, p) > \epsilon_0$ in which this fault-tolerant construction does not hold. In aggregate then, by appealing to the universality of the NAND gate, we have that for sufficiently small $\sigma < \sigma_0$ and $p < p_0$

(where σ_0 and p_0 may be determined by the contour of Fig. 2.11a), our fault-tolerant neuron may achieve fault-tolerant computation with polylogarithmic overhead, thus achieving neural network fault-tolerance as introduced in Section 2.2.1.

Concatenating grid code on top of repetition code

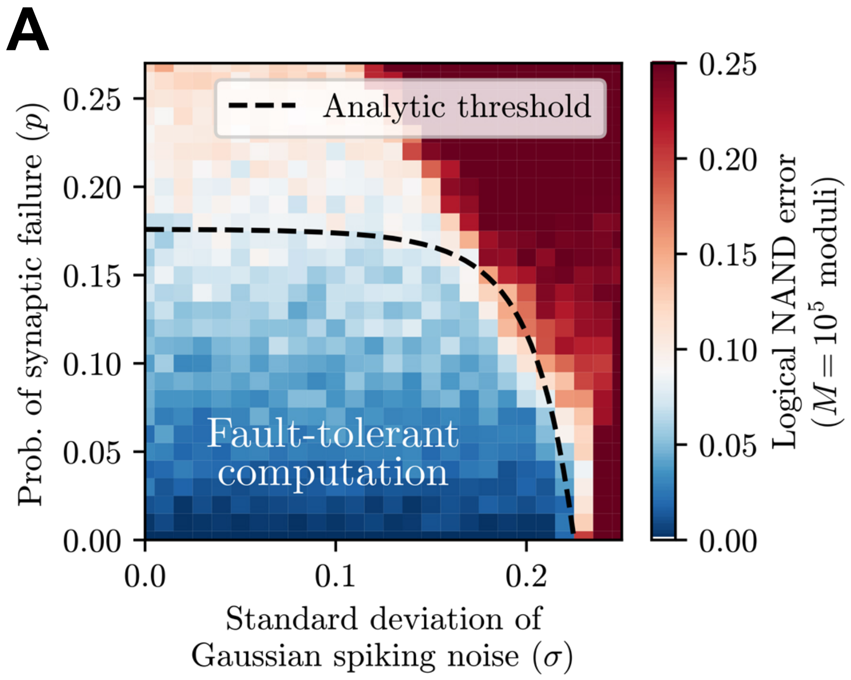
Above, we constructed a neural NAND that uses M moduli, where each modulus is stored without redundancy. While our constructions above have assumed $M = 10^5$, the grid cells in the mammalian cortex contain far fewer moduli, i.e. $M \sim 10$ [FBB08]. However, in the biological setting, each modulus is itself encoded redundantly, with $R \sim 10^3$ to 10^4 repetitions of each modulus [FBB08]. This effectively concatenates the grid code on top of a repetition code, which provides another means by which to decrease the strength of the additive Gaussian noise. Roughly speaking, the central limit theorem reduces the variance σ^2 to σ^2/R, which can drastically reduce the number of moduli required to suppress noise (Fig. 2.10c). To move towards a more biologically feasible setting, we examine a concatenation of the grid code on top of a repetition code. To develop a precise construction, this requires a couple modifications to the above neural NAND gate.

Considering the logical neuron in Fig. 2.9a, the main modification is to replace each phase with R copies of the phase. Each successive layer then averages over the repetitions of the previous layer, correcting for the synaptic failure probability. For example, consider the neural NAND gate with R copies of the first phase $\{\theta^{(1)}_{1,i}\}_{i=1}^{R}$ and R copies of the second phase $\{\theta^{(2)}_{1,i}\}_{i=1}^{R}$. In the absence of the repetition code, the phase ϕ_1 would be computed as $\phi_1 = \theta^{(1)}_1 + \theta^{(1)}_2$ as the neural NAND gate uses logical weights equal to 1. With the repetition code and a synaptic failure probability p, we instead choose

$$\phi_{1,j} = \frac{1}{R} \times \frac{1}{1-p} \sum_{i=1}^{R} \theta^{(1)}_{1,i} + \theta^{(2)}_{1,i}\,, \tag{2.88}$$

which may be implemented by selecting weights $\frac{1}{R} \times \frac{1}{1-p}$, where the factor $\frac{1}{1-p}$ accommodates

A



B

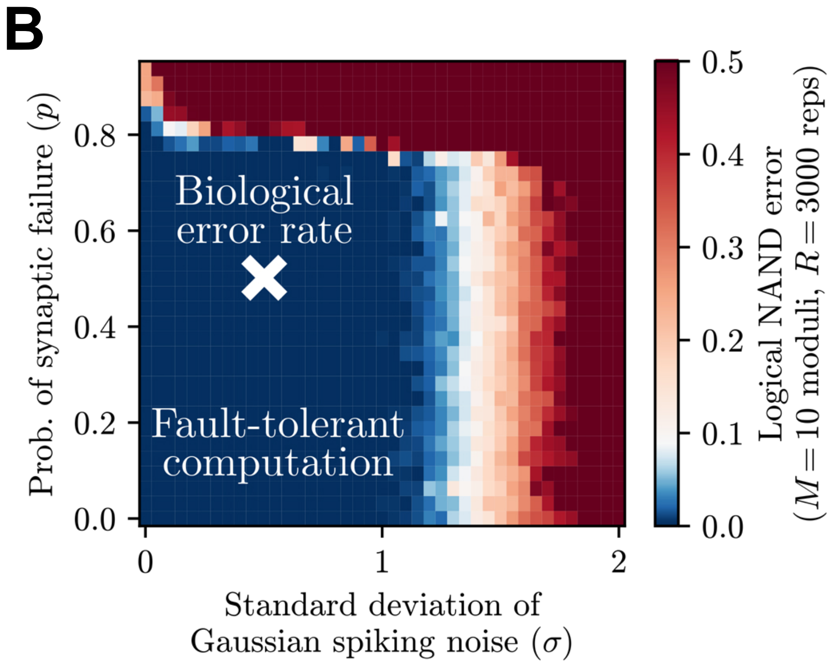


Figure 2.11: (A) The logical error of a neural NAND gate using only the grid code. Also plotted is an analytical fault-tolerance threshold corresponding to NAND error probability $\epsilon_0 \approx 0.09$ required to achieve an arbitrarily low logical error using the optimal NAND fault-tolerance construction of Ref. [EP98]. The region in blue supports fault-tolerant computation, while the region in red suffers faulty computation. (B) Logical error of a neural implementation of the NAND gate using the grid code ($M = 10$ moduli) concatenated with a repetition code ($R = 3000$ repetitions). Notably, the fault-tolerant regime where the error falls below $\epsilon_0 \approx 0.09$ (blue) encompasses biological error rates (white cross).

for synaptic failure. This ensures that i.i.d. sampling of the Gaussian noise over R repetitions will reduce the variance from σ^2 to σ^2/R.

The only remaining modification is to use a step function in the encoder to perform a majority vote over repetitions in the final layer. The goal is to ensure that the character of the noise remains the same after decoding, i.e. the noise after error correction should be describable as a combination of logical bit flips and continuous Gaussian noise. As in Section 2.2.4, consider a set of three three codewords $x_1 = 0$, $x_2 = a$, and $x_3 = 2a$, interpreting x_1 as False and x_2 as True. In the previously studied construction where $R = 1$, the outputs of the error corrected NAND gate corresponding to ϕ_j are simply multiplied by weights $1/\lambda_j$, $1/\lambda_j$, and 0 respectively (see Eqs. (2.82) and (2.83) and Fig. 2.10a). In addition to rescaling by $\frac{1}{1-p}$ to account for synaptic failures, we include an extra discretization step in the encoding stage of error correction (as in Fig. 2.9a). This is accomplished by choosing $e(x_k)$ to be a step function in the encoder of Eq. (2.69). If x_1 is recovered by the decoder, we re-encode $e(x_1) = 0$; and if x_2 is recovered by the decoder, we re-encode $e(x_2) = a/(1-p)$. Since the weight associated with x_3 is zero, the decoding neuron corresponding to x_3 is not connected to the following layer of neurons.

We conduct numerical experiments on this neural NAND gate with redundantly encoded moduli, using $M = 10$ moduli and $R = 3 \times 10^3$ repetitions to remain in the biologically relevant regime. As before, errors are biased due to synaptic failure setting neurons to zero; hence, the threshold of Evans and Pippenger places a lower bound on the true threshold of the neural NAND, where zero output incurred by synaptic failure is appropriately counted as a logical error. Since the central limit theorem performs poorly on the small number of moduli $M = 10$ here, an analytic approximation like that for $\epsilon(\sigma, p)$ (Eq. (A.7)) does not exactly hold. Instead, we numerically estimate the threshold as the contour where the logical NAND error crosses the Evans and Pippenger threshold $\epsilon_0 = (3 - \sqrt{7})/4 \approx 0.09$. We showcase results in Fig. 2.11b, with the threshold contour depicted as the white boundary separating the blue region (which represents fault-tolerant computation) and the red region (which represents faulty computation). This indicates approximate thresholds $\sigma_0 \approx 1.4$ and $p_0 \approx 0.7$. Notably, the fault-tolerant regime encompasses the observed biological error rates (depicted as a white cross) of $\sigma \approx 0.5$ (given a mean of approximately 0.5, due to random outputs in $[0, 1]$) and $p \approx 0.5$ [SW94; HSM93; SK93], thus suggesting that the grid code augmented with a repetition code suffices to enable reliable computation in faulty organisms.

2.2.5 Concluding remarks

In this work, we have demonstrated fault-tolerant constructions for neural networks subject to synaptic failure (Section 2.2.2) and additive Gaussian noise (Section 2.2.3). While synaptic failure is a digital error and may be treated with a traditional repetition code, Gaussian noise represents an analog error, which we treat using the more sophisticated grid code that emerged from studies of the mammalian cortex. We have further used these constructions to build neural networks that can reliably implement any Boolean formula in the presence of both errors modes (Section 2.2.4). In particular, for sufficiently small synaptic failure probability $p < p_0$ and Gaussian noise standard deviation $\sigma < \sigma_0$, our construction enables the computation of arbitrary Boolean formulas (and thus arbitrary computation) with only polylogarithmic overhead, thus achieving neural network fault-tolerance as introduced in Section 2.2.1. These results ultimately describe a phase transition from faulty neural computation into fault-tolerant neural computation.

Our analyses only place a lower bound on the fault-tolerance threshold of neural computation; a more effective neural fault-tolerant construction may be exist. In particular, while the neural network fault-tolerance theorem is phrased in terms of digital Boolean gates composed of analog neurons, the fault-tolerant neural network size requirement of Section 2.2.3 (Eq. (2.81) to be precise) holds for a general construction of neural networks with Gaussian-distributed weights. This standard form of artificial neural networks provides a more direct analog approach to computation without introducing logical digital gates, and it may ultimately realize a more efficient path towards a threshold for the fault-tolerant phase of neural computation.

Framed against the slowing pace of Moore's Law and increasingly prohibitive energy costs of deep learning [Bro+20; JEP+21], the remarkable efficiency of biological computation places central importance on a deep understanding of noisy analog systems. The brain is a canonical example of a noisy analog system that is more energy-efficient than traditional faultless computation. By demonstrating the existence of fault-tolerant neural networks, our work provides a concrete path towards leveraging the favorable properties of such analog neural networks in a neuromorphic setting [Ind+11; Ess+16; Wan+18]. These results may also find use in novel hardware for machine learning acceleration, such as optical computing [McM23] and thermodynamic computing [Con+19], which may achieve more resource-efficient computations at the expense of increased error. In aggregate, our findings are suggestive of the power of naturally occurring error-correcting mechanisms: while the presence of fault-tolerant computation in the brain remains uncertain without experimental verification, we conclude that observed neural error correction codes are sufficient to achieve arbitrarily reliable computation.

Chapter 3

Fault-tolerance in quantum models of computation

While the study of fault-tolerance in classical computation is largely a theoretical exercise with little bearing on the state of the practice, hardware-level fault-tolerance is considered a crucial component of quantum computer architectures and is a central thrust of modern quantum computing research [CTV17]. Most quantum fault-tolerance research focuses on designing reliable quantum circuits subject to noise at the gate-level. Recent work theoretical work has on the quantum singular value transformation subroutine has improved our understanding of how more complicated subroutines may be constructed [Gil+19], potentially providing an alternative way to synthesize quantum computations.

Again in keeping with the theme of studying models of error correction and fault-tolerant design native to less conventional models of computation (Q1), we envision a fault-tolerant computer composed of subroutines—rather than simple gates. To that end, we study an error error correction scheme native to the quantum signal processing subroutine. We show that certain systematic errors in quantum signal processing can be mitigated through redundancy in time. We also rigorously discuss the limitations of such an error correction scheme without access to additional resources. Though not a full hardware-level fault-tolerance scheme per se, our technique may be used in conjunction with traditional quantum circuit-based approaches to achieve fault-tolerance. The introduction of Section 3.1[1], provides an overview of an error correction scheme that works at the level of the algorithm. The remainder of the Chapter[2] expands on the technical results and presents analysis in a novel diagrammatic notation for the analysis of noisy quantum signal processing operators.

3.1 Introduction

Quantum signal processing (QSP) [LYC16; LC19] and its generalization, the quantum singular value transform (QSVT), have provided a framework unifying many important quantum algorithms [Gil+19; Mar+21]. Under this framework, Grover's search [Gro97; Gro98], the quantum Fourier transform [Cop02]—the basis of Shor's factoring algorithm [Sho94]—and quantum simulation algorithms [CW12; Chi+21; LC17; Mar+21; Ber+15] are all described by interleaving sequences of block-encoded signal rotations and single-qubit signal processing rotations.

However, unless the QSVT operators are constructed on top of a fault-tolerant quantum computer [Sho96; AB97; KLZ98; Pre98; Kit03], inherent experimental noise in quantum devices limits the length of realizable QSP and QSVT sequences. Even with a fault-tolerant quantum computer, errors may still arise due to inherent approximations and truncation made in constructing the block-encoding of the subsystem of interest [CV20a; CV22]. These observations lead to an important question: how does one correct errors in a typical QSVT sequence? Of course one may employ existing gate-level error correction methods to every gate in a QSVT circuit, but the unifying perspective of representing quantum operations as polynomial transformations offers an entirely new possibility for studying error correction at

[1] This has appeared as "Error correction of quantum algorithms: arbitrarily accurate recovery of noisy quantum signal processing," by Andrew K. Tan, Yuan Liu, Minh C. Tran, and Isaac L. Chuang in arXiv preprint arXiv:2301.08542 (2023) [Tan+23a].

[2] This has appeared as "Perturbative model of noisy quantum signal processing," by Andrew K. Tan, Yuan Liu, Minh C. Tran, and Isaac L. Chuang in Phys. Rev. A 107, 042429 (2023) [Tan+23b].

the level of the algorithm.

To motivate the study of error correction at the algorithm-level, we introduce a noise model for QSP describing a generic perturbative noise on the signal processing operation. Our companion paper [Tan+23a] expands on the concept of algorithm-level error correction, using the specific coherent error of Section 3.4 as an example, and provides numerical results including an application to a modified Grover's fixed point amplification algorithm. Here we provide a full derivation of the results stated in [Tan+23a] using a novel diagrammatic notation, and introduce a broader class of errors where this notation is useful. This general error model has the advantage of being able to additionally capture incoherent errors.

The rest of the paper is organized as follows. In Section 3.2, we review the QSP framework for quantum algorithms, introduce our general error model, and setup some useful nomenclature. In Section 3.3, we introduce our diagrammatic notation. To demonstrate the utility of our diagrammatic notation, we introduce a specific model of coherent error in Section 3.4 using the notation to construct a scheme for error recovery. We conclude the paper with a discussion of incoherent errors and directions for future work in Section 3.5.

3.2 Preliminaries

We start in Section 3.2.1 with a brief review of QSP and with a specification of the conventional choices made in this paper. This is followed in Section 3.2.2 with the introduction of our model of signal processing noise. Finally, we define the error channel in Section 3.2.3 and introduce notation for the perturbative decomposition of its Kraus operators in Section 3.2.4.

3.2.1 Quantum Signal Processing

A number of conventions exist in the literature surrounding QSP primarily differing in the choice of bases and signal operators. We specify our choices below, which correspond to the 'Wx' convention of [Mar+21].

A length-d QSP operator is parameterized by phases $\vec{\phi} = (\phi_0, \ldots, \phi_d) \in \mathbb{R}^{d+1}$,

$$U_0(\theta; \vec{\phi}) = \mathrm{QSP}(\theta; \vec{\phi}) \equiv e^{i\phi_0 Z} \prod_{j=1}^{d} W(\theta) e^{i\phi_j Z}, \tag{3.1}$$

where the signal operator is a rotation in the X-basis

$$W(\theta) \equiv e^{i\theta X} = \begin{pmatrix} \cos\theta & i\sin\theta \\ i\sin\theta & \cos\theta \end{pmatrix}, \tag{3.2}$$

and X, Y, and Z are the Pauli matrices. The subscript 0 on U_0 is used to indicate that the QSP is noiseless. A general length-d QSP sequence takes the form

$$U_0(\theta; \vec{\phi}) = \begin{pmatrix} P(a) & iQ(a)\sqrt{1-a^2} \\ iQ^*(a)\sqrt{1-a^2} & P^*(a) \end{pmatrix}, \tag{3.3}$$

where $a \equiv \cos\theta$ and $P, Q \in \mathbb{C}[a]$ are polynomials such that [Gil+19, Theorem 4]:

(i) $\deg(P) \leq d$ and $\deg(Q) \leq d-1$;

(ii) P has parity-($d \mod 2$) and Q has parity-($d-1 \mod 2$);

(iii) $\forall a \in [-1, 1] : |P(a)|^2 + (1-a)^2 |Q(a)|^2 = 1$.

We write $U = [\![P_U, Q_U]\!]$ as a shorthand for Eq. (3.3), dropping the subscripts when the QSP unitary is clear by context.

3.2.2 Noise model

We consider a generic noise in the signal processing basis. For length-d QSP with signal processing rotations indexed by $0 \leq j \leq d$, the error of a single signal processing rotation j is characterized by a set of Kraus operators $\{N_\epsilon^{(j,i_j)}\}$ for $i_j \in \{1, \ldots, M_j\}$ where M_j is the number of Kraus operators describing the channel at site j. In this work, we only consider

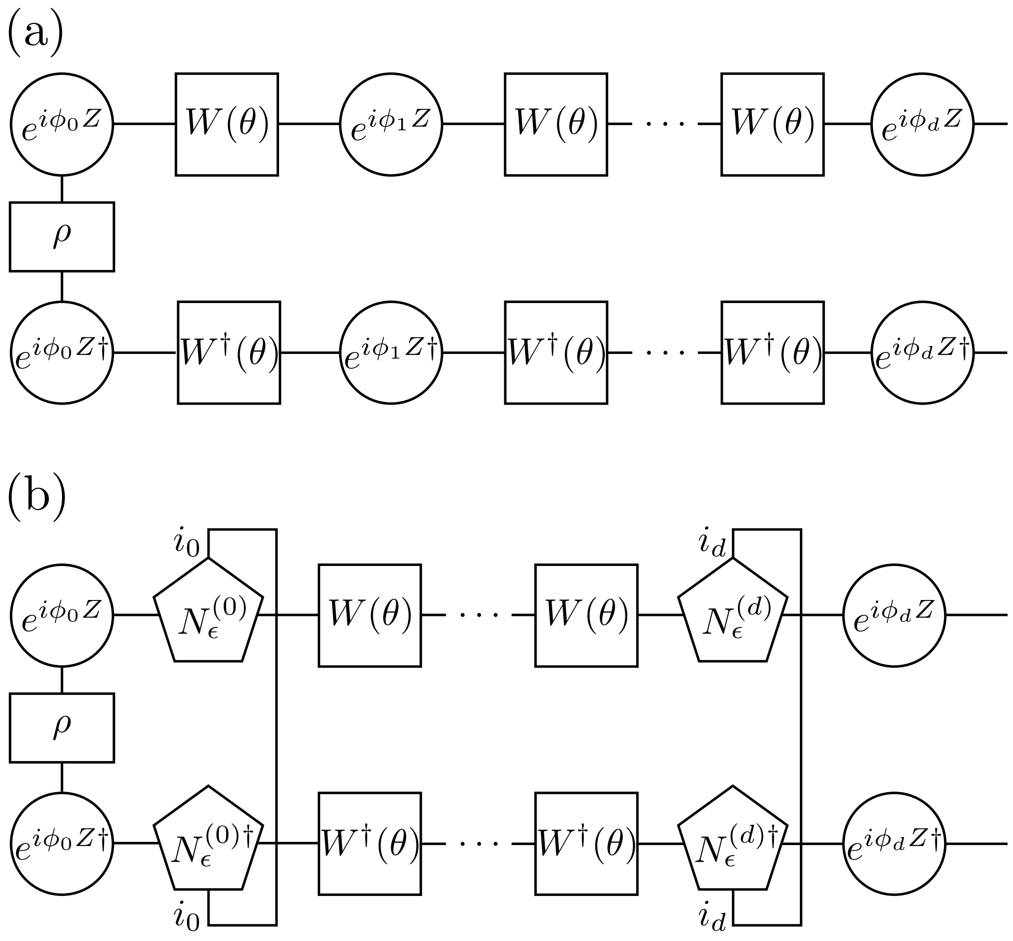


Figure 3.1: Tensor diagram depicting the result of (a) a noiseless and (b) a noisy length-d QSP operator parameterized by phases $\phi_0, \ldots, \phi_d \in \mathbb{R}$ on input state ρ. The noise on the signal processing rotations is characterized by Kraus operators $N_\epsilon^{(j,i_j)}$ for $0 \leq j \leq d$ and is contracted over Kraus operator index i_j. In anticipation of the notation of Section 3.3, we use circles to denote signal processing rotations and squares to denote signal rotations; additionally, pentagons are used to denote Kraus operators and rectangles are used for density matrices.

errors in the signal processing basis with Kraus operators of the the form

$$N_\epsilon^{(j,i_j)} = w_\epsilon^{(j,i_j)}(\phi_j)I + iz_\epsilon^{(j,i_j)}(\phi_j)Z, \tag{3.4}$$

for complex functions $w_\epsilon^{(j,i_j)}$ and $z_\epsilon^{(j,i_j)}$ satisfying the completeness condition $\sum_{i_j} N_\epsilon^{(j,i_j)\dagger} N_\epsilon^{(j,i_j)} = I$ for all j.

Note that in general, we allow the Kraus operators to depend on ϕ_j, but they are assumed to be independent of θ. Further, to allow a perturbative analysis, we assume each Kraus operator also depends on a parameter $\epsilon \ll 1$, and that all $N_\epsilon^{(j,i_j)}$ approach a value proportional to identity as $\epsilon \to 0$.

Given input ρ, such a noisy QSP produces output state

$$\rho'_\epsilon = \sum_{i_0,\ldots,i_d} \left(e^{i\phi_0 Z} N_\epsilon^{(0,i_0)} \prod_{j=1}^{d} W(\theta) e^{i\phi_j Z} N_\epsilon^{(j,i_j)} \right)^\dagger \rho \left(e^{i\phi_0 Z} N_\epsilon^{(0,i_0)} \prod_{j=1}^{d} W(\theta) e^{i\phi_j Z} N_\epsilon^{(j,i_j)} \right), \tag{3.5}$$

which is depicted in Fig. 3.1.

3.2.3 Error channel

We express the result of applying the entire noisy QSP sequence as a channel

$$\mathcal{U}_\epsilon(\theta; \vec{\phi})(\rho) = \rho'_\epsilon, \tag{3.6}$$

with Kraus operators

$$M_\epsilon^{(i_0,\ldots,i_d)} = e^{i\phi_0 Z} N_\epsilon^{(0,i_0)} \prod_{j=1}^{d} W(\theta) e^{i\phi_j Z} N_\epsilon^{(j,i_j)}. \tag{3.7}$$

In order to isolate the effect of the error, we define the *error channel* that produces erroneous state ρ'_ϵ from the ideal result ρ'_0

$$\mathcal{E}_\epsilon(\rho'_0) = \mathcal{E}_\epsilon(U_0^\dagger \rho U_0) \equiv \rho'_\epsilon. \tag{3.8}$$

Combining Eqs. (3.6) to (3.8), we find that the error channel can be written with Kraus operators

$$E_\epsilon^{(i_0,\ldots,i_d)} = U_0^\dagger e^{i\phi_0 Z} N_\epsilon^{(0,i_0)} \prod_{j=1}^{d} W(\theta) e^{i\phi_j Z} N_\epsilon^{(j,i_j)}. \tag{3.9}$$

The error channel has the benefit of being nearly the identity channel in the perturbative regime; that is as $\epsilon \to 0$, we have $\mathcal{E} \to \mathrm{id}$. As a result, all of its its Kraus operators approach a value proportional to the identity matrix, and we can write its Kraus operators in the form

$$\alpha^{(i_0,\ldots,i_d)} I + \epsilon A^{(i_0,\ldots,i_d)} + O(\epsilon^2), \tag{3.10}$$

for $\alpha \in \mathbb{R}$ where A is an operator of the form:

Definition 15 (Standard form, first-order)**.** We say an operator A is in first-order *standard form* of degree-$2d$ if it is written as a weighted sum over QSP operators generated by conjugation of $e^{i\frac{\pi}{2}Z}$

$$\begin{aligned} A = \beta_d &\times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, -\phi_{d-1}, \ldots, -\phi_2, -\phi_1, \pi, \phi_1, \phi_2, \ldots, \phi_{d-1}, \phi_d)) \\ &+ \beta_{d-1} \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, -\phi_{d-1}, \ldots, -\phi_2, \pi, \phi_2, \ldots, \phi_{d-1}, \phi_d)) \\ &+ \ldots \\ &+ \beta_1 \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, \pi, \phi_d)) \\ &+ \beta_0 \times \mathrm{QSP}(\theta; (\pi/2)). \end{aligned} \tag{3.11}$$

where $\beta_i \in \mathbb{R}$ and $\phi_i \in \mathbb{R}$.

The component QSPs in the sum of Definition 15 take on a special form:

Definition 16 (Error component, first-order)**.** Let U be length-d QSP operator parameterized by real phases $(\phi_0, \ldots, \phi_d)$. Then a length-$2r$ QSP V with $1 \leq r \leq d$ is said to be a *first-order error component of QSP U* if it can be written in the following form:

$$V = \text{QSP}(\theta; (-\phi_d - \pi/2, \ldots, -\phi_{d-r+1}, \pi, \phi_{d-r+1}, \ldots, \phi_d)). \tag{3.12}$$

Furthermore, it is assumed that no ϕ_i for $i < d$ is a half-integer multiple of π; otherwise, we can perform elision to simplify the diagram.

A generic error channel Kraus operator is depicted in Fig. 3.1 using the notation of Section 3.3.

3.2.4 Canonical profile

It will often be useful to write an operator in the basis of Pauli matrices. The coordinates of noisy QSP operators in the Pauli basis will generically be polynomials of $\cos\theta$.

Certain operators, such as the Kraus operators of noisy QSP error channels subject to noise of the form in Eq. (3.4), can be written in a special form. We say that an operator $U_\epsilon(\theta)$ admits a canonical expansion if we can write $U_\epsilon = w_\epsilon(\theta)I + i[x_\epsilon(\theta)X + y_\epsilon(\theta)Y + z_\epsilon(\theta)Z]$, for functions $w_\epsilon, x_\epsilon, y_\epsilon, z_\epsilon$ of the form

$$w_\epsilon(\theta) = \cos^2\theta \sum_{k=0}^{\infty} \epsilon^k \sum_{j=-1}^{\infty} \mathcal{P}_j^{(0,k)} \cos^{2j}(\theta), \tag{3.13a}$$

$$x_\epsilon(\theta) = \sin(2\theta) \sum_{k=0}^{\infty} \epsilon^k \sum_{j=0}^{\infty} \mathcal{P}_j^{(x,k)} \cos^{2j}(\theta), \tag{3.13b}$$

$$y_\epsilon(\theta) = \sin(2\theta) \sum_{k=0}^{\infty} \epsilon^k \sum_{j=0}^{\infty} \mathcal{P}_j^{(y,k)} \cos^{2j}(\theta), \tag{3.13c}$$

$$z_\epsilon(\theta) = \cos^2\theta \sum_{k=0}^{\infty} \epsilon^k \sum_{j=-1}^{\infty} \mathcal{P}_j^{(z,k)} \cos^{2j}(\theta), \tag{3.13d}$$

and $\mathcal{P}_j^{(\sigma,k)} \in \mathbb{R}$ for all $\sigma \in \{0, x, y, z\}$ and $j, k \in \mathbb{Z}$. We call $\mathcal{P}$ the *canonical profile* of U_ϵ. For convenience, we allow $j \in \mathbb{Z}$ and define $\mathcal{P}_j^{(x,k)} = \mathcal{P}_j^{(y,k)} = \mathcal{P}_{j-1}^{(z,k)} = 0$ for all $j < 0$ and k.

Our parameterization, particularly the choice of factoring out $\sin(2\theta)$ from the X and Y components, $\cos^2\theta$ from the Z component, and starting the sum of the Z component at $j = -1$, is tailored to the diagrams which appear in the Kraus operators of QSP error channels (we leave the proof of this to Appendix B.1). Note that for unitary U_ϵ, the functions satisfy the following completeness relationship $w_\epsilon(\theta)^2 + x_\epsilon(\theta)^2 + y_\epsilon(\theta)^2 + z_\epsilon(\theta)^2 = 1$ which holds for all θ and ϵ.

3.3 Diagrammatic notation

In this Section, we develop a diagrammatic notation for visualizing quantum signal processing unitaries and demonstrate their utility for reasoning about the Kraus operators of noisy QSP channels.

First, we prove a number of results to motivate the notation and provide a number of basic manipulations.

For QSP U of length-d parameterized by $\vec{\phi} \equiv (\phi_0, \dots, \phi_d)$, we will use notation $\vec{\phi}_{i:j}$ to denote the sub-sequence $(\phi_i, \dots, \phi_j)$, and $U^{(j)}$ to denote the length-j QSP parameterized by $\vec{\phi}_{0:j}$.

Lemma 17 (Unit steps)**.** Let $U_0 = [\![P, Q]\!]$ be a length-d QSP unitary and $k = \deg(P) \le d$. The unitary $U_0' = U_0 e^{i\phi_0 Z} W e^{i\phi_1 Z} = [\![P', Q']\!]$ is a length-$(d+1)$ QSP unitary with either $\deg(P') = k - 1$ or $\deg(P') = k + 1$.

Proof. Computing the product, we find

$$P'(a) = e^{i(\phi_0+\phi_1)} \left(aP(a) - \left(1 - a^2\right) Q(a) e^{-2i\phi_0}\right). \tag{3.14}$$

Since $\deg(P) = k$ by assumption and $\deg(Q) = k - 1$, it must be that $\deg(P') \le k + 1$.

Next, we prove that $\deg(P') \ge k-1$ by contradiction. Assume that $\deg(P') = w < k-1$.

We can then iterate our construction above choosing $U_0'' = U_0' e^{-i(\phi_1 - \frac{\pi}{2})Z} W e^{-i(\phi_0 - \frac{\pi}{2})Z}$ and, by the above argument, we have $\deg(P'') \leq w + 1 < k$. But we have chosen the additional phases such that $U_0'' = U_0 I_0$ where $I_0 = \text{QSP}(\theta; (\phi_0, \pi/2, -\phi_0 + \pi/2))$ is an unbiased operator, i.e. $I_0 = I$. Therefore, $\deg(P'') = k$. This is a contradiction and so it must be that $\deg(P') \geq k - 1$.

Furthermore, we have that $\deg(P') \neq k$ by parity constraints. Therefore, the Lemma follows. □

Definition 18 (QSP degree peak)**.** Let R be an unbiased QSP sequence of length $d \geq 2$ parameterized by $(\phi_0, \ldots, \phi_d) \in \mathbb{R}^{d+1}$. Suppose for some $0 < i < d$ we have $\deg\left(P_{R^{(i)}}\right) = r+1$ and $\deg\left(P_{R^{(i-1)}}\right) = \deg\left(P_{R^{(i+1)}}\right) = r$. We will call position i a *degree peak* of R.

Lemma 19 (QSP elision)**.** Let R be a QSP operator of length $d \geq 2$ parameterized by $(\phi_0, \ldots, \phi_d)$. Position i is a degree peak of R if and only if $\phi_i = \pi\left(n + \frac{1}{2}\right)$ for some $n \in \mathbb{Z}$.

Additionally, R is equivalent to a length-$(d-2)$ QSP parameterized by phases

$$(\phi_0, \ldots, \phi_{i-2}, \phi_{i-1} + \phi_i + \phi_{i+1}, \phi_{i+2}, \ldots, \phi_d). \tag{3.15}$$

We refer to this transformation as *QSP elision.*

Proof. Writing out the product, we find the following relationship between QSP polynomials of $R^{(i-1)}$ and $R^{(i)}$:

$$P_{R^{(i)}} = a e^{i(\phi_{i-1} + \phi_i)} (P_{R^{(i-1)}} + e^{-2i\phi_{i-1}} Q_{R^{(i-1)}}) + \Theta(a^{r-1}). \tag{3.16}$$

By assumption $\deg(P_{R^{(i)}}) = r + 1$ and therefore $\deg(P_{R^{(i-1)}} + e^{-2i\phi_{i-1}} Q_{R^{(i-1)}}) = \deg(P_{R^{(i)}}) - 1 = r$.

Writing out the product for $R^{(i+1)}$, we find

$$P_{R^{(i+1)}} = 2a^2 \cos(\phi_i) e^{i(\phi_{i-1} + \phi_{i+1})} (P_{R^{(i-1)}} + e^{-2i\phi_{i-1}} Q_{R^{(i-1)}}) + \Theta(a^r). \tag{3.17}$$

Looking at the equation above and comparing with the $P_{R^{(i)}}$ we find that if $\deg(P_{R^{(i+1)}}) = r$, it must be that $\cos(\phi_i) = 0$ or equivalently $\phi_i = \pi\big(n + \frac{1}{2}\big)$ for some $n \in \mathbb{Z}$. The converse is also true.

Assuming $\phi_i = \pi(n + \frac{1}{2})$, we find,

$$P_{R^{(i+1)}} = e^{i(\phi_{i-1}+\phi_i+\phi_{i+1})} P_{R^{(i-1)}}, \tag{3.18}$$

This transformation is equivalent to a Z-rotation by $\phi_{i-1} + \phi_i + \phi_{i+1}$ (a length-0 QSP). We can therefore elide the original QSP sequence by replacing the three phases ϕ_{i-1}, ϕ_i, and ϕ_{i+1} with a single phase $\phi_{i-1} + \phi_i + \phi_{i+1}$, thus proving the Lemma. □

A number of useful corollaries follow from Lemma 19 including the construction of inverse QSP operators.

Corollary 20 (Inverse QSPs)**.** Let U be length-d QSP operator parameterized by phases $(\phi_0, \ldots, \phi_d)$. The length-d QSP U' parameterized by $(-\phi_d + \frac{\pi}{2}, -\phi_{d-1}, \ldots, -\phi_1, -\phi_0 - \frac{\pi}{2})$ is the inverse QSP sequence in the sense that $UU' = U'U = I$.

Additionally, Lemma 19 gives us the following uniqueness result for QSP parameterization:

Corollary 21 (Uniqueness of QSP parameterization)**.** Let $U = \mathrm{QSP}(\theta; \vec{\phi})$ be a length-d QSP and let $V = \mathrm{QSP}(\theta; \vec{\psi})$ be a length-d' QSP. Further assume such that no phase ϕ_i, ψ_j is a half-integer multiple of π. Then $U = e^{i\chi}V$ for some global phase $\chi \in [0, 2\pi)$ if and only if $d = d'$ and for all $0 \le i \le d$, $\psi_i - \phi_i = \pi n_i$ for some $n_i \in \mathbb{Z}$. Furthermore, either $\chi = 0$ or $\chi = \pi$.

Proof. The $\Longleftarrow$ direction is a straightforward consequence of $e^{i\pi Z} = -I$ and so we focus on $\Longrightarrow$.

Since by assumption, neither $\vec{\phi}$ nor $\vec{\psi}$ contain a half-integer multiple of π, Lemma 19 implies that neither contain any degree peaks and therefore $\deg(P_U) = d$ and $\deg(P_V) = d'$.

As a result, the QSP unitaries must be of the same length $U = V \implies P_U = P_V \implies \deg(P_U) = \deg(P_V) \implies d = d'$. We can therefore limit our consideration to the case of $d = d'$.

Now we show that the phases must be equivalent up to an integer multiple of π inductively. First consider the case where $d = 0$. In this case, $U = e^{i\chi} V \implies e^{i\phi_0 Z} = e^{i\chi} e^{i\psi_0 Z} \implies \phi_0 = \psi_0 + \pi n_0$ for some $n_0 \in \mathbb{Z}$; furthermore, $\chi = 0$ for n_0 even and $\chi = \pi$ for n_0 odd. Thus the Lemma is satisfied for $d = 0$.

Assuming the Lemma for QSP unitaries of length-d, we show that it holds for QSP unitaries of length-$(d+1)$. Consider QSPs $U = \mathrm{QSP}(\theta; \vec{\phi})$ and $V = \mathrm{QSP}(\theta; \vec{\psi})$ each of length-$(d+1)$ satisfying the conditions of the Lemma. Given that $U = V$, we reduce U to a length-d QSP by right-multiplying both sides by a QSP inverse (Corollary 20) of its final signal processing step $(We^{i\phi_{d+1} Z})^{-1} = e^{-i(\phi_{d+1} - \frac{\pi}{2})Z} W e^{-i\frac{\pi}{2} Z}$. The result is

$$U = V, \tag{3.19}$$

$$\implies \mathrm{QSP}(\theta; \vec{\phi}_{0:d+1}) = \mathrm{QSP}(\theta; \vec{\psi}_{0:d+1}), \tag{3.20}$$

$$\implies \mathrm{QSP}(\theta; \vec{\phi}_{0:d+1}) e^{-i(\phi_{d+1} - \frac{\pi}{2})Z} W e^{-i\frac{\pi}{2} Z} = \mathrm{QSP}(\theta; \vec{\psi}_{0:d+1}) e^{-i(\phi_{d+1} - \frac{\pi}{2})Z} W e^{-i\frac{\pi}{2} Z}, \tag{3.21}$$

$$\implies \mathrm{QSP}(\theta; \vec{\phi}_{0:d}) = \mathrm{QSP}(\theta; \{\psi_0, \ldots, \psi_d, \psi_{d+1} - \phi_{d+1} + \pi/2, -\pi/2\}). \tag{3.22}$$

On the left-hand side of the final equation is a QSP unitary of degree-d; by the inductive hypothesis, it must be the case that the QSP unitary on the right-hand side, which is of length-$(d+2)$, is also of degree-d. This is only possible if we can perform elision at the next-to-last position. By Lemma 19 this requires $\psi_{d+1} - \phi_{d+1} + \frac{\pi}{2} = \pi(n_{d+1} + \frac{1}{2})$ for some $n_{d+1} \in \mathbb{Z}$ which implies $\psi_{d+1} - \phi_{d+1} = n_{d+1}\pi$. Furthermore, we find $\chi \in \{0, \pi\}$ again by noting that $e^{i\pi Z} = -I$. Thus proving the inductive step $\phi_{d+1} - \psi_{d+1} = 2\pi m_{d+1}$ and by extension, the Lemma. □

We can summarize the results of this section using a concise diagrammatic notation. An example of such a plot is given in Section 3.3 and has several notable features:

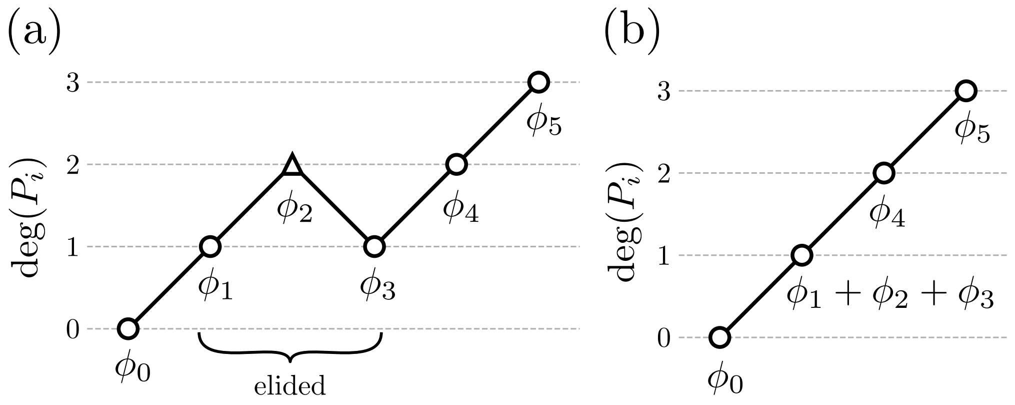


Figure 3.2: Visualization of (a) a length-5 QSP sequence parameterized by $(\phi_0, \cdots, \phi_5)$ and (b) its length-3 elided form by the result of Lemma 19. We will refer to such plots in general as QSP degree plots, often omitting the vertical axis labels to improve legibility.

- Arbitrary Z-rotations are represented by open circles, and we use open triangles to plot QSP phases that are a half-integer multiple of π to distinguish degree-peaks.
- Markers with solid fill are used to indicate additional rotations by $\pi/2$. Markers with checkerboard fill are used to indicate ϵ-noisy rotations.
- Signal operators are represented by solid lines.
- The vertical axis is used to plot the degree of the polynomial $P_{U^{(i)}}$ at position i;
- By Lemma 17, each layer of signal processing either increases or decreases the degree of the polynomial $P_i(a) = \langle 0| U(\theta, \vec{\phi}_{0:i}) |0\rangle$ by exactly one.
- Finally, Lemma 19 provides us with a way of simplifying QSP diagrams with degree-peaks through elision. This is depicted in Section 3.3.

A more involved application of elision can be found in Fig. 3.3, where the diagrammatic notation is used to represent a Kraus operator (Eq. (3.10)) of the error channel of

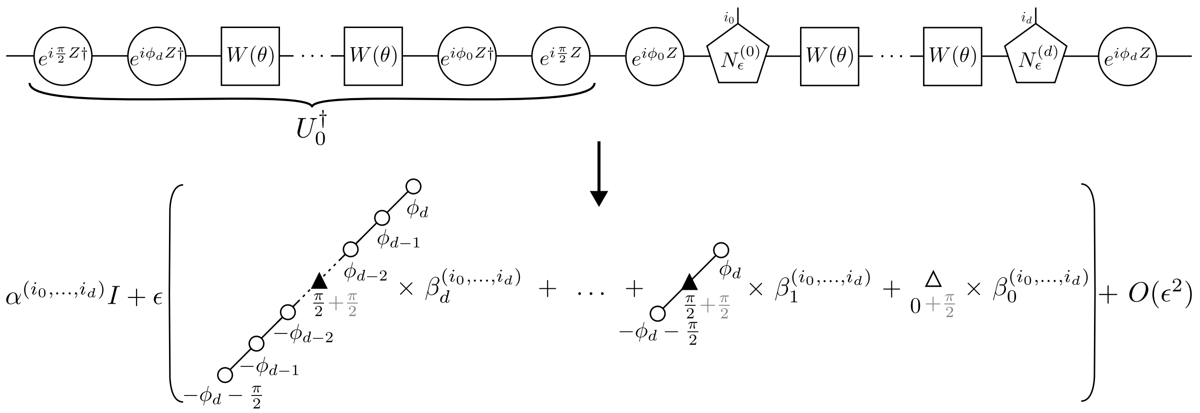


Figure 3.3: QSP degree diagram decomposition of generic error channel Kraus operator for a length-d QSP parameterized by phases $\phi_0, \ldots, \phi_d \in \mathbb{R}$ in our noise model for $\alpha \in \mathbb{R}$ and all $\beta_j \in \mathbb{R}$.

Section 3.2.3.

3.4 A model of coherent error

We consider a model of coherent errors to demonstrate the utility of the notation for reasoning about error correction.

In this error model, we assume that the signal processing operators under- or over-rotate by a fixed multiplicative factor ϵ: $\phi \mapsto \phi(1+\epsilon)$ for all ϕ. While ϵ is unknown a priori, we assume that it is constant throughout the application of the sequence and that it is small, $\epsilon \ll 1$, so that we may expand errors in orders of ϵ. In this case, the error can be characterized by a single Kraus operator

$$N_\epsilon^{(j,1)} = e^{i\epsilon\phi_j Z}, \tag{3.23}$$

at each site $0 \leq j \leq d$. Such an error may be due to imperfections on the hardware control and is akin to models systematic errors which are mitigated using composite pulses [BHC04; LYC14; Lev86].

Using the notation developed in Section 3.3, we first perform a perturbative analysis of both the error channel under this model (Section 3.4.1) and the error channel of possible recovery operations (Section 3.4.2). Next, we show that the most general form of recovery is impossible without additional resources (Section 3.4.3). Working around this constraint, we show that a weaker form of recovery is possible and provide an explicit construction along with an upperbound on the length of the recovered operator (Section 3.4.4). Finally, allowing an additional assumption, we argue a lowerbound on the length of recovered operator which is tight for first-order recovery (Section 3.4.6).

This Section complements our companion paper [Tan+23a], providing the full derivation of stated results using the notation introduced in Section 3.3. The numbering of Theorems in this Section is consistent with that in [Tan+23a]: Theorems 3 to 6 correspond to Theorems 1 to 4 in [Tan+23a].

3.4.1 A perturbative analysis of the error channel

For this simple model of coherent errors, the QSP error channel can be characterized by a single unitary Kraus operator which we call its *error operator* $E_\epsilon \equiv U_0^\dagger U_\epsilon$ (we call the canonical profile of E_ϵ the error profile).

We now perform a perturbative analysis of the error operator under this noise model. Expanding a noisy Z-rotation in orders of ϵ,

$$e^{i\phi(1+\epsilon)Z} = e^{i\phi Z} \sum_{k=0}^{\infty} \epsilon^k \phi^k e^{i\frac{\pi k}{2}Z}. \tag{3.24}$$

Substituting into Eq. (3.1), we obtain

$$U_\epsilon(\theta;\vec{\phi}) = \mathrm{QSP}_\epsilon(\theta;\vec{\phi}) \equiv \left(e^{i\phi_0 Z}\sum_{k_0=0}^{\infty}\epsilon^{k_0}\phi^{k_0}e^{i\frac{\pi}{2}k_0 Z}\right)\prod_{j=1}^{d}\left[W(\theta)\left(e^{i\phi_j Z}\sum_{k_j=0}^{\infty}\epsilon^{k_j}\phi^{k_j}e^{i\frac{\pi}{2}k_j Z}\right)\right]. \tag{3.25}$$

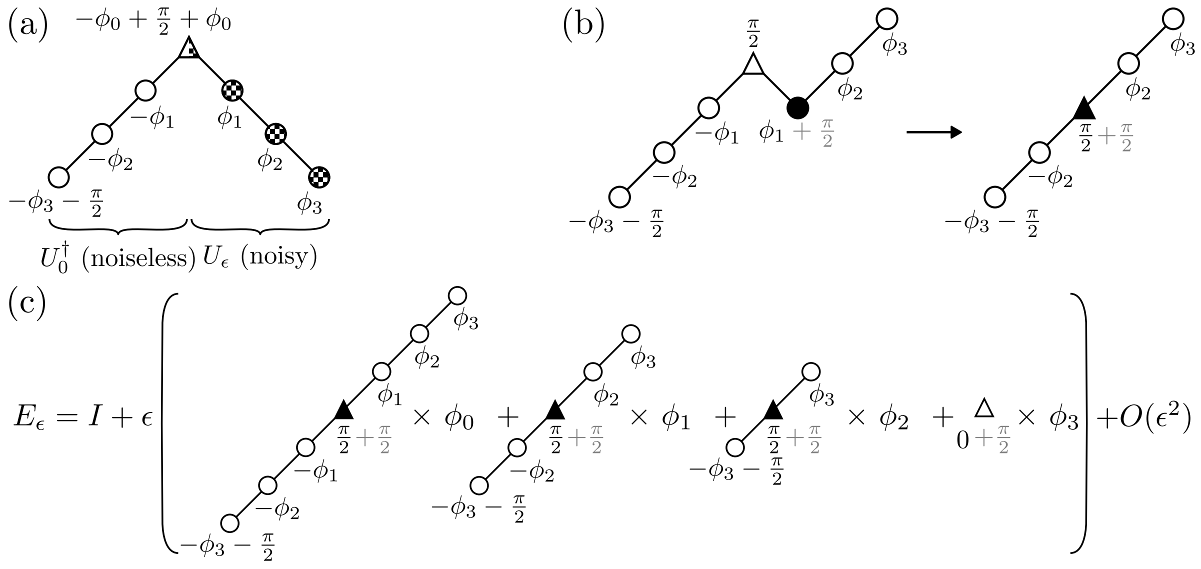


Figure 3.4: Diagrammatic representation of an error operator for a length-3 QSP parameterized by $\phi_i \in \mathbb{R}$. (a) Error operator $E_\epsilon = U_0^\dagger U_\epsilon$. Checkerboard fill indicates ϵ-noisy rotations (note the peak phase is only partially noisy). (b) Analysis of one term in the first-order perturbative expansion of the error operator corresponding to an over-rotation error of the ϕ_1 phase and elided form. The location of $\frac{\pi}{2}$ over-rotation errors is marked by filled markers. (c) Expansion of the error operator showing all diagrams to first-order with corresponding weights.

Rewriting in orders of ϵ,

$$\begin{aligned} U_\epsilon(\theta; \vec{\phi}) = U_0 + \epsilon \big(&\phi_0 e^{i(\phi_0+\frac{\pi}{2})Z} W e^{i\phi_1 Z} W \dots W e^{i\phi_d Z} \\ &+\phi_1 e^{i\phi_0 Z} W e^{i(\phi_1+\frac{\pi}{2})Z} W \dots W e^{i\phi_d Z} \\ &+ \dots \\ &+\phi_d e^{i\phi_0 Z} W e^{i\phi_1 Z} W \dots W e^{i(\phi_d+\frac{\pi}{2})Z}\big) \\ &+ O(\epsilon^2). \end{aligned} \tag{3.26}$$

The first-order term is a sum of $d+1$ QSP unitaries, each a copy of the noiseless QSP with a $\pi/2$ over-rotation at location j weighted by ϕ_j for each index j. Likewise, the k^{th}-order term is a sum of $(d+1)^k$ QSP unitaries corresponding to all possible ways to to insert k over-rotations by $\frac{\pi}{2}$ (including multiple over-rotations at the same index).

To obtain the error operator E_ϵ, we left-multiply by $U_0^\dagger$ which can be written as a noiseless

QSP per the construction of Corollary 20. Each of these sequences can be simplified using repeated application of Lemma 19. To first-order, the result is

$$\begin{aligned}
E_\epsilon = I + \epsilon\,(&\phi_0 \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, -\phi_{d-1}, \ldots, -\phi_2, -\phi_1, \pi, \phi_1, \phi_2, \ldots, \phi_{d-1}, \phi_d)) \\
&+\phi_1 \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, -\phi_{d-1}, \ldots, -\phi_2, \pi, \phi_2, \ldots, \phi_{d-1}, \phi_d)) \\
&+\ldots \qquad\qquad (3.27)\\
&+\phi_{d-1} \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, \pi, \phi_d)) \\
&+\phi_d \times \mathrm{QSP}(\theta; (\pi/2))) + O(\epsilon^2).
\end{aligned}$$

Note that the first-order expansion in Eq. (3.27) consists of a weighted sum of even length QSP unitaries of a special form, which we generalize in Definition 15. Analogous calculations show that, the higher-order terms in the expansion are likewise weighted sums over QSP unitaries of even length. Therefore by Corollary 41, the error operator E_ϵ admits a canonical expansion.

An example in diagrammatic form is provided for a general length-3 QSP in Fig. 3.4.

3.4.2 A perturbative analysis of recovery operators

Given a noisy QSP U_ϵ, we seek a recovery operator R_ϵ, itself a noisy QSP operator, such that their product is $U_\epsilon R_\epsilon$ is 'less noisy' in a sense that will be defined precisely in Section 3.4.4. Since our recovery operation should leave the state unchanged (up to a global phase) as $\epsilon \to 0$, we define the natural class of degree-0 operators and perform a perturbative analysis using the diagrammatic notation of Section 3.3.

Definition 22 (Degree-0 operator)**.** We call a QSP unitary U_ϵ *degree-*0 *to order* $k \geq 1$ if it can be written

$$U_\epsilon = e^{i(z_0 + z_1\epsilon + O(\epsilon^2))Z + i\epsilon^k[(x+O(\epsilon))X + (y+O(\epsilon))Y]}, \qquad (3.28)$$

for some real x, y, z_1 independent of ϵ but possibly functions of θ, and $z_0 \in \mathbb{R}$. Additionally, we call any QSP operator satisfying Eq. (3.28) for some $k \geq 1$ *degree-*0. Equivalently, a QSP

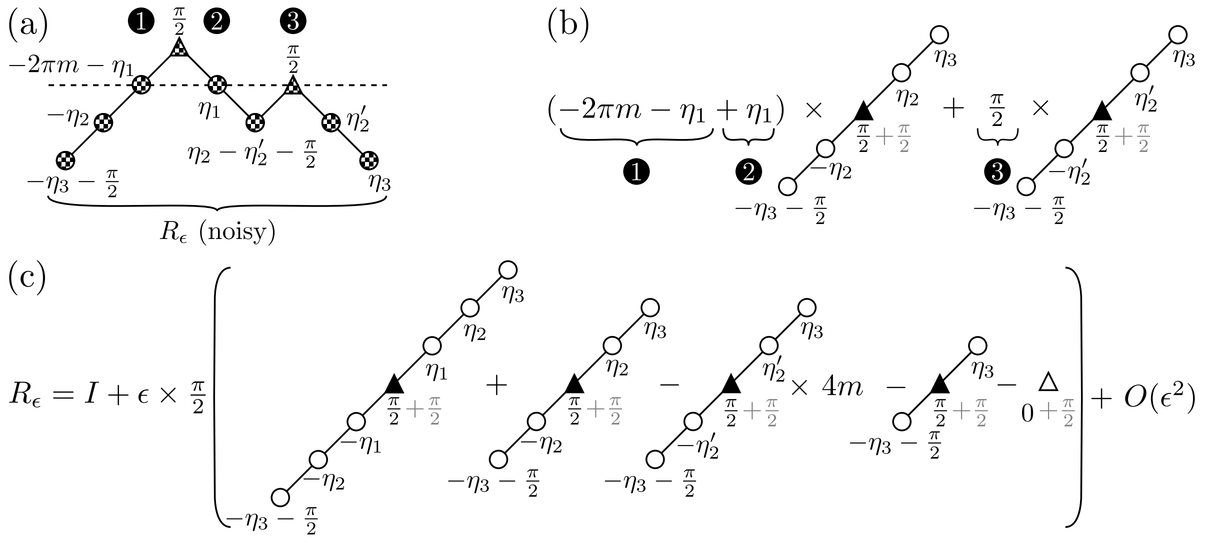


Figure 3.5: Diagrammatic analysis of an irreducible degree-0 QSP parameterized by $\chi = 0$, $\eta_i \in \mathbb{R}$ and $m \in \mathbb{Z}$ (compare with Fig. 3.4). (a) Diagrammatic representation of an irreducible degree-0 QSP R_ϵ. A detailed analysis is performed for over-rotation errors occurring at select locations (labels 1–3) corresponding to errors at degree-2 (dashed line). Checkerboard fill indicates ϵ-noisy rotations. (b) Analysis of diagrams resulting from over-rotation errors at locations labeled in previous sub-figure after elision. The location of $\frac{\pi}{2}$ over-rotation errors is marked by filled markers. (c) Expansion of the recovery operator showing all diagrams to first-order with corresponding weights. Notice that weights are integer multiples of $\pi/2$.

operator U_ϵ is degree-0 if $U_0 = e^{iz_0 Z}$ for some $z_0 \in \mathbb{R}$.

Definition 23 (Unbiased operator)**.** We call a QSP unitary U_ϵ *unbiased to order* $k \geq 1$ if it is degree-0 and $U_0 = I$.

To this end, we study the properties of degree-0 QSP operators, and in particular, the properties of their irreducible building blocks:

Definition 24. A degree-0 QSP unitary of length d is called *irreducible* if $\deg(P^{(i)}) > 0$ for all $0 < i < d$; otherwise a degree-0 QSP is called *reducible.*

We start with the generic form of a degree-0 length-2 QSP unitary. Due to the Lemma 17, all degree-0 QSP unitaries of length-2 are irreducible. Further, the following is a consequence of Lemma 19:

Corollary 25. A length-2 sequence QSP$(\theta; (\phi_0, \phi_1, \phi_2))$ is degree-0 if and only if

$$\phi_0 = \chi + \left(\phi + \pi\left(2m + n + \frac{1}{2}\right)\right), \tag{3.29}$$

$$\phi_1 = \pi\left(n + \frac{1}{2}\right), \tag{3.30}$$

$$\phi_2 = \phi, \tag{3.31}$$

for some $\phi, \chi \in \mathbb{R}$ and $n, m \in \mathbb{Z}$.

We can extend degree-0 QSP operations through the following operation:

Definition 26 (The conjugation super-operator)**.** Given $\eta \in \mathbb{R}$ and $m, n \in \mathbb{Z}$, we use $\mathcal{C}_{m,n,\eta}$ to denote the super-operator that maps a length-d sequence QSP$(\theta; \vec{\phi})$ to

$$\mathcal{C}_{m,n,\eta}\text{QSP}(\theta; \vec{\phi}) \equiv e^{-i(\eta+\pi(2m+n+\frac{1}{2}))Z} W e^{i\pi(n+\frac{1}{2})Z} \text{QSP}(\theta; \vec{\phi}) W e^{i\eta Z}, \tag{3.32}$$

which is a length-$(d+2)$ QSP sequence with phase angles $-(\eta + \pi(2m + n + \frac{1}{2})), \pi(n + \frac{1}{2}) + \phi_0, \phi_1, \ldots, \phi_d$, and η.

Note that the irreducible length-2 degree-0 QSP of Corollary 25 can be written as

$$\text{QSP}(\theta; (\phi_0, \phi_1, \phi_2)) = e^{i\chi Z}\mathcal{C}_{m,n,\phi} I. \tag{3.33}$$

The conjugation $\mathcal{C}_{m,n,\eta}$ super-operator appears naturally in our analysis of the error operator and subsequent construction of the recovery sequence. The effect of conjugation on an operator's canonical profile is detailed in Remark 43.

The conjugation operation is unique in the following sense:

Lemma 27 (Decomposition of irreducible degree-0 QSP unitary)**.** An irreducible degree-0 QSP sequence R of length $d \geq 2$ parameterized by phases $\vec{\phi} \in \mathbb{R}^{d+1}$ can be written as $R = e^{i\chi Z}\mathcal{C}_{m,n,\phi_d} R'$ for some $\chi \in \mathbb{R}$ and $m, n \in \mathbb{Z}$, and unbiased QSP R' of length-$(d-2)$.

Proof. If $d = 2$, then $R = e^{i\chi Z}\mathcal{C}_{m,n,\phi_2}I$ for some $\chi \in \mathbb{R}$ and $m, n \in \mathbb{Z}$ by Corollary 25.

For $d > 2$, we proceed by repeated application of the QSP elision operation (Lemma 19), each time reducing the length of R by 2. In particular, since the unbiased QSP R is irreducible, it has a degree peak at location $2 \le i \le d-2$. Performing elision about position i, we are left with the length-$(d-2)$ irreducible degree-0 QSP sequence. Notably, neither phases ϕ_0 nor ϕ_d are affected by performing elision at location $2 \le i \le d-2$. After $(d/2-1)$ elision steps, we are left with a length-2 QSP parameterized by $\mathrm{QSP}(\theta; (\phi_0, \sum_{i=1}^{d-1} \phi_i, \phi_d))$, where by Lemma 19, we have that

$$\sum_{i=1}^{d-1} \phi_i = \pi\left(n + \frac{1}{2}\right), \tag{3.34}$$

for some $n \in \mathbb{Z}$.

Therefore $\mathrm{QSP}(\theta, \vec{\phi}_{1:d-1}) = e^{i\pi(n+\frac{1}{2})Z}$ or equivalently $e^{-i\pi(n+\frac{1}{2})Z}\mathrm{QSP}(\theta, \vec{\phi}_{1:d-1}) = I$. Thus we can rewrite the original QSP in the desired form $R = e^{i\chi Z}\mathcal{C}_{m,n,\phi_d}R'$ for $\chi \in \mathbb{R}$ and unbiased length-$(d-2)$ QSP $R' \equiv \mathrm{QSP}(\theta; (\phi_1 - \pi(n+\frac{1}{2}), \phi_2, \dots, \phi_{d-1}))$ proving the Lemma. $\square$

A degree-0 operator is a rotated version of its own error operator. Therefore R_ϵ can be written a form similar to that of Definition 15. We aim to show that for the case of degree-0 R_ϵ, these weights are additionally integer multiples of $\frac{\pi}{2}$ save for the degree-0 term.

First, consider the case of irreducible degree-0 QSP $R = \mathrm{QSP}(\theta; (\phi_0, \dots, \phi_d))$. Consider the contributions to the first-order error from over-rotations at the first and last positions (i.e. assume for now errors do not affect positions $0 < i < d$). By Lemma 27 we can write irreducible

$$R = e^{i\chi Z}\mathcal{C}_{m,n,\phi_d}R' = e^{i\phi_0 Z}We^{i\pi(n+\frac{1}{2})}R'We^{i\phi_d Z}, \tag{3.35}$$

for R' unbiased $\chi \in \mathbb{R}$ and $m, n \in \mathbb{Z}$. Over-rotation at the first and last positions occur at degree-0 and therefore both produce an degree-0 error term equivalent to $e^{i(\chi+\frac{\pi}{2})Z}$ and the overall weight of the degree-0 diagram is $\phi_0 + \phi_d = \chi - \frac{\pi}{2}(4m + 2n + 1)$. The same analysis holds for the unbiased R', however $\phi_i + \phi_{d-i}$ must be an integer multiple of $\pi/2$ for $0 < i < d$

by Lemma 19, and therefore the error diagram must have weight that is a integer multiple of $\pi/2$. Furthermore, the weights are preserved by the linearity of error profile to first-order under conjugation and product. Thus, the same holds for higher degree error diagrams.

Now consider a general degree-0 QSP R decomposed into r constituent irreducible components,

$$R = e^{i\chi_1 Z} J^{(1)} e^{i\chi_2 Z} J^{(2)} \dots e^{i\chi_r Z} J^{(r)}, \tag{3.36}$$

where $\chi_1, \dots, \chi_r \in \mathbb{R}$, and each $J^{(j)}$ is an irreducible unbiased QSP. We can in general write an degree-0 QSP R_ϵ up to first-order in ϵ as follows:

$$R_\epsilon = e^{i\chi Z}\left[I + \left(\sum_i c_i e^{i\frac{\pi}{2}Z} + \frac{\pi}{2}\sum_i d_i U_i \right) + O(\epsilon^2) \right], \tag{3.37}$$

for $\chi = \chi_1 + \dots + \chi_r$, $c_i \in \mathbb{R}$, $d_i \in \mathbb{Z}$ and QSP unitaries U_i of even length. Additionally, each U_i is of the form

$$U_i = \mathcal{C}_{m_{i,d}, n_{i,d}, \eta_{i,d}} \dots \mathcal{C}_{m_{i,1}, n_{i,1}, \eta_{i,1}} e^{i\frac{\pi}{2}}. \tag{3.38}$$

A diagrammatic analysis is provided for an example length-8 irreducible degree-0 QSP in Fig. 3.5.

3.4.3 Z-error is not correctable in general

A natural question to ask is how one should define recovery and if it is possible, given access only to such noisy signal processing rotations. First, we show the impossibility of the most general form of error correction:

Theorem 3 (No correction of Z-error). *Let U_ϵ be a length-d noisy QSP unitary parameterized by $(\phi_0, \dots, \phi_d) \in \mathbb{R}^{d+1}$. For general phases ϕ_i, no noisy QSP unitary U'_ϵ exists such that for any $k \geq 1$, for all states $|\psi\rangle$,*

$$|\langle\psi|U'_\epsilon|\psi\rangle|^2 = |\langle\psi|U_0|\psi\rangle|^2 + O(\epsilon^{k+1}). \tag{3.39}$$

The condition given by Eq. (3.39) of Theorem 3 is equivalent to requiring

$$U'_\epsilon = U_0 e^{i\chi} e^{i\epsilon^{k+1}(xX+yY+zZ+O(\epsilon))}, \tag{3.40}$$

for some global phase $\chi \in \mathbb{R}$ and x, y, and z functions of θ.

We continue with a few results needed in our proof of the impossibility result.

Lemma 28 (Bottom-degree term of degree-0 QSP)**.** Let U_ϵ be a degree-0 QSP with $U_0 = e^{i\chi Z}$ for $\chi \in \mathbb{R}$ (as required by Eq. (3.40)). Then the bottom-degree Z term in its error profile $\mathcal{P}$ to first-order in ϵ is

$$\mathcal{P}_{-1}^{(z,1)} = (\chi + m\pi)\cos\chi, \tag{3.41}$$

for some $m \in \mathbb{Z}$.

Proof. The expansion of a general degree-0 QSP U to first-order is given by Eq. (3.37). Remark 43 gives us the lowest degree Z coefficients in the canonical expansion of each constituent diagram: the contribution to the lowest degree Z term is given by the bottom-left component of a product of the B matrices of Eq. (B.9), which for all degree ≥ 2 diagrams is 1 and for the degree-0 diagram $e^{i\frac{\pi}{2}Z}$ is -1. After a careful accounting of the weights, we find that the sum from all diagrams to this lowest degree term is $\chi + m\pi$ for $m \in \mathbb{Z}$. Finally, the overall $e^{i\chi Z}$ rotation of the first-order term in Eq. (3.37) results in an overall multiplicative factor of $\cos\chi$ on the expansion by Remark 42. Together, this results in $\mathcal{R}_{-1}^{(z,1)}$ of the form required by the Lemma. □

We are now ready to prove Theorem 3.

Proof. First we show that we cannot fully recover a noisy length-0 QSP $U_\epsilon \equiv e^{i\phi_0(1+\epsilon)Z}$ to first-order for general $\phi_0 \in \mathbb{R}$. For full recovery, the QSP must satisfy Eq. (3.40), and therefore either $U'_0 = e^{i\phi_0 Z}$ or $U'_0 = e^{i(\phi_0+\pi)Z} = -e^{i\phi_0 Z}$; therefore it U' must be a degree-0 QSP. We see immediately that its bottom-degree term, given by Lemma 28, cannot be corrected in general (i.e. unless ϕ_0 is an integer multiple of $\pi/2$).

Now we generalize the result for QSPs of length $d > 0$. For contradiction, suppose that there exists an error correction function $\mathrm{EC} : \mathbb{R}^{d+1} \to \mathbb{R}^{d'+1}$ for some $d' > d$ that is capable of mapping an arbitrary length-d QSP sequence to one that is corrected to first-order. That is, suppose for any $\vec{\phi} \in \mathbb{R}^{d+1}$ parameterizing QSP $U_\epsilon = \mathrm{QSP}_\epsilon(\theta, \vec{\phi})$ we have $\vec{\psi} = \mathrm{EC}(\vec{\phi})$ and $U'_\epsilon = \mathrm{QSP}_\epsilon(\theta, \vec{\psi})$ satisfying Eq. (3.40). We can simulate a length-0 QSP operator $\mathrm{QSP}(\theta, (\phi_0))$ by appending a recovered length-d QSP and a recovered version of its inverse (Corollary 20). For concreteness, we can choose phases $\vec{\phi}_1, \vec{\phi}_2 \in \mathbb{R}^{d+1}$,

$$\vec{\phi}_1 = (-\pi/2, 0, \dots, 0, \pi/2), \tag{3.42}$$

$$\vec{\phi}_2 = (0, \dots, 0, \phi_0). \tag{3.43}$$

Let $R_\epsilon = \mathrm{QSP}_\epsilon(\theta, \vec{\phi}_1)$ and $S_\epsilon = \mathrm{QSP}_\epsilon(\theta, \vec{\phi}_2)$. Further let $\vec{\psi}_1 = \mathrm{EC}(\vec{\phi}_1)$, $\vec{\psi}_2 = \mathrm{EC}(\vec{\phi}_2)$, and $R'_\epsilon = \mathrm{QSP}_\epsilon(\theta, \vec{\psi}_1)$ and $S'_\epsilon = \mathrm{QSP}_\epsilon(\theta, \vec{\psi}_2)$. Note that by construction $R_0 S_0 = e^{i\phi_0 Z}$ as desired and therefore $R'_0 S'_0 = e^{i\phi_0 Z}$. Further, if both R'_ϵ and S'_ϵ satisfy Eq. (3.40) for $k \geq 1$, then resulting length-$2d'$ QSP $R'_\epsilon S'_\epsilon$ will also be fully corrected to order k. This contradicts our original result for $d = 0$ and therefore EC cannot exist for any $d \geq 0$ thus proving the Theorem. □

3.4.4 First-order recovery

In light of the impossibility result presented in Section 3.4.3, we shift our attention to XY error recovery. We show that it is possible to perform this restricted form of recovery and make use of the tools developed in Section 3.4.2 to provide a general construction for XY recovery operators:

Theorem 4 (Recoverability)**.** *Given any noisy QSP operator $U_\epsilon(\theta)$ of length d and an integer $k \geq 1$, there exists a recovery sequence $R_\epsilon(\theta)$ satisfying*

$$|\langle 0|U_\epsilon R_\epsilon|0\rangle|^2 = |\langle 0|U_0|0\rangle|^2 + O(\epsilon^{k+1}), \tag{3.44}$$

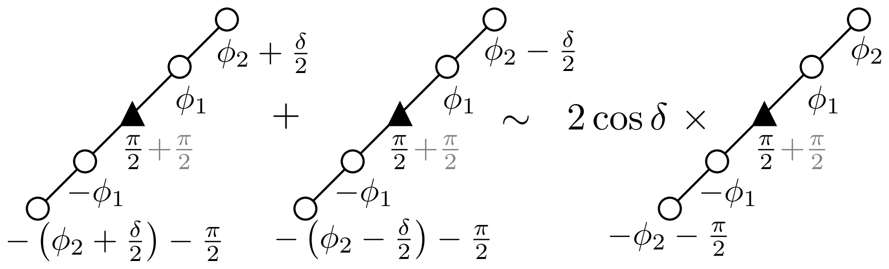

$\phi_2 + \frac{\delta}{2}$
ϕ_1
$\frac{\pi}{2} + \frac{\pi}{2}$
$-\phi_1$
$-\left(\phi_2 + \frac{\delta}{2}\right) - \frac{\pi}{2}$
$+$
$\phi_2 - \frac{\delta}{2}$
ϕ_1
$\frac{\pi}{2} + \frac{\pi}{2}$
$-\phi_1$
$-\left(\phi_2 - \frac{\delta}{2}\right) - \frac{\pi}{2}$
$\sim \; 2\cos\delta \; \times$
ϕ_2
ϕ_1
$\frac{\pi}{2} + \frac{\pi}{2}$
$-\phi_1$
$-\phi_2 - \frac{\pi}{2}$

Figure 3.6: Diagrammatic representations of the XY-equivalence of counter-rotated diagrams.

for all θ.

The condition given by Eq. (3.44) of Theorem 4 is equivalent to requiring

$$U'_\epsilon = U_0 e^{i(\chi+O(\epsilon))Z+\epsilon^{k+1}((x+O(\epsilon))X+(y+O(\epsilon))Y)}, \tag{3.45}$$

for some $\chi \in \mathbb{R}$, and x, y, and z functions of θ.

We make the following definition in light of Eq. (3.45):

Definition 29 (XY-equivalence)**.** We say that two operators $U = w(\theta)I + i[x(\theta)X + y(\theta)Y + z(\theta)Z]$ and $V = w'(\theta)I + i[x'(\theta)X + y'(\theta)Y + z'(\theta)Z]$ are *XY-equivalent* if $x(\theta) = x'(\theta)$ and $y(\theta) = y'(\theta)$. We denote this $U \sim V$.

We provide an explicit construction using unbiased recovery operators (i.e. $R_0 = I$). An upperbound on the length of the recovery operator R_ϵ will be a corollary of our construction:

Theorem 5 (Upper bound on recovery length)**.** *Given any noisy QSP operator $U_\epsilon(\theta)$ of length d with c distinct phases (up to factors of 2π) and an integer $k \geq 1$, there exists a recovery sequence $R_\epsilon(\theta)$ satisfying*

$$|\langle 0|U_\epsilon R_\epsilon|0\rangle|^2 = |\langle 0|U_0|0\rangle|^2 + O(\epsilon^{k+1}), \tag{3.46}$$

for all θ. Furthermore, there exists a QSP operator satisfying the above with length at most $O(2^k c^{k(k+1)/2} d)$.

To show Theorem 4, we provide an explicit construction for R_ϵ.

The irreducible components of our recovery operator will be length-$2r$ recovery operators constructed by conjugating the identity operator. We make use of an integral degree of freedom, namely the freedom to over-rotate by factors of 2π,

$$\begin{aligned}
&\mathrm{QSP}_\epsilon(\theta; (-\phi_d - \pi/2, -\phi_{d-1}, \ldots, -\phi_{d-r+1}, \pi/2, \phi_{d-r+1} + 2\pi m_{d-r+1}, \ldots, \phi_d + 2\pi m_d)) = \\
&\qquad I + \epsilon \Big(-\frac{n\pi}{2} \times \mathrm{QSP}(\theta; (\pi/2)) \\
&\qquad\quad + 2\pi m_d \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, \pi, \phi_d))) \\
&\qquad\quad + 2\pi m_{d-1} \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, -\phi_{d-1}, \pi, \phi_{d-1}, \phi_d))) \\
&\qquad\quad\ \vdots \\
&\qquad\quad + \frac{n\pi}{2} \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, \ldots, -\phi_{d-r+1}, \pi, \phi_{d-r+1}, \ldots, \phi_d)) \Big) \\
&\qquad\quad + O(\epsilon^2).
\end{aligned} \tag{3.47}$$

for some $n \in \mathbb{Z}$ and $m_i \in \mathbb{Z}$ to be specified later.

There is a striking similarity between the error operator expansion in Eq. (3.27) and the recovery component of Eq. (3.47) which are visualized in Fig. 3.4 and Fig. 3.5 respectively. We take advantage of this fact to construct our recovery operator.

To be concrete, consider a noisy QSP $U_\epsilon = \mathrm{QSP}(\theta; (\phi_0, \ldots, \phi_d))$. Its error operator E_ϵ to first-order can be decomposed into a sum of even-length QSPs from length $0, 2, \ldots, 2d$ (Eq. (3.27)). The length-$2r$ diagram is in general is

$$\phi_{d-r} \times \mathrm{QSP}(\theta; (-\phi_d - \pi/2, \ldots, -\phi_{d-r+1}, \pi, \phi_{d-r+1}, \ldots, \phi_d)). \tag{3.48}$$

The length-$2r$ recovery operator of the form in Eq. (3.47) can be chosen to match this by setting all $m_i = 0$. Since canonical profiles add to leading-order Remark 46, we can simply add an additional $\pi/2$ shift added to the final phase $\phi_d \mapsto \phi_d + \pi$ to negate the X and Y

components of the recovery operator at first-order: this can be verified using Remark 42.

The only remaining challenge is to match the ϕ_{d-r} weight of the degree-$2r$ term in Eq. (3.48). Here we make use of the following trigonometric identity

$$\begin{pmatrix} \sin(\eta+\delta) \\ -\cos(\eta+\delta) \end{pmatrix} + \begin{pmatrix} \sin(\eta-\delta) \\ -\cos(\eta-\delta) \end{pmatrix} = 2\cos(\delta) \begin{pmatrix} \sin(\eta) \\ -\cos(\eta) \end{pmatrix}. \tag{3.49}$$

By duplicating the length-$2r$ sequence in Eq. (3.47) and counter-rotating each copy by an amount $\delta/2$, we can construct a sequence that is XY-equivalent to a re-scaled version of the original (shown diagrammatically in Fig. 3.6). To fully cancel the length-$2r$ diagram in Eq. (3.48), we append two length-$2r$ recovery QSPs

$$\mathcal{C}_{0,n,\phi_d+\pi/2\pm\delta}\mathcal{C}_{0,n,\phi_{d-1}}\ldots\mathcal{C}_{0,n,\phi_{d-r+1}}I, \tag{3.50}$$

choosing $n \in \mathbb{Z}$ such that there is a solution to $\delta = \frac{1}{2}\cos^{-1}\left(\frac{\phi_{d-r}}{n\pi}\right)$. This can be verified using Remark 42 and is represented diagrammatically in Fig. 3.6).

In summary, we have canceled the degree-$2r$ diagram using a length-$4r$ QSP operator. We repeat this for each diagram of length-$2r$ in the first-order expansion of the error operator for $r \in \{2, 4, \ldots, 2d\}$; the length-0 term contributes only to the Z component of the error, which can be ignored. Overall, the recovery of each diagram takes a length $\Theta(r)$ QSP and we need to correct $\Theta(d)$ diagrams, resulting in a final recovery operator of length $\Theta(d^2)$.

For a generic length-d QSP with all phases distinct, we cannot do better using this method, as we still need to perform counter-rotation d times. But for a length-d QSP operator with c distinct phases, we can group diagrams that are scaled by the same amount; each group can be corrected using a single length-$2r$ diagram by appropriately choosing m_i and n. Overall, if there are c distinct phases, there will be c distinct groups, each requiring a separate counter-rotated diagram of length $\Theta(d)$. Note that we consider phases to be equivalent if they differ by an integer multiple of 2π as these can be matched within the same group by an appropriate choice of m_i and n. Thus the overall complexity using this scheme yields the improved $\Theta(cd)$ for QSP diagrams with high phase degeneracy. As an example, the first-order recovery phases for the special case of a QSP with one unique phase are provided in Remark 53. This shows Theorems 4 and 5 for $k = 1$.

3.4.5 Sketch of recovery procedure

We now provide a high-level summary of the first-order recovery construction of the preceding sections and sketch out the proof of Theorems 4 and 5 for $k > 1$.

Expanding the error operator in orders of ϵ, we find that the contribution at each order can be written as a sum of QSP operators: the first-order components are of the form Definition 16. A similar expansion shows that irreducible degree-0 operators (Definition 24) can be expanded in a similar form except for the fact that the coefficients in a degree-0 operator's first-order expansion must be integer multiples of $\pi/2$ whereas the coefficients in the expansion of error operators are unconstrained. These expansions are shown diagrammatically for error operators in Fig. 3.4 and irreducible degree-0 operators in Fig. 3.5. A first-order recovery operator satisfying Eq. (3.45) can be constructed by concatenating irreducible degree-0 operators making use of the counter-rotation trick of Eq. (3.49) for continuous rescaling. One counter-rotation is required for each unique phase in the original QSP with each counter-rotated unit a QSP of length $\Theta(d)$. Generically this gives an $\Theta(d^2)$ first-order recovery procedure, but special cases, i.e. QSPs with high phase degeneracies can admit shorter recovery operators. A QSP with c unique phases can be recovered to first-order with a recovery operator of length $\Theta(cd)$. Grover's algorithm is a notable example of a QSP with high phase degeneracy (see Remark 53).

Subsequent recovery occurs order-by-order, making use of the additive property of leading-order terms (Remark 46). The higher-order expansions of both error and recovery operators can be written in terms of the generalized error components of Definition 49 (up to XY-equivalence). Higher-order recovery units are defined in Remark 51 in analogy with the

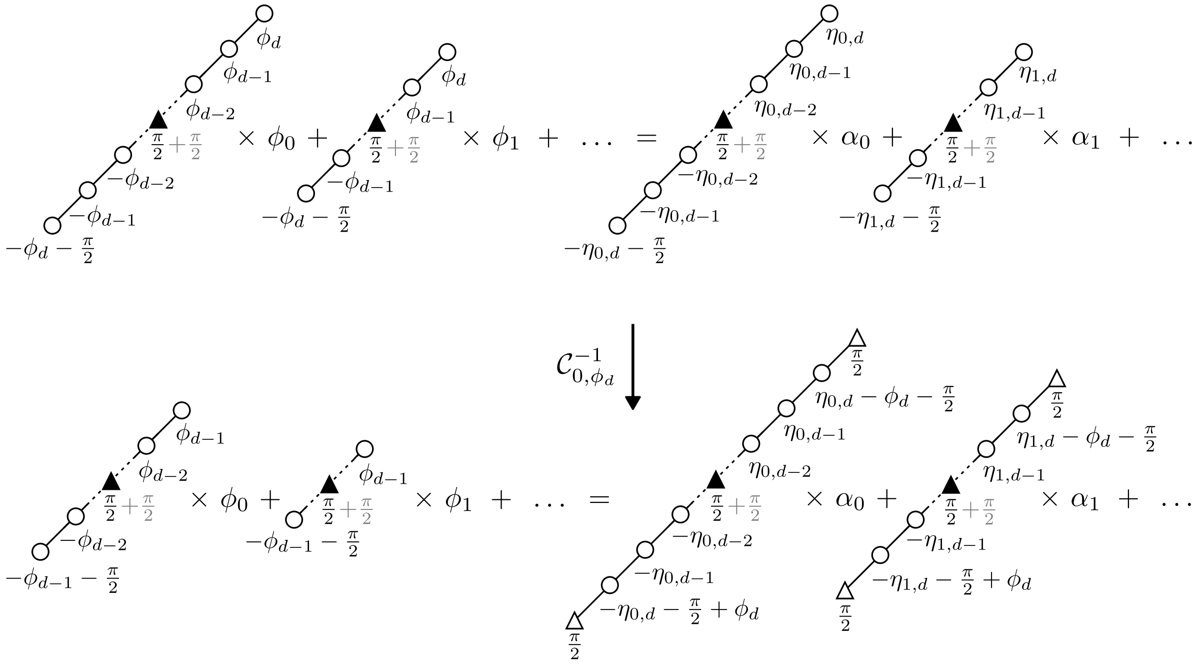


Figure 3.7: Diagrammatic representation of one anti-conjugation by step in the proof of Theorem 6 for a length-d QSP. The left-hand side depicts the first-order error terms and the right-hand side depicts a proposed set of recovery diagrams. Only the highest two degree terms in each sequence are shown as lower-degree diagrams cannot interfere assuming sufficiently large d and $f(a) = O(1)$. The anti-conjugation by $\mathcal{C}_{0,\phi_d}^{-1}$ decreases the degree all terms of the error operator by 2 (save for degree-0 term which does not affect the analysis); in order for the right-hand side to match, it must be that $\eta_{i,d} = \phi_d$ for all recovery diagrams i.

irreducible unbiased operators of Eq. (3.47) used for first-order recovery. The fact that we use unbiased operators for recovery places additional constraints on the coefficients of the recover operator's expansion which may be overcome through repetition of recovery units and judicious application of the counter-rotation trick. The key bottleneck in the required length of the recovery sequence is again the number of required counter-rotations, which ultimately yields the result of Theorem 5. A more detailed analysis of the higher-order recovery construction is left to Appendix B.2.

3.4.6 Lower bound

We now show that, given an additional assumption, the length of our recovery sequence for first-order recovery is asymptotically optimal.

Theorem 6 (Lower bound on recovery length)**.** *There exists a length-d QSP sequence U_ϵ such that for any XY recovery QSP R_ϵ of order $k \geq 1$ satisfying*

$$U_0^\dagger U_\epsilon R_\epsilon = I + \epsilon f(a) e^{i\frac{\pi}{2}Z} + O(\epsilon^2), \tag{3.51}$$

for function $f(a) = O(a^0)$, R_ϵ has length $\Omega(d^2)$.

The assumption on the first-order Z component in Theorem 6 (i.e. $f(a) = O(a^0)$) is required for technical reasons, but can also be seen as a desire to limit the complexity of the recovery sequence. While we conjecture that this assumption can be removed, it is important to point out that the condition for XY recovery (Eq. (3.45)) does not itself place any limits on $f(a)$; and, in fact, neither the recovery construction of Section 3.4.4 nor the construction of Appendix B.3 presented satisfy this requirement, instead having $f(a) = \Omega(d)$.

First we introduce the inverse of the conjugation super-operator of Definition 26. We denote this operation $\mathcal{C}_{0,\eta}^{-1}$ such that $\mathcal{C}_{n,\eta}^{-1} \circ \mathcal{C}_{m,n,\eta} = \text{id}$ for all $\eta \in \mathbb{R}$ and $m, n \in \mathbb{Z}$. Additional details can be found in Remark 45.

Lemma 30 (Two error components cannot be combined, first-order)**.** Let U and V be first-order error components of degree-$2r$ (i.e. of the form Definition 49 with all $b_i = 0$), parameterized by $\phi_{d-r+1}, \ldots, \phi_d$ and $\psi_{d-r+1}, \ldots, \psi_d$ respectively. Their weighted sum, $\alpha U + \beta V$ for $\alpha, \beta \in \mathbb{R}$ can be written as scaled single error component if and only if $\psi_i - \phi_i = n_i \pi$ for $n_i \in \mathbb{Z}$ for all $d - r + 1 \leq i \leq d$.

Proof. The $\Longleftarrow$ direction follows directly from the fact that $e^{i\pi Z} = -I$. For the $\Longrightarrow$ direction, consider that error components are QSP operators and therefore must be unitary. Therefore, we must have for some $c \in \mathbb{R}$,

$$\begin{aligned}(\alpha U + \beta V)(\alpha U + \beta V)^\dagger &= (\alpha^2 + \beta^2)I + \alpha\beta(UV^\dagger + VU^\dagger), \\ &= (\alpha^2 + \beta^2)I - \alpha\beta(UV + VU), \\ &= cI,\end{aligned} \tag{3.52}$$

where we have used the fact that first-order error components are unitary as well as anti-Hermitian (i.e. $V^{-1} = V^\dagger = -V$).

Since U and V are of form Definition 49, their canonical expansions $\mathcal{P}$ and $\mathcal{P}'$ have $\mathcal{P}_j^{(0,1)} = \mathcal{P}'^{(0,1)}_j = 0$ for all j by Remark 43. Thus, in order for the final equality in Eq. (3.52) to hold, we must have $UV = VU = \pm I$ or equivalently $U^\dagger = -U = \pm V$. The result holds by application of Corollary 21. □

We are now ready to prove Theorem 6.

Proof. Let $U_\epsilon = \text{QSP}(\theta; (\phi_0, \ldots, \phi_d))$ be a noisy QSP of length $d > 1$ QSP with error operator E_ϵ and R_ϵ any recovery sequence satisfying Eq. (3.44) for $k \geq 1$.

From Eq. (3.37), we see that each error component scaled by an independent real-value requires a separate irreducible recovery sequence of length $\Theta(d)$. To prove the Theorem, we show that generically $\Omega(d)$ independently scaled error components are required. We argue that to approximate the first-order error operator of Eq. (3.27), we need as sequence of degree $2d, 2(d-1), 2(d-2), \ldots$ error components. In fact, given the restrictive condition of $f(a) = O(a^0)$, the only approximation is one that identical to the error operator up to the π degrees of freedom allowed by Corollary 21.

Assume that we have found an approximation to first-order for an error operator of degree-$(2d)$. We proceed inductively, for the first $\Theta(d)$ diagrams by anti-conjugating thereby reducing the error operator to one of degree-$(2(d-1))$, neglecting the lowest-degree terms. Consider the first-order error terms of both E_ϵ and R_ϵ written in the form Definition 15. Anti-conjugating the error $\mathcal{C}_{0,\phi_d}^{-1} E_\epsilon$, results in all contributing diagrams decreasing in order by two (save for the degree-0 diagram) as the outermost phases of each diagram can be elided (Lemma 19). Therefore anti-conjugating the recovery operator $\mathcal{C}_{0,\phi_d}^{-1} R_\epsilon$ must likewise result in a two degree reduction. One step of the procedure is depicted in Fig. 3.7. Since by assumption, the difference in the Z-component is $f(a) = O(a^0)$, it cannot interfere with the top two degree diagrams for sufficiently large d; and the two highest-degree diagrams in R

must have outermost phase ϕ_d and be scaled by ϕ_0 and ϕ_1 respectively as in E_ϵ. This can be seen by using Remark 45 and Lemma 30. We can iterate this procedure $\Theta(d)$ times, before $f(a) = O(a^0)$ becomes relevant, each time requiring outermost phase of ϕ_{d-r+1} with scaling by ϕ_{r-1}.

Thus the top $\Theta(d)$ degree diagrams in the recovery operator must be identical to that in the error operator. If all ϕ_i distinct, $\Theta(d)$ independently scaled error components are required, each of length $\Theta(d)$. Thus showing the lower bound of $\Omega(d^2)$ for general length-d QSPs. □

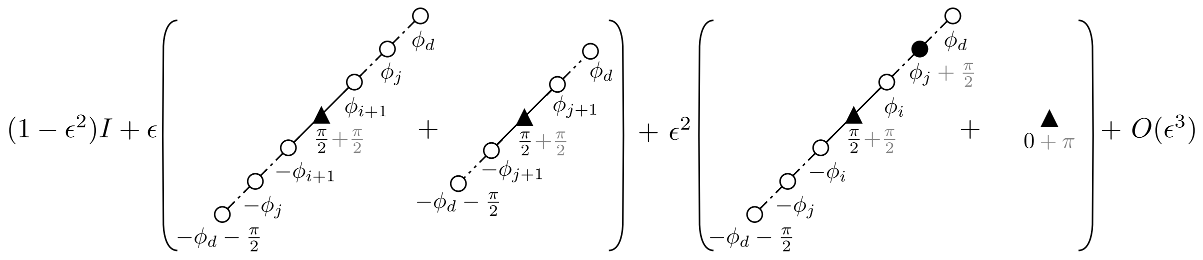
QSPs with phase degeneracies are able to circumvent this lower bound as in Theorem 5. This motivates the exploration of families of polynomials that can be generated (or approximated) by QSPs with $o(d)$ unique phases.

3.5 Concluding remarks

We have introduced a model of perturbative noise in the signal processing basis of QSP, and provided a set of diagrammatic tools useful for reasoning about such noise. The utility of these techniques are demonstrated by application to a model of coherent noise, that of a multiplicative under- or over- rotation, where we have developed and analyzed a novel method of ancilla-free recovery. We now discuss some directions for future work.

Strengthening the lower bound for the coherent error model.— Comparing Theorem 5 to Theorem 6 reveals that our construction is optimal for $k = 1$. However, our lower bound is independent of k and is therefore loose for $k > 1$ and presents a direction for future work. An important limitation of our current construction is the recursive construction of higher-order recovery unitaries (Remark 51) which requires a doubling in length for each order in k. It remains an open question whether an order-k unbiased sequence can be constructed using sub-exponential resources.

Closing these gaps between the upper and lower bounds has an important implication for



$(1-\epsilon^2)I + \epsilon$
ϕ_d
ϕ_j
ϕ_{i+1}
$\frac{\pi}{2}+\frac{\pi}{2}$
$-\phi_{i+1}$
$-\phi_j$
$-\phi_d - \frac{\pi}{2}$
$+$
ϕ_d
ϕ_{j+1}
$\frac{\pi}{2}+\frac{\pi}{2}$
$-\phi_{j+1}$
$-\phi_d - \frac{\pi}{2}$
$+\epsilon^2$
ϕ_d
$\phi_j + \frac{\pi}{2}$
ϕ_i
$\frac{\pi}{2}+\frac{\pi}{2}$
$-\phi_i$
$-\phi_j$
$-\phi_d - \frac{\pi}{2}$
$+$
$0+\pi$
$+O(\epsilon^3)$

Figure 3.8: Example Kraus operator for the error channel of a noisy QSP with phase damping at each site. The diagram corresponds to the Kraus operator with a phase flip at site i and j only with $0 \le i < j \le d$.

quantum computation, given that even the $O(d^2)$ scaling in our construction for first-order error correction would negate all quantum advantage (quadratic speedup) in most fixed-point quantum unstructured search [YLC14].

Incoherent errors.— The study of coherent errors has the benefit that one unitary Kraus operator is sufficient to describe both the noisy signal processing rotation and the QSP error channel; however, our formalism can also be applied to models of incoherent error. One significant limitation in the incoherent case is that the number of such Kraus operators grows exponentially with d.

In the case of incoherent noise, each Kraus operator of the error channel, rather than the error operator of Section 3.4.1, can be written in the form of Definition 15. As an example, consider the noisy signal processing rotation corresponding to the phase damping channel with two Kraus operators

$$N_\epsilon^{(j,1)} = \sqrt{1-\epsilon}\,I, \tag{3.53a}$$

$$N_\epsilon^{(j,2)} = \sqrt{\epsilon}Z, \tag{3.53b}$$

at each site $0 \le j \le d$. An example Kraus operator of the error channel is shown diagrammatically in Fig. 3.8.

While it is already technically challenging to construct recovery sequences given the simple coherent error model that we consider, it is absolutely crucial in the future to analyze the recovery sequence in the presence of an extensive source of random errors. These random errors typically introduce entropy into the quantum circuit and often arise in various quantum algorithms and physical devices.

Generalizing the diagrammatic notation.— To allow more complicated error sources, including errors in the signal basis, we anticipate further development of the diagrammatic perturbative expansion used in the present work as a formal tool to analyze error propagation in QSP. We hope such diagrammatic tools can serve as a complimentary picture to aid in future development of noisy QSP recovery strategies.

Combining with standard quantum error correction (QEC).— Whereas standard QEC techniques work by moving entropy into ancillary Hilbert spaces [Pre98; KL97; Alb+18], one can view our construction as rotating errors into the Z-component. The inability to correct Z-component of error proves a limitation of our method, as the Z-error can be important for situations when the QSP sequence need to be coherently concatenated with another quantum circuit [Mar+23]. However, our ancilla-free recovery technique can be concatenated with a standard QEC code to remove the remaining errors. For example, for incoherent errors, it may be possible to find a recovery channel that effectively standardizes the error channel e.g. transforming the error channel into a phase damping channel; this can then be concatenated with standard QEC codes tailored for phase damping errors. One can envision a combination of our ancilla-free recovery technique with standard QEC codes would provide a tunable trade-off between the required number of ancilla and gate depth.

Chapter 4

Resource savings from fault-tolerance

As previously discussed, there is a significant disparity of interest in classical versus quantum fault-tolerance results. One obvious reason for this is the difference in the currently achievable physical error rates in these cases. An alternate framing is the following: is is much cheaper to build a physically reliable classical gate than a physically reliable quantum gate. While the laws of physics do not preclude the possibility of building a nearly perfect quantum gate, those studying new ways to design fault-tolerant quantum systems are making an implicit calculation that it will be cheaper to build a fault-tolerant quantum computer than to make a sufficiently perfect quantum gate. Similar reasoning may be applied to fault-tolerant classical computation. While the balance appears clearly in the favor of non-fault-tolerant design in modern transistor-based logic, this may not be the case generically for novel lower-power designs [Sha+08; Sha+18] or alternate architectures [CV16; CV20b]. This line of research also has implications on fault-tolerance and reliability in emergent models of classical information processing, such as the observation of error correcting codes in mammalian brains [SF11].

Following this line of reasoning, we turn to the second motivating question of this thesis (Q2): the cost of fault-tolerance. We would like to understand not only when fault-tolerant design is possible but also when it is desirable. We propose the study of the resource cost of fault-tolerance as one potential route towards an answer—here we take a resource to be anything that is both desirable and scarce. In this Chapter[1], we provide a careful accounting of resource utilization in von Neumann's original fault-tolerant construction. We present a general framework to account for this overhead cost in order to effectively compare fault-tolerant to non-fault-tolerant approaches for computation, in the limit of small logical error rates. Using this detailed accounting, we determine explicit boundaries at which fault-tolerant designs become more efficient than designs that achieve comparable reliability through direct consumption of resources. In particular, we show that hardware-level fault-tolerance is always preferred when the resource cost grows faster than $\log(1/\epsilon)$ asymptotically for small ϵ.

4.1 Introduction

Motivation and related work— While fault-tolerance is an interesting theoretical exercise, one may wonder if it is of practical relevance: why bother with the additional complexity required to design a fault-tolerant circuit, when one can simply build a more perfect gate? It is true that for standard transistor-based logic gates have such low physical error rates [SPW09] that one is often better served using lightweight error detection methods at the software-level rather than implementing full hardware-level fault-tolerance. However, for low-power and nano-scale semiconductor devices, we are approaching a point where increased noise and statistical variations in manufacturing have lead some to call for new, more robust, computational paradigms [Sha+08; Sha+18]. Inspiringly, Chatterjee and Varshney [CV16; CV20b] applied the negative fault-tolerance results of [ES99] to place bounds on the energy-reliability trade-offs allowed for nano-scale circuits and deep feed-forward neural networks. Their approach provides insightful scaling results, but to make these ideas useful for practical computational systems, the theory of fault-tolerance bounds must become constructive.

[1]This has appeared as "Resource savings from fault-tolerant circuit design," by Andrew K. Tan and Isaac L. Chuang in arXiv preprint arXiv:2311.02132 (2023) [TC23].

Suppose that instead of perfect computation, the goal is to offer some specific level of reliability. This is increasingly a desirable systems engineering goal for computation, particularly when algorithmic outputs are probabilistic or the problem is inherently non-deterministic, e.g. as often is the case in machine learning. Fault-tolerance constructions can offer such an engineering trade-off: by increasing the size of the code, a computation can be performed by a noisy computer to arbitrary precision using polylogarithmic overhead in the number of gates [Neu56; Pip88; Fed89; EP98; ES99; NC10]. Importantly, fault-tolerance need not just be used to obtain a vanishingly small error rate; the desired error rate can be dialed in by changing the amount of redundancy employed. And moreover, the resource cost for fault-tolerance may come in many forms, not just the energy consumed, but also the space or time required. Thus, given the in-principle relatively modest, (poly)logarithmic, overhead required by fault-tolerant designs [Neu56; NC10], it seems natural to wonder if there are constructive approaches to show whether fault-tolerance may actually provide net savings for a broad class of resource–reliability trade-offs.

Thaker et al. [Tha+05; Tha+08] showed that in principle fault-tolerant constructions based on *recursive triple modular redundancy* may be more resource efficient than their non-fault-tolerant counterparts. Impens [Imp04] studied the same recursive construction as a way to trade resources for reliability, i.e. reliability as a fungible resource, and showed that fault-tolerant constructions may be more resource-efficient than their non-fault-tolerant counterparts if the computational primitive follows certain reliability–resource trade-offs. While similar to this work in spirit, both Thaker et al. and Impens use a recursive concatenation-based fault-tolerant design in which overhead scaling is polylogarithmic and constants are difficult to obtain owing to the fact that the size of the resulting fault-tolerant circuits grow exponentially with the level of recursion.

Our approach builds on this body of prior work, and goes further by employing a constructive fault-tolerant procedure which allows precise overhead estimates with no unbounded constants. Our construction allows the overhead to be separated into three main components: the first contribution is due to the code size requirement to attain the desired level of certainty, effectively a concentration bound; the second contribution is from the number of gates required to the error correcting circuit; and finally, a third contribution comes from the statistical dependence in a circuit's outputs, which depends on the details of the circuit's connectivity. We use a fault-tolerant construction based on fixed-depth error correction circuits in which overhead is logarithmic, thus simplifying the analysis compared with the recursive construction used by Thaker et al. and Impens. Our fixed-depth model allows us to numerically estimate constant factors required to determine the point at which fault-tolerant designs become more efficient than designs that achieve comparable reliability through direct consumption of resources.

Roadmap— The rest of the paper is organized as follows. First, we formalize the notion of a constant-depth fault-tolerant construction. Focusing on the simplest, depth-2 construction, we perform a careful analysis of the asymptotic overhead in the number of gates (Section 4.2). We find an overhead that is logarithmic in the overhead of desired reliability and numerically determine the constant factors. This is followed by the introduction of the asymptotic resource–reliability trade-off (Section 4.3). Here we find three qualitatively different cases and determine the regions of design space in which the fault-tolerant construction may be more resource efficient. Finally, we discuss potential avenues to further improve the fault-tolerant overhead and speculate on abstract models of computation where our results may provide insights (Section 4.4).

4.2 Fault-tolerance number overhead

First, we review a few basics of circuit-based fault-tolerance. In the most commonly studied scenario [Neu56], we are given a set of Boolean gates subject to some noise—these are the *basic units* of our computation. For our purposes, we will assume that a noisy gate behaves as an ideal gate save for a probability ϵ_{P} that its output is flipped; for simplicity, we assume additionally that all basic gates are subject to the same level of noise ϵ_P, and that the noise is independent. The goal is to build a *logical unit* using a number of basic gates which acts on encoded data such that its errs with error probability ϵ_L. Exactly how such a fault-tolerant logical unit may be constructed, and exactly what it means for it to make an error will be discussed later. What is important, however, is that in addition to the overhead introduced by building these logical units, these logical units only remain reliable up for $\epsilon_\mathrm{P} < \epsilon^*$ where ϵ^* is a threshold. Note that while we call ϵ^* the threshold, for our fault-tolerant constructions ϵ^* may more accurately be described a pseudothreshold, with the threshold being reserved for the limiting pseudothreshold [Svo+05]. To summarize our notation:

- ϵ_P is the physical error rate of a basic unit;
- ϵ_L is the error rate of a logical unit; and,
- ϵ^* is the fault-tolerance pseudothreshold.

In general, we have $\epsilon_\mathrm{L} \ll \epsilon_\mathrm{P} < \epsilon^*$.

There are two ways one might go about designing a circuit using faulty components to achieve a target logical error rate ϵ_L in light of the existence of fault-tolerant constructions. In the first, i.e. the non-fault-tolerant route, we may simply elect to work with higher fidelity gates, choosing physical error rate $\epsilon_\mathrm{P} = \epsilon_\mathrm{L}$. Alternatively, we may use a fault-tolerant construction, allowing us to choose to operate with gates at an intermediate error rate $\epsilon_\mathrm{P} \in (\epsilon_\mathrm{L}, \epsilon^*)$, where ϵ^* is the pseudothreshold of our fault-tolerant construction. The resource cost of the two routes is generally very different, and our goal will be to accurately model these costs so that the two routes may be compared quantitatively.

Next, we formalize the concept a fault-tolerant circuit construction. For the purposes of our discussion we make the following definition:

Definition 31 (Circuit)**.** A *circuit* is a triple $\mathcal{C} = (G, L, K, F)$ where

- L is a set of labels ℓ;
- K is a set of positive integers indexed by elements of L, $k_\ell \in \mathbb{Z}_+$;
- G is a directed acyclic graph with vertices V and edges E where each

vertex is associated with a label $\ell \in L$, and furthermore each vertex with label ℓ has either in-degree k_ℓ or in-degree 0; and,

- F is a set of Boolean functions indexed by elements of L, $F_\ell : \{0,1\}^{k_\ell} \to \{0,1\}$.

Vertices with in-degree 0 *inputs*, and those with out-degree 0 are called *outputs*.

We can associate with each label $\ell \in L$ in our definition with a type of gate with specified fan-in k_ℓ; each vertex in graph G then represents a gate with edges describing connectivity. Vertices with in-degree 0 are assumed to take a number of input wires dependent on the gate type and vertices with out-degree 0 are gates that provide a single output bit for each assignment of input wires.

Using our formal definition of a circuit, we turn to formalizing the fault-tolerant construction.

Definition 32 (Fault-tolerant circuit construction)**.** Let $\mathcal{C} = (G, L, K, F)$ be a circuit. Further let $\mathcal{R}_n = (e_n, d_n)$ for $n \in \mathbb{Z}_+$ be a $[n,1]$ error correcting block with encoder $e_n : \{0,1\} \to \{0,1\}^n$ and decoder $d_n : \{0,1\}^n \to \{0,1\}$. For a fixed gate set, a *fault-tolerant construction* $\mathcal{FT}_n$ over code $\mathcal{R}_n$ is specified by a family of operators that maps each vertex v of type ℓ in the graph G to a circuit C_v, i.e. $\mathrm{FT}_n^{(\ell)} : v \mapsto \mathcal{C}_v$, for $n \in \mathbb{Z}_+$ and all $\ell \in L$ satisfying the following properties:

i) The circuit $\mathrm{F}_n^{(\ell)}(v) = C_v$ can be written $C_v = (G_v, L, K, F)$, i.e. using the set of gates as $\mathcal{C}$;

ii) C_v has exactly k_ℓ *bundles* of inputs each of size n, and one size-n bundle output;

iii) for all n, the depth of circuit C_v is bounded by some constant $D \in \mathbb{Z}_+$;

iv) if all signals within the input bundles are set to $e_n(x_i)$, $i \in \{1, \ldots, k_\ell\}$, the outputs of C_v satisfies truth table $F_\ell \circ d_n$;

v) if input signals are subject to noise of strength Δ and the outputs of each gate are subject to noise ϵ_{P}, the output bundle C_v must amplify the signal for some non-empty range of Δ up to some pseudothreshold $\epsilon^* > 0$ for all $n > n_0 \in \mathbb{Z}_+$ (amplification formally defined in [SWH20]); and,

vi) in this amplification region, the probability of a logical error ϵ_{L} (i.e. a mismatch between the decoded output and desired truth table) as $n \to \infty$ satisfies $\epsilon_{\mathrm{L}} \sim e^{-\theta(n)}$.

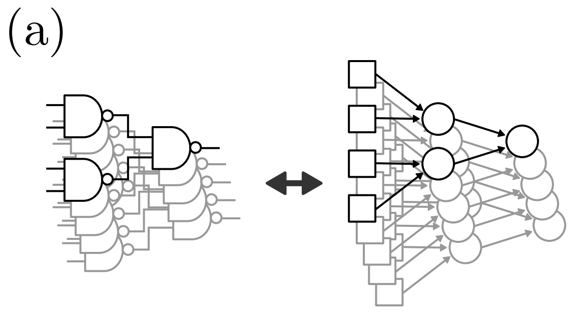
Given the family of operations F_n, one may derive an equivalent fault-tolerant by replacing each vertex using FT_n and connecting input and output bundles according to G. In general, a randomizing permutation may need to be applied to input or output bundles in order to minimize dependence between wires when gates are applied in sequence.

We denote by $\mathcal{FT}_n(\mathcal{C})$ the fault-tolerant circuit derived from $\mathcal{C}$ using this procedure.

Note that requirement Definition 32iii excludes the concatenation-based approaches to fault-tolerance employed, for example in [NC10; Imp04; Svo+05; Tha+05]. This is being deliberately done, in order to separate the width and depth components of the fault-tolerant construction discussed in Section 4.2.1. In the nomenclature of this paper, each level of the concatenation-based approach is equivalent to the choice of a different error correction architecture (see Remark 35).

Also, while Definition 32 allows data to be encoded in a general error correcting code, for the purposes of our discussion, we will consider following fault-tolerant construction based on the $[n, 1]$ repetition code and depth-2 majority circuit for error correction:

Remark 33 (Depth-2 fault-tolerant construction)**.** Consider the construction studied by [Neu56; EP98; Ung07] using 2-input NAND gates, i.e. $L = \{\text{NAND}\}$ and $k_{\text{NAND}} = 2$. Our operator $\mathrm{FT}_n^{(\text{NAND})}$ replaces each NAND gate in the original circuit with n NAND gates. These n NAND gates are then followed by $2n$ gates used for error correction. If we label each gate by a tuple (l, i) for layer $l \in \{1, 2\}$ and index $i \in \{0, \dots, n-1\}$, the construction connects (l, i) to the output of gates $(l-1, i)$ and $(l-1, i-1 \mod n)$; where $(0, i)$ denotes the ith input wire. The error correction component of this construction is depicted in Fig. 4.1.



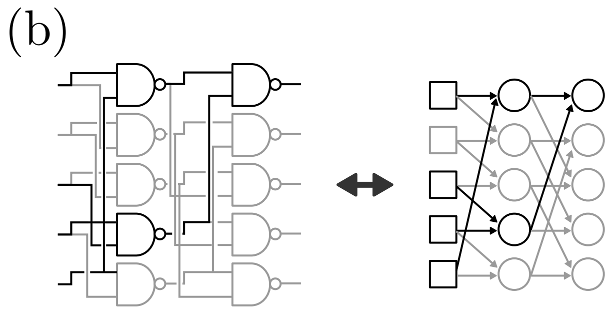


Figure 4.1: Error correcting depth-2 majority (a) formula, and (b) an equivalent circuit for the repetition code of size $n = 5$. In both cases, the circuit is shown on the left, with the induced Bayesian network depicted on the right (squares representing input wires, and circles representing computed wires). The computation path of one output wire is highlighted in black throughout with the rest of the circuit in grey.

We discuss the properties of depth-2 construction of Remark 33 in Sections 4.2.2 and 4.2.3.

Equipped with $\mathcal{FT}_n$, we are able to convert any circuit into an equivalent fault-tolerant circuit. The subject of the remainder of this paper will be a careful accounting of the resources required to target a logical error rate using this construction. Assuming that the resource utilization of a circuit is proportional to the number of basic units of which the circuit is comprised, the number of basic units required by the fault-tolerant circuit is of central importance.

We define the *number overhead* (or equivalently the *gate overhead* for circuit-based models of computation) to be the number of additional basic components required by $\mathcal{FT}_n(\mathcal{C})$ when compared with $\mathcal{C}$. For the remainder of this Section, we turn our focus to the asymptotics of this number overhead in terms of the number of basic units required to achieve a desired logical error rate, specifically the constants in the exponent of Definition 32vi. The *resource overhead* required for fault-tolerance can be calculated given the number overhead and the resource–reliability trade-off for the basic unit and will be the subject of Section 4.3.

4.2.1 Scaling arguments

For our analysis, we separate the overhead into three components. The first is due to the required code size, i.e. its 'width' overhead, which depends on the target error rate ϵ_{L}, and the error rate of the individual output wires ϵ. The second, the 'depth' overhead, is related to the fraction of layers dedicated to error correction and is dependent on the operating operating point ϵ_{P} and a fiducial error rate $\Delta \in (0, 1/2)$; in other words, if r denotes the number of wires in the repetition code of size n in the 1 state, a logical 1 is encoded if $r > n(1-\Delta)$, a logical 0 is encoded if $r < n\Delta$, and otherwise the signal is considered erroneous. In the fault-tolerant regime, we have that the probability of error in any wire can be maintained $\Delta < \frac{1}{2} - \Omega(1)$ independently of circuit depth. The final component is due to an effective reduction of code size due to statistical dependencies in the output of circuits which is absent in the case of formulas.

Width overhead— A key property of fault-tolerance is that a constant, depth-independent logical error rate is maintained by interleaving computation with layers dedicated to error correction. In order to accomplish this, the data must be encoded in an error correcting code throughout the computation. Increasing the size of this code is the key to reducing logical errors and is the width overhead. Since the logical error rate is suppressed by $e^{\Theta(d)}$, where d is the distance of the code. From a simple concentration bound, we find that it suffices to choose a code of distance

$$d(\epsilon, \epsilon_{\mathrm{L}}) = \Theta\left(\log\left(\frac{1}{\epsilon_{\mathrm{L}}}\right)\right). \tag{4.1}$$

Assuming the use of a linear-distance code, this implies a width overhead of $\Theta(\log(1/\epsilon_{\mathrm{L}}))$. In the fault-tolerance setting, we must account both for errors introduced during computation and in the process of error correction and thus the hidden constant factor in Eq. (4.1) is in general also dependent on ϵ_{P} and Δ. In general, we write the width overhead as $n(\epsilon_{\mathrm{L}}; \epsilon_{\mathrm{P}}, \Delta)$.

One may object to the fact that in the fault-tolerant construction, the inputs and outputs of the computation are encoded in a bundles of n wires, and not strictly comparable to that of the non-fault-tolerant construction. While this is true, we may assume that we have access to a perfect (or otherwise highly reliable) encoder and decoder which we may invoke respectively at the beginning and end of the computation. The use of these idealized encoders and decoders is simply a constant cost which contributes negligibly to the overall resource cost in the limit of long computations, which is the primary concern of fault-tolerant constructions.

Depth overhead— For any error correction construction satisfying Definition 32, there is a range of physical error rates, i.e. $\epsilon_{\mathrm{P}} < \epsilon^*$, and fiducial rates Δ, which can be tolerated. In general, we may choose an ϵ_{P} and Δ so long as it is within the ranges allowable by the error correcting circuit. In practice, ϵ_{P} is chosen by physical constraints (i.e. to minimize resource overhead), and Δ is chosen to minimize the previously discussed width overhead. By Definition 32iii, the error correcting circuit has constant a depth D independent of code size, and D is precisely the depth overhead.

Dependency overhead— Finally, since a circuit's outputs are in general statistically dependent, the width overhead may need to be scaled to appropriately counteract that dependence, We characterize the degree to which the error correcting circuits are independent by a factor χ, which we call the *independency* (or equivalently the *dependency* χ^{-1}). In order for a family of error correcting circuits to be valid for a fault-tolerant construction, it must be that $\chi > 0$. This is possible since, by Definition 32, the family of error correcting circuits is comprised of gates of constant fan-in and constant depth which effectively limits the possible dependency of the circuit's outputs—of course this assume non-pathological wiring which we will discuss in Section 4.2.3.

This constant χ is in general difficult to estimate as it depends on the specifics of the circuit construction, which becomes exponentially (with the number of wires in the circuit) to calculate exactly. In order to numerically estimate χ, we represent a noisy circuit as a Bayesian network where each wire is represented by a node, and a directed edge connects input wires to output wires of each gate; an example using the error correcting formula of [EP98] and equivalent error correcting circuit of [Neu56], along with their induced Bayesian network representations is depicted in Fig. 4.1. Using this Bayesian network, we are able to estimate a circuit's performance to high precision and estimate its asymptotic behavior.

In the following, we seek an estimate of the coefficient hidden in Eq. (4.1) (for small

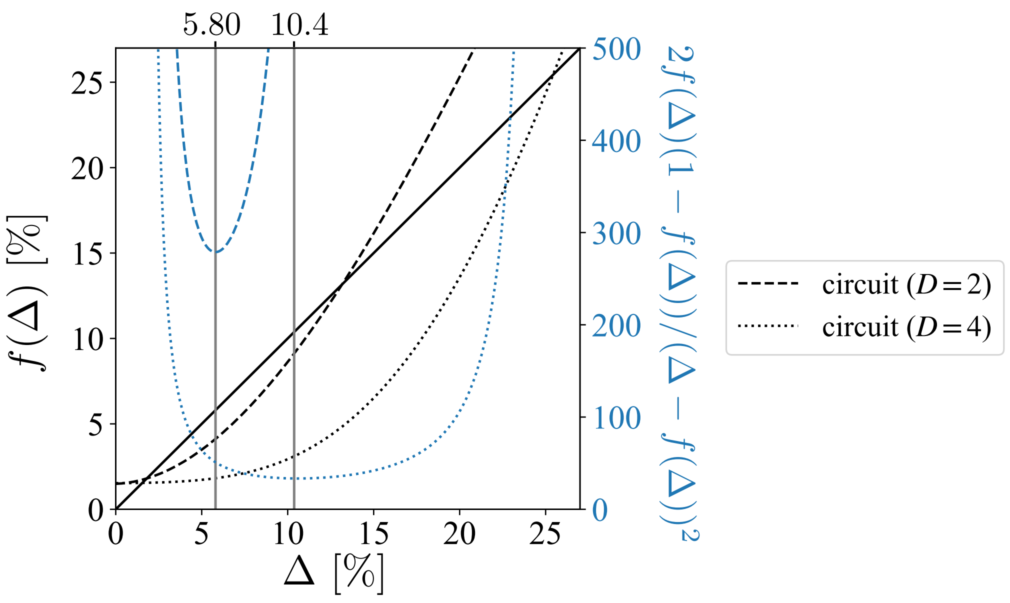


Figure 4.2: Output error rate $f_{\epsilon_P}(\Delta)$ (on the left y-axis) as a function input error rate Δ for $\epsilon_{\mathrm{P}} = 0.5\%$ from Eq. (4.2) compared to the $y = \Delta$ line (solid black) for different error correction circuit constructions. The leading term in the code size $n(\epsilon_L; \epsilon_P, \Delta)$, the coefficient of $\log(1/\epsilon_{\mathrm{L}})$ in Eq. (4.5), is shown on the right-hand y-axis and plotted with blue lines, using the same x-axis. Its minima for the depth-2 and depth-4 majority formulas are marked (solid grey vertical lines). For the depth-2 majority formula (Remark 33) this gives a minimum required code size of $n \approx 279 \log(1/\epsilon_{\mathrm{L}})$, and for the depth-4 majority formula (Remark 34) this gives a minimum required code size of $n \approx 11.4 \log(1/\epsilon_{\mathrm{L}})$.

ϵ_{L}), and an estimate of χ for the fault-tolerant construction using 2-input NAND gates of Remark 33.

4.2.2 Formulas

We start with the simpler case of formulas for which outputs are independent. In the case of formulas, the Bayesian network takes the form of a directed tree and each node is required to have out-degree 1 (except for the output(s) with out-degree 0), the roots of all disjoint sub-trees are independent conditioned on their ancestors. Thus the analysis is simplified by the independence of the output wires.

Following von Neumann's construction [Neu56], given $4n$ independent copies of each signal X and Y, one performs the NAND gate $4n$ times and takes the majority using the configuration of Fig. 4.1. Assuming each input wire errs with probability at most Δ, the output wire errs with probability bounded by

$$f_{\epsilon_{\mathrm{P}}}(\Delta) = 1 - \epsilon_{\mathrm{P}} + (2\epsilon_{\mathrm{P}} - 1)\left((2\epsilon_{\mathrm{P}} - 1)((\Delta - 2)\Delta(2\epsilon_{\mathrm{P}} - 1) + \epsilon_{\mathrm{P}})^2 - \epsilon_{\mathrm{P}} + 1\right)^2. \tag{4.2}$$

For large n, we can approximate the binomial distributed output with a normal distribution as a result of the central limit theorem. Given the independence of the output wires, we have

$$\epsilon_{\mathrm{L}}(n; \epsilon_{\mathrm{P}}, \Delta) = \Pr\left[Z \geq \sqrt{n}\left(\frac{\Delta - f_{\epsilon_{\mathrm{P}}}(\Delta)}{\sqrt{f_{\epsilon_{\mathrm{P}}}(\Delta)(1 - f_{\epsilon_{\mathrm{P}}}(\Delta))}}\right)\right], \tag{4.3}$$

where Z is a standard normal random variable, $\Delta \in (0, 1/2)$ is a fiducial error rate, and we have dropped the subscript ϵ_{P} for convenience. Using Eq. (4.3), we find in the limit of large n,

$$\epsilon_{\mathrm{L}} = \left[\frac{f_{\epsilon_{\mathrm{P}}}(\Delta)(1 - f_{\epsilon_{\mathrm{P}}}(\Delta))}{\Delta - f_{\epsilon_{\mathrm{P}}}(\Delta)} \frac{1}{\sqrt{2\pi n}} + O\left(n^{-\frac{3}{2}}\right)\right] \exp\left(-\frac{n(\Delta - f_{\epsilon_{\mathrm{P}}}(\Delta))^2}{2f_{\epsilon_{\mathrm{P}}}(\Delta)(1 - f_{\epsilon_{\mathrm{P}}}(\Delta))}\right), \tag{4.4}$$

giving a code size,

$$n(\epsilon_{\mathrm{L}}; \epsilon_{\mathrm{P}}, \Delta) = \frac{2f_{\epsilon_{\mathrm{P}}}(\Delta)(1 - f_{\epsilon_{\mathrm{P}}}(\Delta))}{(f_{\epsilon_{\mathrm{P}}}(\Delta) - \Delta)^2} \log\left(\frac{1}{\epsilon_{\mathrm{L}}}\right) + O\left(\log\log\left(\frac{1}{\epsilon_{\mathrm{L}}}\right)\right). \tag{4.5}$$

In practice, for a given architecture and ϵ_{P}, one chooses Δ in order to maximize the coefficient of $\sqrt{n}$ in Eq. (4.3). For this fault-tolerant construction with output error rate bounded by Eq. (4.2), we find a pseudothreshold of $\epsilon^* \approx 1.077\%$. Given a physical error rate below this threshold, we may choose an operating point between the fixed-points of Eq. (4.2). As an example, for $\epsilon_{\mathrm{P}} \approx 0.5\%$, these fixed points occur around $\Delta \approx 1.81\%$ and $\Delta \approx 13.2\%$; and we find from Eq. (4.3) an optimal fiducial rate of $\Delta \approx 5.80\%$ as shown in Fig. 4.2 which implies a required code size of $n \approx 279 \log(1/\epsilon_{\mathrm{L}})$. Further note that in this fault-tolerant construction, two of every three layers are dedicated to error correction, i.e. the depth overhead is $D = 2$.

Remark 34 (Larger error correcting formulas)**.** In general, the pseudothreshold may be improved by increasing the size of the error correcting circuit.

For the example of computing with 2-input NAND gates, the majority operation can be constructed using full binary trees of even depth $D \geq 2$; for code size n, this formula representation requires $n(2^D - 1)$ gates. The depth-4 majority formula has an increased pseudothreshold of $\epsilon^* \approx 2.515\%$, strictly greater than that of the depth-2 majority gate. In addition to the increased threshold, the larger majority formula also has different fixed points and a different optimal fiducial rate. For example, for $\epsilon_\mathrm{P} = 0.5\%$, the depth-4 formula reduces input errors on expectation for fiducials between $\Delta \approx 1.55\%$ and $\Delta \approx 25.4\%$, with optimal performance for $\Delta \approx 10.4\%$.

While the analysis of formulas is easier and provides insight into the fault-tolerant construction, it is not suitable for implementation as the number of required gates grows exponentially with depth: since sub-formulas must be duplicated each time its result is required in order to maintain the fan-out 1 condition. More realistically, by allowing larger fan-out, we can 'fold' this formula representation into a more general circuit.

4.2.3 Circuits

As previously discussed, the analysis for circuits is complicated by the fact that outputs are no longer necessarily independent. In addition to the width and depth overheads, the independency $\chi \in (0, 1]$ contributes to the asymptotic number overhead of a fault-tolerant circuit. Intuitively, since output wires are no longer independent, logical error rates are higher than in equivalent formulas which by definition have $\chi = 1$.

From Definition 32iii, error correcting circuits are of constant depth, and therefore output wires are dependent only on constant number of input wires as depicted in Fig. 4.1 (this is our main reason for including requirement iii, and excluding concatenation-based constructions). Therefore the same asymptotic scaling $n = \Theta(\log(1/\epsilon_\mathrm{L}))$ holds as in the case with formulas but with a smaller effective code size $n_\mathrm{eff} \equiv \chi n$, where we call the constant $\chi \in (0, 1]$

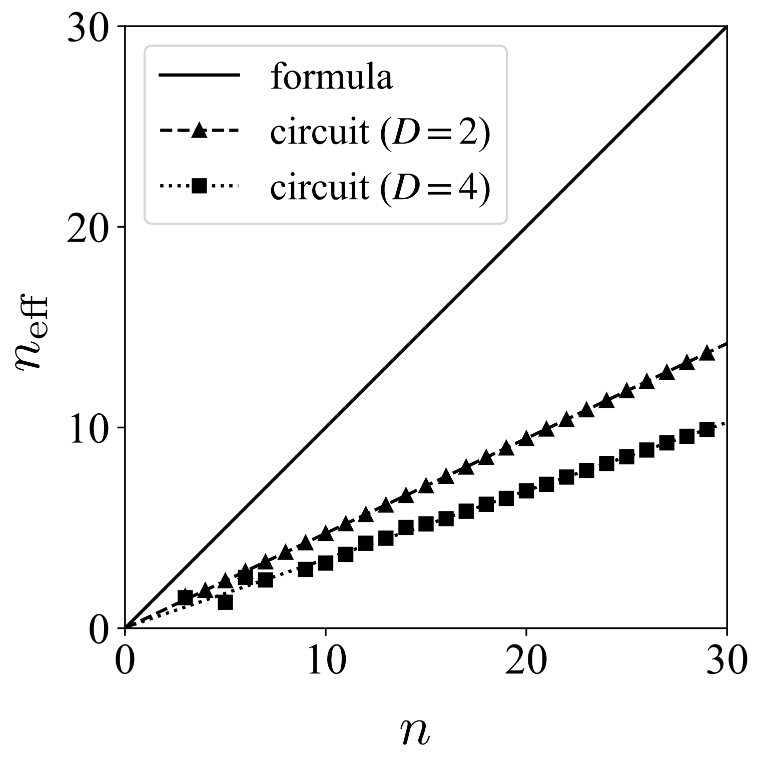


Figure 4.3: The effective code size for formulas and circuits for the fault-tolerant construction of Remark 33 and Remark 35 operating subject to physical error rate $\epsilon_{\text{P}} = 0.5\%$ and at their optimal fiducial point. The slope of the formula (solid) line is 1 and the fitted lines for the $D = 2$ (dashed) and $D = 4$ circuits (dotted). We find $n_{\text{eff}} \approx 0.47n$ and $n_{\text{eff}} \approx 0.34n$ for the depth-2 and depth-4 error correction circuits respectively.

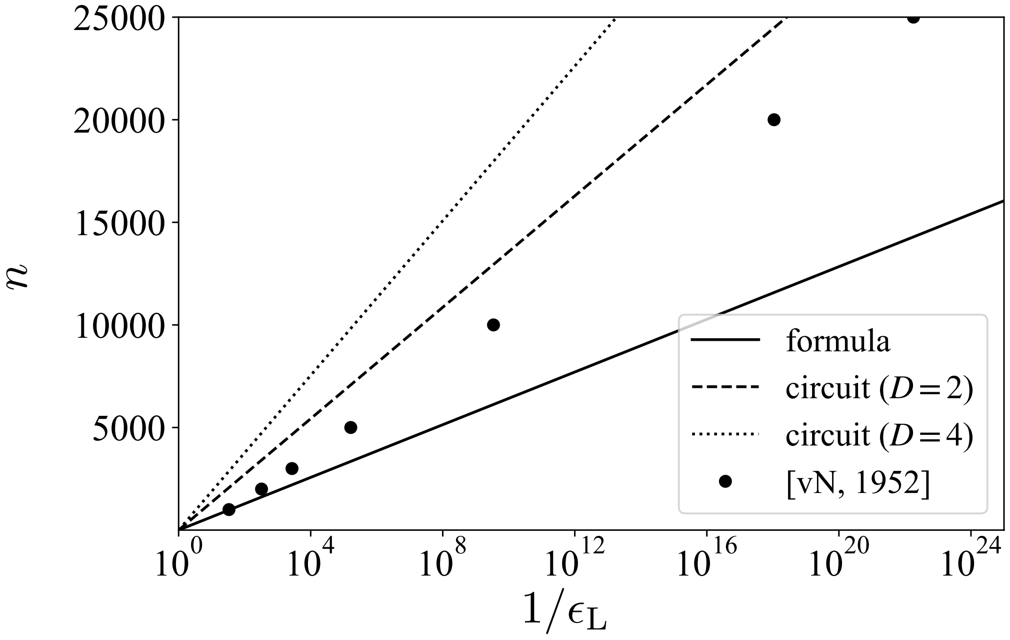


Figure 4.4: The code size required to achieve a target logical error rate ϵ_{L} for odd n at $\epsilon_{\mathrm{P}} = 0.5\%$ for different majority circuit constructions at their respective optimal fiducial point. The code size required using the majority formula (solid) is $n \approx 642 \log_{10}(1/\epsilon_{\mathrm{L}})$, for the depth-2 majority circuit (dashed) code size is $n \approx 1360 \log_{10}(1/\epsilon_{\mathrm{L}})$, and for the depth-4 majority circuit (dotted) the code size is $n \approx 1880 \log_{10}(1/\epsilon_{\mathrm{L}})$. Also plotted is von Neumann's approximation for a similar formula-based error correction scheme with the same physical error rate and slightly different choice of fiducial $\Delta = 7.0\%$ [Neu56, Section 10.5.2].

(Definition 32vi precludes $\chi = 0$). In the case of the depth-2 NAND tree majority circuit, output wires are dependent on at most 4 input wires and therefore $\chi \geq 1/4$. Use belief propagation to numerically approximate error rates using the induced Bayesian network representation (see Fig. 4.1), we find an independency of $\chi \approx 0.47$ for the depth-2 error correction circuit (see Fig. 4.3). In general χ is dependent on the error correction circuit and choice of fiducial Δ.

Of course, larger error correcting circuits, beyond the depth-2 construction of Remark 33, can also be used to perform error correction in a fault-tolerant construction. We provide a discussion of such circuits below:

Remark 35 (Larger error correcting circuits)**.** The majority formulas of Remark 34 can be 'wrapped' into depth-D majority circuits using nD NAND gates. For optimal performance, a path should connect each output gate to the maximal number, i.e. 2^D, of distinct input wires (for simplicity assume $n \geq 2^D$). If we label each gate by a tuple (ℓ, i) for layer $\ell \in \{1, \dots, D\}$ and index $i \in \{0, \dots, n-1\}$, one such construction is to connect gate (ℓ, i) to the output of gates $(\ell - 1, i)$ and $(\ell - 1, i - 2^{\ell-1} \mod n)$; where $(0, i)$ denotes the i^{th} input wire.

As with the depth-2 case (Remark 33), larger majority circuits are less effective at error correction than their formula counterparts due to the dependence of the output signals. Numerics show an independency of $\xi \approx 0.34$ for depth-4 circuits operating at $\epsilon_{\mathrm{P}} = 0.5\%$ at the optimal fiducial rate.

It is widely conjectured, though not rigorously proven, that the pseudothreshold of such a circuit approaches the Evans–Pippenger rate of $\epsilon_{\mathrm{EP}} = (3 - \sqrt{7})/4 \approx 0.08856$ as we take $D \to \infty$ [Neu56; EP98; Ung07].

Note that there are other ways to fold the formulas into circuits which may result in poor performance. For example, imagine that all outputs were made dependent on the same 2^D inputs independently of n. In this case, we will find that increasing the code size does not decrease the error and $\chi = 0$. This is true despite the fact that it shares the same unfolded formula representation as the construction described above. Though this is a pathological example, one may find that some reasonable circuit constructions result in lower independency.

Notice that each computation gate in the non-fault-tolerant circuit is repeated $n(\epsilon_\mathrm{L}; \epsilon_\mathrm{P}, \Delta)$ times in the fault-tolerant circuit, and followed by an error correction circuit of size $D \times n(\epsilon_\mathrm{L}; \epsilon_\mathrm{P}, \Delta)$. Therefore, for fixed ϵ_P, the gate overhead is

$$\eta_\#(\epsilon_\mathrm{L}; \epsilon_\mathrm{P}, \Delta, D, \chi) = \frac{1}{\chi}(D+1)n(\epsilon_\mathrm{L}; \epsilon_\mathrm{P}, \Delta), \tag{4.6}$$

where we use $\eta_\#$ to denote the overhead in the number of gates. When a particular error correction construction is assumed, we may drop explicit dependence on the variables to the right of the semicolon in the arguments of $\eta_\#$ and n, i.e. D, χ, ϵ_P, and Δ.

To summarize, the asymptotic number overhead of a fault-tolerant circuit can be deconstructed into three contributions:

- the width or code size overhead $n(\epsilon_\mathrm{L}; \epsilon_\mathrm{P}, \Delta)$, which can be obtained through the analysis of its equivalent formula representation;
- the depth overhead, i.e. the constant in Definition 32iii; and,
- the dependency overhead χ^{-1}, which is a function of how the formula is 'folded' into a circuit, and can be obtained by numerical means (e.g. using belief propagation or Monte Carlo sampling).

4.3 Fault-tolerance resource overhead

Having derived the number overhead (Eq. (4.6)) of the fault-tolerant construction of Remark 33, we turn to its resource overhead. To this end, we introduce the notion of a *resource–reliability trade-off* which associates a resource cost for each gate as a function of the physical error rate. In order to formalize the problem, let $W : (0, \epsilon^*) \to \mathbb{R}_{\geq 0}$ denote the resource–reliability trade-off: $W(\epsilon_\mathrm{P})$ represents the resource cost of a single gate operating with physical error rate ϵ_P.

Using the resource–reliability trade-off in conjunction with the number overhead (Section 4.2), we study the cases in which either the fault-tolerant or non-fault-tolerant constructions may be preferred in order to reduce overall resource utilization. While the specific resources of interest are application-specific—e.g. power dissipation [Imp04; Tha+05; CV16; CV20b], layout area [Imp04; Tha+05; Gad+06; Oat15] for transistor-based gates, or even economic costs—we find general conclusions that depend only on the asymptotic behavior of this resource–reliability trade-off in the low-error limit $\epsilon \to 0$.

We make the physical assumption that $W(\epsilon_{\mathrm{P}})$ is monotonically non-increasing with respect to ϵ_{P}: that is, a reduced physical error rate necessitates equal or larger resource expenditure. With the resource–reliability function in hand, we can state our main asymptotic result:

Remark 36 (Fault-tolerant overhead theorem)**.** Suppose we have a fault-tolerant construction with number overhead $\eta_{\#}(\epsilon_{\mathrm{L}}; \epsilon_{\mathrm{P}}, \Delta, D, \chi)$, where ϵ_{L} is the target logical error rate and ϵ_{P}, Δ, D, and χ are parameters of the fault-tolerant construction. In the limit of small logical error rates $\epsilon_{\mathrm{L}} \to 0$, the non-fault-tolerant construction is preferred for resource–reliability trade-offs which are asymptotically $W(\epsilon_{\mathrm{P}}) = o(\log(1/\epsilon_{\mathrm{P}}))$ as $\epsilon_{\mathrm{P}} \to 0$. Conversely, in the same small logical error rate limit, the fault-tolerant construction is preferred for resource–reliability trade-offs which are asymptotically $W(\epsilon_{\mathrm{P}}) = \omega(\log(1/\epsilon_{\mathrm{P}}))$ as $\epsilon_{\mathrm{P}} \to 0$.

Remark 36 is a statement about the low logical error rate asymptotics of the resource–reliability trade-off; however, for most practical purposes it is sufficient to achieve a low, but finite, logical error rate. For a sense of scale, a study estimated error rates of ~ 10 errors per billion hours of operation per gate in modern transistor-based logic [WA08]. Assuming such gates perform 10^9 operations per second, this corresponds to a logical per gate error rate of $\epsilon_{\mathrm{L}} \approx 3 \times 10^{-21}$. In the subsequent analysis, we estimate the benefit of fault-tolerance (or lack thereof) assuming values for the behavior of the resource–reliability trade-off required to achieve logical error rates down to this level.

To further motivate the resource–reliability trade-off, $W(\epsilon_{\mathrm{P}})$, consider that the output of each gate is a bit $x \in \{0,1\}$ encoded in some real-valued physical degree of freedom $\{-R, +R\}$, where $R > 0$ denotes the signal strength. Suppose the signal is corrupted by noise e, which for our purposes will be a symmetric, zero mean random variable, so that our overall signal is $X_\pm = \pm R + e$ where e denotes an error term. We consider $X_\pm$ to be correct if it lies on the correct side of 0, and therefore the physical error rate is $\epsilon_\mathrm{P}(R) = \Pr[X_+ < 0|R] = \Pr[X_- > 0|R]$. Further assume that the noise e is i.i.d. at the output of each gate, and that the resource utilization is proportional to $W \propto L$.

This model allows us to relate the resource–reliability trade-off $W(\epsilon_\mathrm{P})$ to the distribution of the noise e. For our setting of interest, the resource utilization in the limit of low logical error rates, there are broadly three scenarios to consider: exponentially-tailed, light-tailed, and heavy-tailed noise distribution. We discuss these cases in sequence below.

4.3.1 Exponentially-tailed noise distribution

First, consider the marginal case where the resource–reliability trade-off has the same asymptotic scaling as the number overhead, i.e. $W(\epsilon_\mathrm{P}) = \Theta(\log(1/\epsilon_\mathrm{P}))$. In our abstract model, this would correspond to an error e with exponentially distributed tails so that as $\epsilon_\mathrm{P} \to 0$ or equivalently $R \gg \alpha$, we have

$$\epsilon_\mathrm{P} \sim C \exp\left(-\frac{R}{\alpha}\right), \tag{4.7}$$

for some constants $\alpha > 0$ and $C > 0$.

In this case as $\epsilon \to 0$,

$$W(\epsilon_\mathrm{P}) \sim \alpha A \log\left(\frac{1}{\epsilon_\mathrm{P}}\right), \tag{4.8}$$

where $A > 0$ is a proportionality constant.

Combining the number overhead (Eq. (4.6)) and assuming exponentially-tailed resource–reliability trade-off (Eq. (4.8)), we find the resource overhead as $\epsilon_\mathrm{L} \to 0$ required for fault-

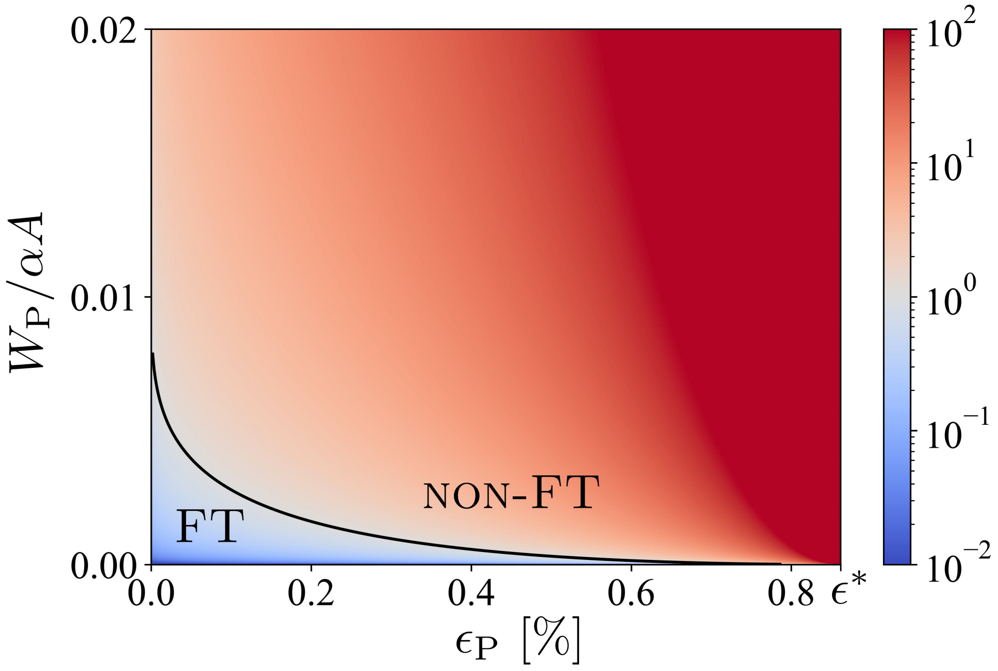


Figure 4.5: Plot of the asymptotic fault-tolerance overhead in the $\epsilon_{\mathrm{L}} \to 0$ limit for exponentially-tailed resource functions as predicted by Eq. (4.9) for a fault-tolerant construction based on the depth-2 majority circuit of Remark 33 and optimal fiducial Δ. The region where the fault-tolerant construction is less resource intensive is marked (FT) and its complement is marked (NON-FT).

tolerance to be

$$\begin{aligned}\eta(\epsilon_{\mathrm{L}};\epsilon_{\mathrm{P}},\Delta,D,\chi) &= \frac{W(\epsilon_{\mathrm{P}})}{W(\epsilon_{\mathrm{L}})}(D+1)n(\epsilon_{\mathrm{L}};\epsilon_{\mathrm{P}},\Delta),\\ &= \left(\frac{W_{\mathrm{P}}}{\alpha A}\right)\left(\frac{2(D+1)f_{\epsilon_{\mathrm{P}}}(\Delta)(1-f_{\epsilon_{\mathrm{P}}}(\Delta))}{\chi(f_{\epsilon_{\mathrm{P}}}(\Delta)-\Delta)^2}\right),\end{aligned} \tag{4.9}$$

where $W_{\mathrm{P}} > 0$ represents the resource utilization at finite physical error rate ϵ. As with Eq. (4.6), when a particular error correction construction is assumed, we may drop explicit dependence on the variables to the right of the semicolon in the arguments of η.

Note the fault-tolerance overhead as $\epsilon_{\mathrm{L}} \to 0$ is independent of ϵ_{L} in this case of an exponentially-tailed error function (or resource–reliability trade-off satisfying Eq. (4.8)). In the case of exponentially-tailed errors, both increasing R and increasing the code size n (assuming a linear distance code) suppress errors exponentially; and thus, which approach is preferred to achieve a logical error rate $\epsilon_{\mathrm{L}} \to 0$ depends on the specifics of both the resource–reliability trade-off and the fault-tolerant construction. In Eq. (4.9) relevant terms have been grouped into those that depend on the asymptotic behavior of the resource utilization model: W_{P}, α and A; and those that depend on the fault-tolerant circuit construction: ϵ_{P}, $f_{\epsilon_{\mathrm{P}}}$, Δ, D, and χ. The overhead is plotted in Fig. 4.5. Regions are indicated where one construction, i.e. fault-tolerant or non-fault-tolerant, is preferred.

4.3.2 Light-tailed resource function

In the case of light-tailed error distributions, i.e. those whose tails are super-exponential, we will show that the non-fault-tolerant construction is always preferred as the target error rate $\epsilon_{\mathrm{L}} \to 0$. For simplicity, consider the case of an error distribution with a Gaussian tail. For $R \geq 0$,

$$\epsilon_{\mathrm{P}}(R) \sim \left[\frac{C}{\sqrt{2\pi}R} + O\left(\frac{1}{R^3}\right)\right]\exp\left(-\frac{R^2}{2\sigma^2}\right), \tag{4.10}$$

for constants $C > 0$ and $\sigma > 0$ as $R \to \infty$.

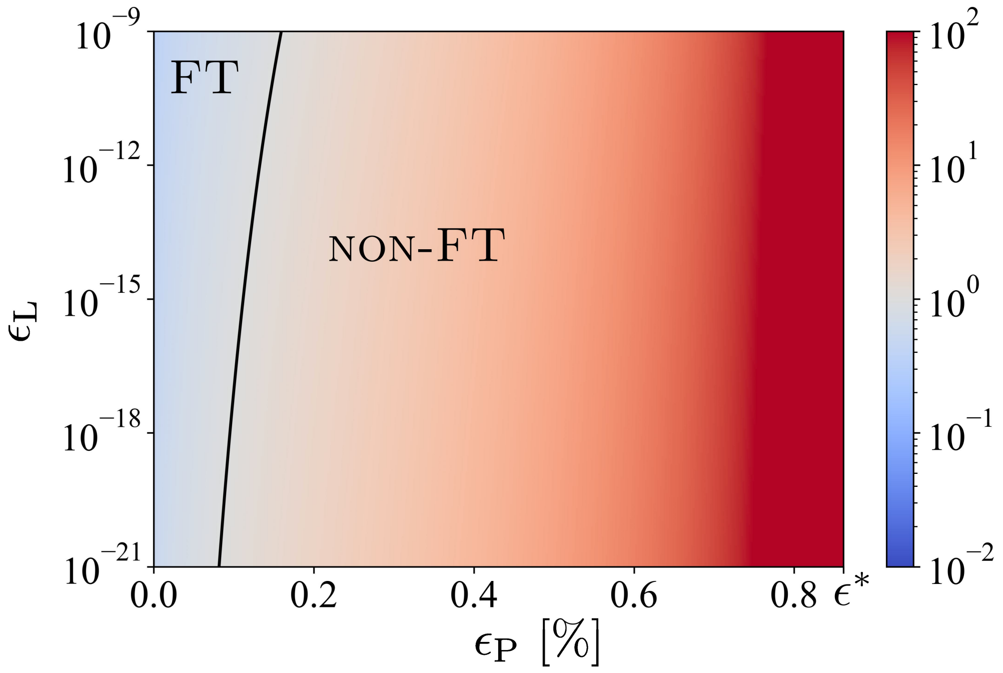


Figure 4.6: Plot of the asymptotic fault-tolerance overhead in the $\epsilon_L \to 0$ limit for Gaussian-tailed errors as predicted by Eq. (4.12) for a fault-tolerant construction based on the depth-2 majority circuit of Remark 33, optimal fiducial Δ, and with constants $W_P/\sqrt{2}\sigma A = 2\times10^{-4}$. The region where $\eta < 1$ indicates the regions where the fault-tolerant construction is less resource intensive is marked (FT) and its complement is marked (NON-FT). It is possible that no FT region exists for some settings of the constants $W_P/\sqrt{2}\sigma A$.

In this case as $\epsilon \to 0$,

$$W(\epsilon_{\mathrm{P}}) \sim \sqrt{2}\sigma A \,\mathrm{erfc}^{-1}(2\epsilon_{\mathrm{P}}), \tag{4.11}$$

where $A > 0$ is a proportionality constant.

Using the number overhead (Eq. (4.6)), we find the following expression for the overhead in the case of a Gaussian-distributed error e,

$$\eta_{\mathrm{Gaussian}}(\epsilon_{\mathrm{L}}; \epsilon_{\mathrm{P}}, \Delta, D, \chi) = \left(\frac{W_{\mathrm{P}}}{\sqrt{2}\sigma A \,\mathrm{erfc}^{-1}(2\epsilon_{\mathrm{L}})}\right)\left(\frac{2(D+1)f_{\epsilon_{\mathrm{P}}}(\Delta)(1-f_{\epsilon_{\mathrm{P}}}(\Delta))}{\chi(f_{\epsilon_{\mathrm{P}}}(\Delta)-\Delta)^2}\log(1/\epsilon_{\mathrm{L}})\right), \tag{4.12}$$

where W_{P} is the resource utilization at finite physical error rate ϵ_{P}.

From Eq. (4.12) we observe that in the $\epsilon_{\mathrm{L}} \to 0$ limit, $\eta_{\mathrm{Gaussian}} > 1$, and therefore the non-fault-tolerant construction is preferred. It is nonetheless possible that for some finite ϵ_{L}, there exists a setting of ϵ_{P} where the fault-tolerant construction is more resource efficient in this case as the efficiency at finite logical error rates depends on $W_{\mathrm{P}}/\sqrt{2}\sigma A$, which depends on resource utilization at finite physical error rates. One setting of parameters that admits a region of fault-tolerant resource savings is shown in Fig. 4.6.

Light-tailed error distributions, can arise in a number of scenarios. Gaussian-distributed noise may occur naturally as a result of the central limit theorem when noise sources are the sum of many independent components. The fundamental physical limits on power dissipation are themselves light-tailed resource–reliability trade-offs. Landauer famously argued that in an idealized computer, the only dissipation necessary is proportional to the amount of logical irreversibility [Lan61]; Fredkin and Toffoli subsequently showed that computations may be designed to require only constant amounts of logical irreversibility [FT82]. Given a well isolated system with idealized control of microscopic degrees of freedom, one would find a resource–reliability trade-off such that $\lim_{\epsilon \to 0} W(\epsilon) = C$ for some constant C (the primary resource of concern in this case would likely not be power dissipation but rather the economic cost of engineering such a system).

We note that while these results are for Gaussian-tailed noise, the qualitative result—

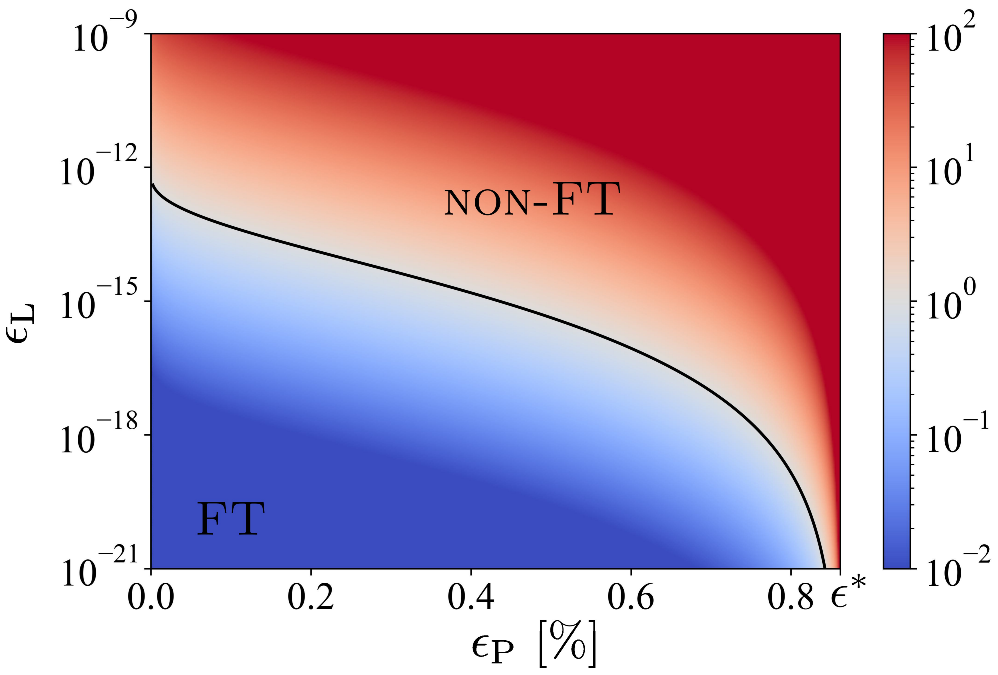


Figure 4.7: Plot of the asymptotic fault-tolerance overhead in the $\epsilon_{\mathrm{L}} \to 0$ limit for Pareto-tailed errors as predicted by Eq. (4.15) for a fault-tolerant construction based on the depth-2 majority circuit of Remark 33, optimal fiducial Δ, $\gamma = 2$, and constants $W_{\mathrm{P}}/A\beta = 3 \times 10^2$. The region where the fault-tolerant construction is less resource intensive is marked (FT) and its complement is marked (NON-FT). In this case of a heavy-tailed resource function, the fault-tolerant construction is always preferred as the target error rate goes to zero.

namely that the non-fault-tolerant construction is preferred in the $\epsilon_{\mathrm{L}} \to 0$ limit—hold as long as the noise has super-exponential tails. This can be understood as physical error rates may be suppressed through by increased resource utilization faster than the exponential suppression of logical error rates using a linear-distance error correcting code.

4.3.3 Heavy-tailed resource function

In the case of heavy-tailed error distributions, i.e. those whose tails are sub-exponential, we find that the fault-tolerant construction is always preferred as the target error rate $\epsilon_{\mathrm{L}} \to 0$ (Remark 36). For simplicity, assume that the error e is distributed according to a symmetric

Pareto distribution so that for $R \geq 0$,

$$\epsilon_{\mathrm{P}}(R) = \frac{1}{2}\left(\frac{\beta}{\beta + R}\right)^{\gamma}, \tag{4.13}$$

for some constants $\beta > 0$ and $\gamma > 0$.

In this case for $\epsilon_{\mathrm{P}} \to 0$ we have,

$$W(\epsilon_{\mathrm{P}}) \sim A\beta\left(\frac{1}{(2\epsilon_{\mathrm{P}})^{1/\gamma}} - 1\right), \tag{4.14}$$

where $A > 0$ is again a proportionality constant.

Using the number overhead (Eq. (4.6)), we find the following expression for the overhead in the case of a Pareto-tailed error e,

$$\eta(\epsilon_{\mathrm{L}}; \epsilon_{\mathrm{P}}, \Delta, D, \chi)_{\mathrm{Pareto},\gamma} = \left(\frac{W_{\mathrm{P}}}{A\beta}\frac{(2\epsilon_{\mathrm{L}})^{1/\gamma}}{1-(2\epsilon_{\mathrm{L}})^{1/\gamma}}\right)\left(\frac{2(D+1)f_{\epsilon_{\mathrm{P}}}(\Delta)(1-f_{\epsilon_{\mathrm{P}}}(\Delta))}{\chi(f_{\epsilon_{\mathrm{P}}}(\Delta)-\Delta)^2}\log(1/\epsilon_{\mathrm{L}})\right), \tag{4.15}$$

where W_{P} is again the resource utilization at finite physical error rate ϵ_{P}.

From Eq. (4.15) we observe that in the $\epsilon_{\mathrm{L}} \to 0$ limit, $\eta_{\mathrm{Pareto}} < 1$, and therefore the fault-tolerant construction is preferred. A representative plot of η_{Pareto} is shown in Fig. 4.7.

There are a number of ways in which a heavy-tailed error distribution may arise. One example discussed by Thaker et al. is the case where the resource of interest is of CMOS feature sizes, where the salient resource is the feature area [Tha+05]. Cohen et al. demonstrated experimentally that CMOS technology exposed to high levels of alpha radiation resulted in a resource–reliability trade-off with asymptotically Pareto tails with $\gamma \approx 2.5$ [Coh+99].

We note that while these results are for symmetric Pareto-distributed noise, the qualitative result—namely that the fault-tolerant construction is preferred in the $\epsilon_{\mathrm{L}} \to 0$ limit—hold as long as the noise has sub-exponential tails. This can be understood as (sufficiently) independent repetitions of a calculation in this case suppress logical error rates exponentially, faster than is achievable by increasing the resource utilization of individual gates under this resource–reliability trade-off.

4.4 Concluding remarks

We have devised a method to perform an asymptotic analysis of the number of additional gates required by fault-tolerant constructions compared to the non-fault-tolerant circuits from which they are derived. Using this accounting, we guide the design of resource-efficient fault-tolerant circuits based on the resource–reliability trade-off of the basic unit.

In Section 4.2, we first formalized the notion of a fault-tolerant construction. Key to our definition was the restriction to constant-depth error correcting circuits (Definition 32iii), which is in contrast to the recursive concatenation-based approaches whose overheads have previously been studied. In concatenation-based constructions, the fault-tolerant circuit is constructed through recursive application of the FT_n of Definition 32 at the level of the logical gate [NC10; Imp04; Tha+05; Svo+05] — usually for fixed n, e.g. $n = 3$ in the recursive triple modular redundancy constructions of [Imp04; Tha+05]. While the concatenation-based approach is useful for the analysis of asymptotic thresholds through intermediate pseudothreshold, e.g. using a flow-map model [Svo+05], it conflates what we have termed the depth and width overheads since concatenation increases both simultaneously. For the analysis at fixed and finite ϵ_P where we are satisfied to accept pseudothresholds (instead of true asymptotic thresholds), we may fix the depth D of the error correcting circuit while nonetheless increasing code size n thereby decreasing the logical error rate ϵ_L. This subtly in Definition 32 allowed us to interpret the number overhead as coming from three distinct components: the width overhead, the depth overhead, and the dependency overhead (Section 4.2.1). While the width and the depth overheads are most readily analyzed using fault-tolerant formulas (Section 4.2.2), the dependency overhead captures the deviation from the formula resulting from the realities of the circuit model and was numerically estimated using inference techniques on the Bayesian networks induced by the noisy circuit

(Section 4.2.3).

In Section 4.3, we formally introduced the resource–reliability trade-off and stated the main asymptotic result (Remark 36) which depends crucially on the $\epsilon_{\mathrm{P}} \to 0$ behavior of the resource–reliability curve. We studied the regions in the fault-tolerant construction parameter space in which fault-tolerance may be preferred based on assumptions about the finite-ϵ_{P} behavior of the resource–reliability function. We show three qualitatively different results: in the case where $W(\epsilon_{\mathrm{P}}) = \Theta(\log(1/\epsilon_{\mathrm{P}}))$, the existence of a region of fault-tolerant resource savings is asymptotically independent of logical error rate ϵ_{L} and depends solely on constant factors (Section 4.3.1); if $W(\epsilon_{\mathrm{P}}) = o(\log(1/\epsilon_{\mathrm{P}}))$, the existence of a region of fault-tolerant resource savings may be possible for relatively large values of ϵ_{L} depending on finite-ϵ_{P} behavior (Section 4.3.2); and if $W(\epsilon) = \omega(\log(1/\epsilon))$, then the fault-tolerant construction always preferred for sufficiently small ϵ_{L} (Section 4.3.3).

Beyond the repetition-code— The repetition code has been the subject of most inquiry into classical fault-tolerant constructions, with little attention given to the study of other codes [Neu56; HW91; Pip88; ES03]. This is in stark contrast with the study of fault-tolerant models of quantum computation, where the development and analysis of new codes is a primary research thrust [CTV17]. The repetition code has largely been unchallenged in the classical setting in part due to matching negative results showing that the repetition code is sufficient to yield the optimal threshold error rate for commonly studied gate sets [EP98; ES03]. Furthermore, the repetition code's simplicity yields both constant-depth (transversal) logical operations and constant-depth error correction, both of which are all the more important in the classical setting as no error-free computation is allowed; in contrast, the typical setting for fault-tolerant quantum schemes assumes access to noiseless classical computation. However, the analysis of fault-tolerance overheads, particularly the constant factor analysis, may provide another way to compare fault-tolerant constructions in which other codes may be more favorable than the repetition code.

Notably, our restriction to bounded depth error correction circuits also limits us to codes which can be corrected locally, i.e. only observing a constant number of input wires [Yek+12]. This self-imposed restriction may be lifted by loosening Definition 32iii to allow the error correcting circuit's depth to grow with the code size at the cost of introducing ϵ_{L}-dependency to the depth overhead and affecting the asymptotic result of Remark 36.

Fault-tolerance in other systems— Returning to the motivation for von Neumann's original work [Neu56]: biological systems perform noisy information processing and seems natural to ask whether they take advantage from some form of fault-tolerance. While the highly-structured von Neumann construction for fault-tolerance, i.e. alternation of computation and error correction, may be unrealistic in most systems, general principles may hold and may indeed be preferred due to more favorable resource efficiency or reliability. For example, error correction mechanisms have been discovered in biology [For81; SF11; Bat19], and some studies have looked into fault-tolerance of biologically-inspired network models of computation [Zlo+22]. Signatures of error correction and fault-tolerance have also been observed to emerge in noisy Boolean networks made to undergo evolution-like dynamics [MFC23]. Given the ubiquity of error correction in biology, might it be possible that there are also signatures of fault-tolerance? Resource efficiency may be one way to probe the development of both error correction and fault-tolerance in biological systems.

In addition to naturally occurring biological systems, there many engineered systems in which information processing occurs in a distributed and networked manner, e.g. electronic social networks and financial markets [Kir+08]. Information flow in these networks are subject to noise and resource constraints. Furthermore, many of these systems, such as in financial markets, the noise is often heavy-tailed [IIW15], which is precisely the scenario where fault-tolerant design may be preferred. Again, the highly-structured von Neumann construction for fault-tolerance may be unrealistic in these cases, but it may be the case that principles of fault-tolerant design may provide resource savings in the design of complicated networks.

Chapter 5

Connections to information theory

From its inception, the theory of fault-tolerance has had many deep ties with the information theory. Quantities such as computation capacity [WC63] have been defined in analogy with Shannon's channel capacity. More recently, information theoretic inequalities have been the basis of upper bounds on fault-tolerance [ES99; Kem+08]. In this Chapter, we study connections between fault-tolerance and information theory (Q3).

In Section 5.1[1], we study the problem of optimally designing an optimal encoding of a message over a noisy channel. While in previous chapters we considered only simple noise models, we now take on the task of encoding a random variable through an arbitrary noisy channel. Suppose, for example, that we are given access to a noisy channel $X \to Y$, where X and Y are discrete random variables. Our goal is to find encodings of $X \to Z$ which are maximally preserved under the noisy channel subject to a constraint on the information content in Z. We examine a particular quantification of this trade-off, the Deterministic Information Bottleneck trade-off, which can be viewed as a noisy compression algorithm. We apply our algorithm to a number of tasks: compressing the English alphabet, extracting informative color classes from natural images, and compressing a group theory-inspired dataset. Our algorithm extends previous work, being able to uncover convex regions

[1]This has appeared as "Pareto-optimal clustering with the primal deterministic information bottleneck," by Andrew K. Tan, Max Tegmark, and Isaac L. Chuang in Entropy (2022), 24, 771 [TTC22].

of the trade-off frontier. We demonstrate how the full structure of this frontier may be useful in the selection of a Pareto-optimal encoding.

In Section 5.2, we provide a review of existing upper bounds on fault-tolerance thresholds. Existing results are placed in context based on the high-level structure of their arguments and we review the results of Section 2.1 in this light. This Section is unpublished and contains no new results and appears mainly as a collection thoughts regarding the construction of tight negative results on fault-tolerance in circuits. The aim is to point towards the development of a new class of information-like quantities—through the review of existing fault-tolerance upper bounds—that are tailored to a particular model of noisy computation.

5.1 Pareto-optimal clustering

5.1.1 Introduction

Many important machine learning tasks can be cast as an optimization of two objectives that are fundamentally in conflict: performance and parsimony. In an auto-encoder, this trade-off is between the fidelity of the reconstruction and narrowness of the bottleneck. In the rate-distortion setting, the quantities of interest are the distortion, as quantified by a prescribed distortion function, and the captured information. For clustering, the trade-off is between intra-cluster variation and the number of clusters. While these problems come in many flavors—with different motivations, domains, objectives, and solutions—what is common to all such multi-objective trade-offs is the existence of a Pareto frontier, representing the boundary separating feasible solutions from infeasible ones. In a two-objective optimization problem, this boundary is generically a one-dimensional curve in the objective plane, representing solutions to the trade-off where increasing performance along one axis necessarily decreases performance along the other.

The shape of the frontier, at least locally, is important for model selection: prominent corners on the frontier are often more robust to changes in the inputs and therefore corre-

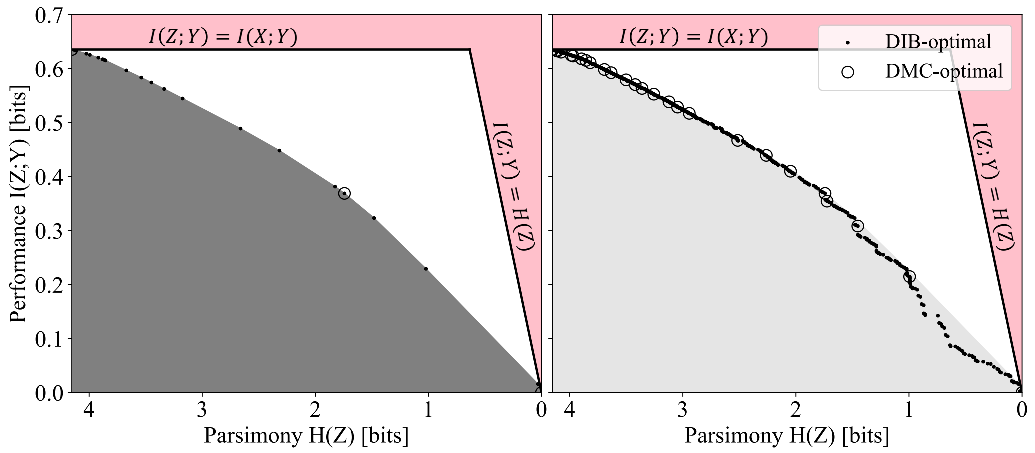


Figure 5.1: Comparison of the Lagrangian DIB (left) and primal DIB (right) frontiers discovered by Algorithm 1 for the English alphabet compression task discussed in Section 5.1.3. The shaded regions indicate the convex hull of the points found by the Pareto Mapper algorithm. Clusterings inside the shaded region, while Pareto optimal, are not optimal in the Lagrangian formulation.

spond to more desirable solutions. The global frontier can provide additional insights, such as giving a sense of interesting scales in the objective function. Structure often exists at multiple scales; for hierarchical clustering problems, these are the scales at which the data naturally resolve. Unfortunately, much of this useful structure (see Fig. 5.1) is inaccessible to optimizers of the more commonly studied convex relaxations of the trade-offs. Optimization over discrete search spaces poses a particular difficulty to convex relaxed formulations, as most points on the convex hull are not feasible solutions, and Pareto optimal solutions are masked by the hull. While the optimization of the Lagrangian relaxation is often sufficient for finding a point on or near the frontier, we, in contrast, seek to map out the entire frontier of the trade-off and therefore choose to tackle the primal problem directly.

We focus on the general problem of the deterministic encoding of a discrete domain. For a finite set of inputs, X, which we identify with the integers $[X] \equiv \{1, \ldots, |X|\}$, we seek a mapping to a set Z, where $|Z| \leq |X|$. The search space is therefore the space of functions $f : [X] \to [Z]$, which we call "encodings" or equivalently, "hard clusterings", where $Z = f(X)$ is interpreted as the number of the cluster to which X is assigned. We evaluate the encodings using the Deterministic Information Bottleneck objective, but regardless of which objectives

Algorithm 1 Pareto Mapper: ε-greedy agglomerative search

Input: Joint distribution $X, Y \sim p_{XY}$, and search parameter ε
Output: Approximate Pareto frontier P

```
1:  procedure PARETO_MAPPER($p_{XY}, \varepsilon$)
2:      Pareto set $P = \emptyset$                              ▷ Initialize maintained Pareto set
3:      Queue $Q = \emptyset$                                   ▷ Initialize search queue
4:      Point $p = (\mathrm{x} = -\mathrm{H}(p_X), \mathrm{y} = \mathrm{I}(p_{X;Y}), \mathrm{f} = \mathrm{id})$     ▷ Evaluate trivial clustering
5:      $P \leftarrow$ INSERT($p, P$)
6:      $Q \leftarrow$ ENQUEUE(id, $Q$)    ▷ Start with identity clustering $\mathrm{id} : [n] \to [n]$ where $n = |X|$
7:      while $Q$ is not $\emptyset$ do
8:          $f =$ DEQUEUE($Q$)
9:          $n = |\,\mathrm{range}(f)|$
10:         for $0 < i < j < n$ do                   ▷ Loop over all pairs of output clusters of $f$
11:             $f' = c_{i,j} \circ f$                          ▷ Merge clusters $i, j$ output $f$
12:             Point $p = \mathrm{Point}(\mathrm{x} = -\mathrm{H}(p_{f'(X)}), \mathrm{y} = \mathrm{I}(p_{f'(X);Y}), \mathrm{f} = f')$
13:             $d =$ PARETO_DISTANCE($p, P$)
14:             $P \leftarrow$ PARETO_ADD($p, P$)  ▷ Keep track of point and clustering in Pareto set
15:             with probability $e^{-d/\varepsilon}$, $Q \leftarrow$ ENQUEUE($f', Q$)
16:
        return $P$
```

are chosen, we will refer to all two-objective optimization problems over the space of such functions f as “clustering problems”.

The goal of this paper is to motivate the study of the Pareto frontiers to primal clustering problems and to present a practical method for their discovery.

Objectives and relation to prior work

We focus on the task of lossy compression, which is a trade-off between retaining the salient features of a source and parsimony. Rate-distortion theory provides the theoretical underpinnings for studying lossy data compression of a source random variable X into a compressed representation Z [CT06]. In this formalism, a distortion function quantifies the dissatisfaction with a given encoding, which is balanced against the complexity of the compressed representation as measured by $I(Z;X)$. In the well-known Information Bottleneck (IB) problem [TPB00], the goal is to preserve information about a target random variable Y as measured by the mutual information $I(Z;Y)$; the IB can be viewed as a rate-distortion problem with the Kullback–Leibler divergence, $D_{KL}(p_{Y|X}||p_{Y|Z})$, serving as the measure of distortion. In recent years, a number of similar bottlenecks have also been proposed inspired by the IB problem [SS17; AF18; Fis20]. We focus on one of these bottlenecks known as the Deterministic Information Bottleneck (DIB) [SS17].

The Deterministic Information Bottleneck

In the DIB problem, we are given random variables X and Y with joint probability mass function (PMF) p_{XY}, and we would like to maximize $I(Z;Y)$ subject to a constraint on $H(Z)$. As in [SS17], we further restrict ourselves to the compression of discrete domains, where X, Y and Z are finite, discrete random variables. We note that DIB-optimal encodings are deterministic encodings $Z = f(X)$ [SS17], and we can therefore focus on searching through the space of functions $f : [X] \to [Z]$, justifying the interpretation of DIB as a clustering problem. Since the optimization is being performed over a discrete domain in this case, not all points along the frontier are achievable. Nonetheless, we define the Pareto frontier piecewise as the curve that takes on the minimum vertical value between any two adjacent points on the frontier.

Formally, given p_{XY}, the DIB problem seeks an encoding $f^* : X \to Z$ such that, $Z^* = f^*(X)$ maximizes the relevant information captured for a given entropy limit H^*:

$$f^*_{\text{primal}} \equiv \underset{f:H[f(X)]\leq H^*}{\arg\max} \; I\left(Y; f(X)\right). \tag{5.1}$$

We will refer to the constrained version of the DIB problem in Eq. (5.1) as the *primal* DIB problem, to differentiate it from its more commonly studied Lagrangian form [SS17]:

$$f^*_{\text{Lagrangian}} \equiv \arg\max_{f} I\left(f(X); Y\right) - \beta H\left(f(X)\right). \tag{5.2}$$

In this form, which we call the *Lagrangian* DIB, a trade-off parameter β is used instead of the entropy bound H^* to parameterize f_* and quantify the importance of memorization relative to parsimony. The Lagrangian relaxation has the benefit of removing the non-linear constraint at the cost of optimizing a proxy to the original function, known as the DIB Lagrangian, but it comes at the cost of being unable to access points off the convex hull of the trade-off. We note that while we use the terminology 'primal DIB' to differentiate it from its Lagrangian form, we do not study its 'dual' version in this paper.

Many algorithms have been proposed for optimizing the IB, and more recently, the DIB objectives [HWD17]. An iterative method that generalizes the Blahut-Arimoto algorithm was proposed alongside the IB [TPB00] and DIB [SS17] algorithms. For the hierarchical clustering of finite, discrete random variables X and Y using the IB objective, both divisive [PTL93] and agglomerative [ST99] methods have been studied. Relationships between geometric clustering and information theoretic clustering can also be used to optimize the IB objective in certain limits [Ban+05]. More recently, methods using neural network-based variational bounds have been proposed [Ale+17]. However, despite the wealth of proposed methods for optimizing the (D)IB, past authors [HWD17; TPB00; SS17; Ale+17; Sti20; And+04] has focused only on the Lagrangian form of Eq. (5.2) and are therefore unable to find convex portions of the frontier.

Frontiers of the DIB Lagrangian and primal DIB trade-offs are contrasted in Fig. 5.1, with the shaded gray region indicating the shroud that the optimization of the Lagrangian relaxation places on the frontier (the particular frontier presented is discussed in Section 5.1.3). Points within the shaded region are not accessible to the Lagrangian formulation of the problem as they do not optimize the Lagrangian. We also note that while the determinicity of solutions is a consequence of optimizing the Lagrangian DIB [SS17], the convex regions of the primal DIB frontier is known to contain soft clusterings [KTV18; TW19]. In our work, the restriction to hard clusterings can be seen as a part of the problem statement. Finally, we adopt the convention of flipping the horizontal axis as in [TW19] which more closely matches the usual interpretation of a Pareto frontier where points further up and to the right are more desirable.

Discrete Memoryless Channels

A closely related trade-off is that between $I(Z;Y)$ and the number of clusters $|Z|$, which has been extensively studied in the literature on the compression of discrete memoryless channels (DMCs) [KY14; ZK16; HWD17]. In Fig. 5.1 and the other frontier plots presented in Section 5.1.3, the DMC optimal points are plotted as open circles. The DIB and DMC trade-offs are similar enough that they are sometimes referred to interchangeably [HWD17]: some previous proposals for solutions to the IB [ST99] are better described as solutions to the DMC trade-off. We would like to make this distinction explicit, as we seek to demonstrate the richness of the DIB frontier over that of the DMC frontier.

Pareto Front Learning

In recent work by Navon et al. [Nav+20], the authors define the problem of Pareto Front Learning (PFL) as the task of discovering the Pareto frontier in a multi-objective optimization problem, allowing for a model to be selected from the frontier at runtime. Recent hypernetwork-based approaches to PFL [Nav+20; Lin+20] are promising being both scalable and in principle capable of discovering the entirety of the primal frontier. Although we use a different approach, our work can be seen as an extension to the existing methods for PFL to discrete search spaces. Our Pareto Mapper algorithm performs PFL for the task of hard clustering, and our analysis provides evidence for the tractability of the PFL in this setting.

We also note similarities to the problems of Multi-Task Learning and Multi-Objective Optimization. The main difference between these tasks and the PFL setup is the ability to select Pareto-optimal models at runtime. We direct the reader to [Nav+20], which provides a more comprehensive overview of recent work on these related problems.

Motivation and Objectives

Our work is, in spirit, a generalization of [TW19], which demonstrated a method for mapping out the primal DIB frontier for the special case of binary classification (i.e., $|Y| = 2$). Although we deviate from their assumptions, assuming that X is discrete (rather than continuous in [TW19]), and being limited to deterministic encodings (rather than stochastic ones in [TW19]), and thus our results are not strictly comparable, our goal of mapping out the primal Pareto frontier is done in the same spirit.

The most immediate motivation for mapping out the primal Pareto frontier is that its shape is useful for model selection: given multiple candidate solutions, each being near the frontier, we would often like to be able to privilege one over the others. For example, one typically favors the points that are the most robust to input noise, that is, those that are most separated from their neighbors, appearing as concave corners in the frontier. For the problem of geometric clustering with the Lagrangian DIB, the angle between piecewise linear portions of its frontier, known as the "kink angle", has been proposed as a criterion for model selection [SS19]. Using the primal DIB frontier, we can use distance from the frontier as a sharper criterion for model selection; this is particularly evident in the example discussed in Section 5.1.3, where the most natural solutions are clearly prominent in the primal frontier but have zero kink angle. The structure of this frontier also encodes useful information about the data. For clustering, corners in this frontier often indicate scales of interests: those at which the data best resolve into distinct clusters. Determining these scales is the goal of hierarchical clustering.

Unlike the previously studied case of binary classification [TW19], no polynomial time algorithm is known for finding optimal clusterings for general $|Y| > 2$ [ZK16]. Finding an optimal solution to the DIB problem (i.e., one point near the frontier) is known to be equivalent to k-means in certain limits [SB04; SS19], which is itself an NP-hard problem [Awa+15]: mapping out the entirety of the frontier is no easier. More fundamentally, the number of possible encoders is known to grow super-exponentially with $|X|$; therefore, it is not known whether we can even hope to store an entire DIB frontier in memory. Another issue is that of the generalization of the frontier in the presence of sampling noise. Estimation of mutual information and entropy for finite datasets is known to be a difficult problem with many common estimators either being biased, having high variance, or both [NSB02; Pan03; KSG04; NBR04; Poo+19]. This issue is of particular significance in our case as a noisy point on the objective plane can mask other, potentially more desirable, clusterings.

It is these gaps in the optimization of DIB and DIB-like objectives that we seek to address. Firstly, existing work on optimization concerns itself only with finding a point on or near the frontier. These algorithms may be used to map out the Pareto frontier, but they need to be run multiple times with special care taken in sampling the constraint in order to attain the desired resolution of the frontier. Furthermore, we observe empirically that almost all of the DIB Pareto frontier is in fact convex. The majority of the existing algorithms applicable to DIB-like trade-offs optimize the Lagrangian DIB [Ale+17; SS17] and are therefore unable to capture the complete structure of the DIB frontier. Existing agglomerative methods [ST99] are implicitly solving for the related but distinct DMC frontier, which has much less structure than the DIB frontier. Finally, existing methods have assumed access to the true distribution p_{XY} or otherwise used the maximum likelihood (ML) point estimators [SS17; TW19], which are known to be biased and have high variance for entropy and mutual information, which can have a significant effect on the makeup of the frontier.

Roadmap

The rest of this paper is organized as follows. In Section 5.1.2, we tackle the issue of finding the Pareto frontier in practice by proposing a simple agglomerative algorithm capable of mapping out the Pareto frontier in one pass (Section 5.1.2) and propose a modification that can be used to select robust clusterings with quantified uncertainties from the frontier when the sampling error is significant (Section 5.1.2). We then present our analytic and experimental results in Section 5.1.3. In Section 5.1.3, we provide evidence for the sparsity of the Pareto frontier, giving hope that it is possible to efficiently study it in practice. To demonstrate our algorithm and build intuition for the Pareto frontier structure, we apply it to three different DIB optimization problems in Section 5.1.3: compressing the English alphabet, extracting informative color classes from natural images, and discovering group–theoretical structure. Finally, in Section 5.1.4, we discuss related results and directions for future work.

5.1.2 Methods

We design our algorithm around two main requirements: firstly, we would like to be able to optimize the primal objective of Eq. (5.1) directly, thereby allowing us to discover convex portions of the frontier; secondly, we would like a method that records the entire frontier in one pass rather than finding a single point with each run. While the task of finding the exact Pareto frontier is expected to be hard in general, Theorem 7 applied to Example 39, gives us hope that the size of the Pareto frontier grows at most polynomially with input size $|X|$. As is often the case when dealing with the statistics of extreme values, we expect that points near the frontier are rare and propose a pruned search technique with the hope that significant portions of the search space can be pruned away in practice. In the spirit of the probabilistic analysis provided above, we would like an algorithm that samples from a distribution that favors near-optimal encoders, thereby accelerating the convergence of our search. For this reason, we favor an agglomerative technique, with the intuition that there are good encoders that can be derived from merging the output classes of other good encoders. An agglomerative approach has the additional benefit of being able to record the entire frontier in one pass. For these reasons, we propose an agglomerative pruned search algorithm for mapping the Pareto frontier in Section 5.1.2. We also describe in Section 5.1.2 a modification of the algorithm that can be applied to situations where only a finite number of samples are available.

The Pareto Mapper

Our method, dubbed the *Pareto Mapper* (Algorithm 1), is a stochastic pruned agglomerative search algorithm with a tunable parameter ϵ that controls the search depth. The algorithm is initialized by enqueuing the identity encoder, $f = \text{id}$, into the search queue. At each subsequent step (illustrated in Fig. 5.2), an encoder is dequeued. A set containing all of the Pareto-optimal encoders thus far encountered is maintained as the algorithm proceeds. All encoders that can be constructed by merging two of the given encoder's output classes (there are $O(n^2)$ of these) are evaluated against the frontier of encoders seen so far; we call encoders derived this way child encoders. If a child encoder is a distance d from the current frontier, we enqueue its children with probability $e^{-d/\epsilon}$ and discard it otherwise, resulting in a search over an effective length-scale ϵ from the frontier. The selection of ϵ tunes the trade-off between accuracy and search time: $\epsilon = 0$ corresponds to a greedy search that does not evaluate sub-optimal encodings, and $\epsilon \to \infty$ corresponds to a brute-force search over all possible encodings. As the search progresses, the Pareto frontier is refined, and we are able to prune a larger majority of the proposed encoders. The output of our algorithm is a Pareto set of all found Pareto optimal clusterings of the given trade-off. The Pareto frontier at any given moment is stored in a data structure, called a *Pareto set*, which is a list of point structures. A point structure, p, contains fields for both objectives $p.\text{x}$, $p.\text{y}$, and optional fields for storing the uncertainties $p.\text{dx}$, $p.\text{dy}$ and clustering function $p.\text{f}$. The Pareto set is maintained so that the Pareto-optimality of a point can be checked against a Pareto set of size m in $\Theta(\log m)$ operations. Insertion into the data structure requires in the worst case $\Theta(m)$ operations, which is optimal, as a new point could dominate $\Theta(m)$ existing points necessitating their removal. We define the distance from the frontier as the minimum Euclidean distance that a point would need to be displaced before it is Pareto-optimal, which also requires in the worst case $\Theta(m)$ operations. A list of pairs $(H(Z), I(Z;Y))$, sorted by its first index, provides a simple implementation of the Pareto set. The pseudocode for important auxiliary functions such as PARETO_ADD and PARETO_DISTANCE is provided in

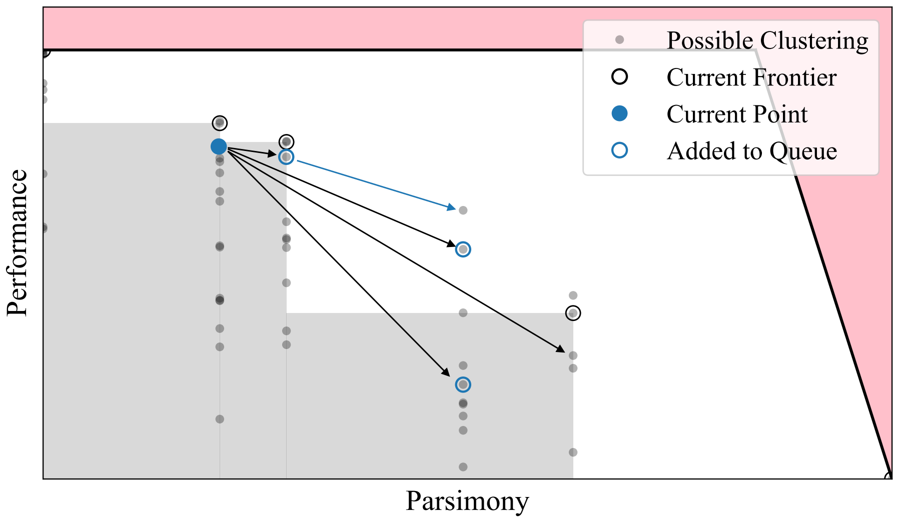
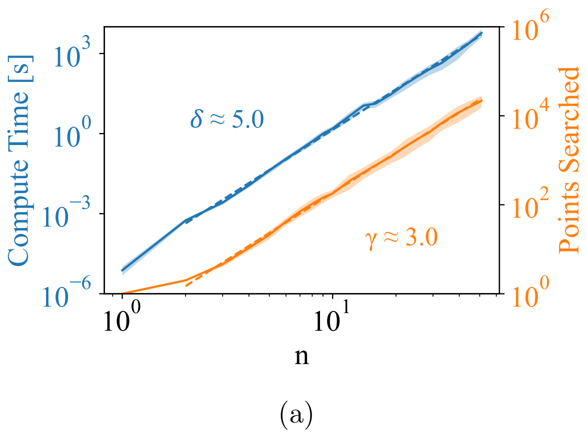


Figure 5.2: Illustration of one step in the main loop of the Pareto Mapper (Algorithm 1) mid-run. For pedagogical purposes, all possible encoders (filled gray circle) are plotted on the objective plane. The Pareto optimal points searched so far are marked with open black circles, and the region of the objective plane they dominate is shaded in gray. The black arrows show neighboring encoders and newly enqueued encoders are marked by open blue circles; encodings that are optimal with respect to the current frontier are enqueued with certainty and sub-optimal encodings enqueued with probability $e^{-d/\epsilon}$, where d is the distance from the frontier. Note that some Pareto optimal points are only accessible through sub-optimal encoders (blue arrow).



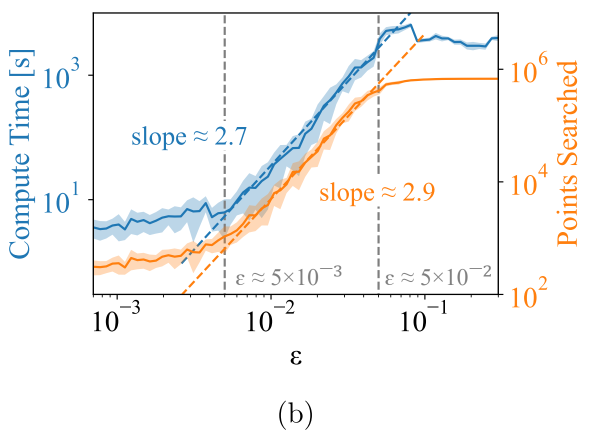
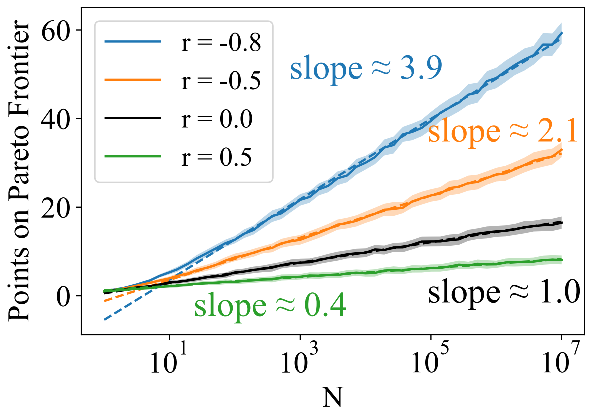


Figure 5.3: Scaling of computation time (left scale) and points searched (right scale) as a function of (a) input size n at $\epsilon = 0$, where we find compute time scales as $O(n^\delta)$ and the size of the Pareto set scales as $O(n^\gamma)$; (b) search parameter ϵ for randomly generated p_{XY} of fixed size.

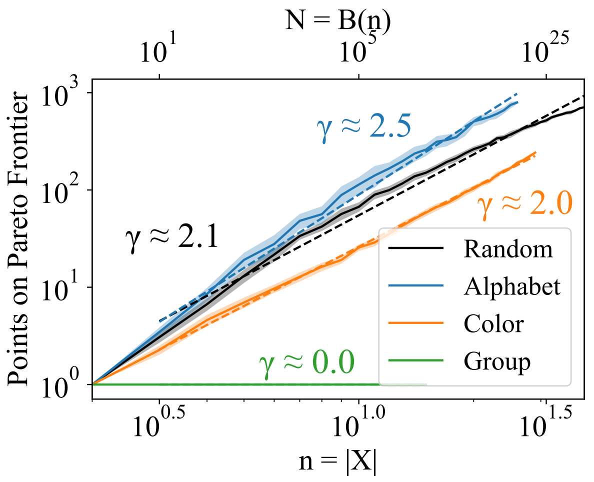
Appendix C.2.

Although we have provided evidence for polynomial scaling of size of the Pareto set, it is not obvious if the polynomial scaling of the Pareto set translates to the polynomial scaling of our algorithm, which depends primarily on how quickly the search space can be pruned away by evaluation against the Pareto frontier. To demonstrate the polynomial scaling of our algorithm with n, we evaluate the performance of the Pareto Mapper on randomly generated p_{XY}. Since $\epsilon \to \infty$ corresponds to a brute-force search, and therefore has no hope of having polynomial runtime, we focus on the $\epsilon \to 0$ case; we show later, in Section 5.1.3, that $\epsilon \to 0$ is often sufficient to achieve good results. For Fig. 5.3a, we randomly sample p_{XY} uniformly over the simplex of dimension $|X||Y| - 1$ varying $|X|$ with fixed $|Y| = 30$. We find that the scaling is indeed polynomial. Comparing with the scaling of the size of the Pareto set shown in Fig. 5.4b, we see that approximately $O(n)$ points are searched for each point on the Pareto frontier. While the computation time, empirically estimated to be $\Theta(n^{5.0})$, is limiting in practice, we note that it is indeed polynomial, which is sufficient for our purposes.

We also evaluate the scaling of our algorithm with ϵ. Again, we randomly sample p_{XY} uniformly over the simplex of dimension $|X||Y| - 1$, this time fixing $|X| = 11$ and $|Y| = 30$, with results plotted in Fig. 5.3b. We find that the relevant scale for distances is between



(a)



(b)

Figure 5.4: Scaling of number of points on the Pareto frontier (a) as a function of $N = |S|$ for bivariate Gaussian distributed (U, V) with specified correlation $r \equiv \sigma_{UV}/\sigma_U \sigma_V$, and (b) for the DIB problem with input size n where we find $|\,\text{Pareto}(S)| = O(n^\gamma)$.

$\epsilon_- \approx 5 \times 10^{-3}$ and $\epsilon_+ \approx 5 \times 10^{-2}$; while the specifics of the characteristic range for ϵ depends on the dataset, we empirically find that while ϵ_+ remains constant, ϵ_- decreases as n increases. This is consistent with the fact that as n increases, the DIB plane becomes denser, and the average separation between points decreases. This would suggest that there is an n above which the runtime of the Pareto Mapper exhibits exponential scaling for any $\epsilon > 0$. In the absence of noise, one can run the Pareto Mapper at a number of different values of ϵ evaluating precision and recall with respect to the combined frontier to evaluate convergence (see Fig. 5.8b). We discuss how to set ϵ in the presence of noise due to sampling in Section 5.1.3.

Robust Pareto Mapper

So far, we have assumed access to the true joint distribution p_{XY}. Normally, in practice, we are only provided samples from this distribution and must estimate both objective functions, the mutual information $I(Z;Y)$ and entropy $H(Z)$, from the empirical distribution $\hat{p}_{XY}$. Despite the uncertainty in these estimates, we would like to find clusterings that are Pareto optimal with respect to the true distribution. Here, we propose a number of modifications to Algorithm 1 that allow us to quantify our uncertainty about the frontier and thereby produce a frontier that is robust to sampling noise. The modified algorithm, dubbed the *Robust Pareto Mapper* (Algorithm 2), is described below.

Given samples from the joint distribution, we construct the empirical joint distribution and run the Pareto Mapper (Algorithm 1) replacing the entropy and mutual information functions with their respective estimators. We use the entropy estimator due to Nemenman, Shafee, and Bialek (NSB) [NSB02], as it is a Bayesian estimator that provides not only a point estimate but also provides some bearing on its uncertainty, although another suitable estimator can be substituted. We find that our method works well even with point estimators, in which case resampling techniques (e.g., bootstrapping) are used to obtain the uncertainty in the estimate. After running the Pareto Mapper, points that are not significantly different from other points in the Pareto set are removed. This filtering operation considers points in order of ascending uncertainty, as measured by the product of its standard deviations in $H(Z)$, and $I(Z;Y)$. Subsequent points are added as long as they do not overlap with the confidence interval in either H or I with a previously added point, and they are removed otherwise. There is some discretion in choosing the confidence interval, which we have chosen empirically to keep the discovered frontier robust between independent runs. This filtering step is demonstrated in Section 5.1.3.

Algorithm 2 Robust Pareto Mapper: dealing with finite data

Input: Empirical joint distribution $\hat{p}_{XY}$, search parameter ε, and sample size S
Output: Approximate Pareto frontier P with uncertainties

```
1: procedure ROBUST_PARETO_MAPPER($\hat{p}_{XY}, \varepsilon$)
2:     Pareto set $P \leftarrow$ PARETO_MAPPER($\hat{p}_{XY}, \epsilon$)    ▷ Run PARETO_MAPPER with suitable estimators
3:     Pareto set $P' = \emptyset$    ▷ Initialize set of robust encoders
4:     for $p \in P$ do    ▷ This step can be skipped if an interval estimator is used above
5:         ($p$.dx, $p$.dy) $\leftarrow$ RESAMPLE($p, \hat{p}_{XY}$)    ▷ Store uncertainty of points on frontier
6:     for $p \in P$ in ascending order of uncertainty do
7:         if $p$ is significantly different than all $q \in P'$ then
8:             $P' \leftarrow$ PARETO_ADD($p, P'$)    ▷ Filter with preference for points with low variance
       return $P'$
```

5.1.3 Results

General properties of Pareto frontiers

Before introducing the specifics of the DIB problem, we would like to understand a few general properties of the Pareto frontier. The most immediate challenge we face is the size of our search space. For an input of size $|X|$, the number of points on the DMC frontier is bounded by $|X|$, but there is no such limit on the DIB frontier. Given the combinatorial growth of the number of possible clusterings with $|X|$, it is not immediately clear that it is possible to list all of the points on the frontier, let alone find them. If we are to have any chance at discovering and specifying the DIB frontier, it must be that DIB-optimal points are sparse within the space of possible clusterings, where sparse is taken to mean that the size of the frontier scales at most polynomially with the number of items to be compressed.

In this section, we provide sufficient conditions for the sparsity of the Pareto set in general and present a number of illustrative examples. We then apply these scaling relationships to the DIB search space and provide numerical evidence that the number of points grows only polynomially with $n \equiv |X|$ for most two-objective trade-off tasks.

Argument for the Sparsity of the Pareto Frontier— First, we will formally define a few useful terms. Let $S = \{\vec{s_i}\}_{i=1}^{N}$ be a sample of N independent and identically distributed (i.i.d.) bivariate random variables representing points $\vec{s_i} = (U_i, V_i)$ in the Pareto plane.

Definition 37. A point $(U, V) \in S$ is *maximal* with respect to S, or equivalently called *Pareto*-optimal, if $\forall (U_i, V_i) \in S$, $\exists V_i > V \implies U > U_i$. In other words, a point is maximal with respect to S if there are no points in S both above and to its left.

Definition 38. For a set of points $S \subset \mathbb{R}^2$, the *Pareto set* $\text{Pareto}(S) \subseteq S$ is the largest subset of S such that all $(U, V) \in \text{Pareto}(S)$ are maximal with respect to S.

Now, we can state the main theorem of this section, which we prove in Appendix C.1.

Theorem 7. *Let $S = \{(U_i, V_i)\}_{i=1}^N$ be a set of bivariate random variables drawn i.i.d. from a distribution with Lipschitz continuous CDF $F(u, v)$, and invertible marginal CDFs F_U, F_V. Define the region*

$$R_N \equiv \left\{(u, v) \in [0, 1] \times [0, 1] : u + v - C(u, v) \geq e^{-\frac{1}{N}}\right\}, \tag{5.3}$$

where $C(u, v)$ denotes the copula of (U_i, V_i), which is the function that satisfies $F(u, v) = C(F_U(u), F_V(v))$.

Then, if the Lebesgue measure of this region $\lambda(R_N) = \Theta\left(\frac{\ell(N)}{N}\right)$ as $N \to \infty$, we have $\mathbb{E}\left[|\operatorname{Pareto}(S)|\right] = \Theta(\ell(N))$.

Example 39. Let us consider the case of independent random variables with copula $C(u, v) = uv$. Note that in this case, the level curves $u + v - C(u, v) = e^{-\frac{1}{N}}$ are given by $v = \frac{e^{-\frac{1}{N}} - u}{1-u}$. We can then integrate to find the area of the region R_N

$$\lambda(R_N) = 1 - \int_0^{e^{-\frac{1}{N}}} \frac{e^{-\frac{1}{N}} - u}{1 - u} du = e^{-1/n}\left(1 - e^{1/n}\right)\left(\log\left(1 - e^{-1/n}\right) - 1\right). \tag{5.4}$$

Expanding for large N, we find that $\lambda(R_N) = \frac{\log N}{N} + O(N^{-1})$. We see that this satisfies the conditions for Theorem 7 with $\ell(N) = \log N$, giving $\mathbb{E}_S\left[|\operatorname{Pareto}(S)|\right] = \Theta(\log N)$.

Additional examples can be found in Appendix C.1. Numerically, we see that for independent random variables U and V, the predicted scaling holds even down to relatively small N; furthermore, the linear relationship also holds for correlated Gaussian random variables U, V (Fig. 5.4a). The logarithmic sparsity of the Pareto frontier allows us to remain hopeful that it is possible, at least in principle, to fully map out the DIB frontier for deterministic encodings of discrete domains despite the super-exponential number of possible encoders.

Dependence on Number of Items $|X|$— The analysis above gives us hope that Pareto-optimal points are generally polylogarithmically sparse in $N \equiv |S|$, i.e., $|\operatorname{Pareto}(S)| = O((\log N)^\gamma)$. Of course, the scenario with which we are concerned is one where the random variables $U = -H(Z)$ and $V = I(Z;Y)$; the randomness of U and V in this case comes from the choice of encoder $f : X \to Z$ which, for convenience, we assume is drawn i.i.d. from some distribution over the space of possible encodings. Note that conditioned on the distribution of X and Y, the points $(H(f(X)), I(f(X);Y))$ are indeed independent, although the agglomerative method we use to sample from the search space introduces dependence; however, in our case, this dependence likely helps the convergence of the algorithm.

In the DIB problem, and clustering problems more generally, we define $n \equiv |X|$ to be the size of the input. The search space is over all possible encoders $f : X \to Z$, which has size $N = B(n)$ where $B(n)$ are the Bell numbers. Asymptotically, the Bell numbers grow super-exponentially: $\ln B(n) \sim n \ln n - n \ln \ln n$, making an exhaustive search for the frontier intractable. We provide numerical evidence in Fig. 5.4b that the sparsity of the Pareto set holds in this case, with its size scaling as $O(\text{poly}(n))$, or equivalently, $O(\text{polylog}(N))$. While in the worst case, all B_n clusterings can be DIB-optimal (the case where $(p_{XY})_{ij} = \text{diag}(\vec{r})$ for r_i drawn randomly from the $(n-1)$-dimensional simplex results in clusterings with strict negative monotonic dependence on the DIB plane, and therefore all points are DIB-optimal, see Appendix C.1), our experiments show that in practice, the size of the Pareto frontier (and compute time) grows polynomially with the number of input classes n (Fig. 5.3), with the degree of the polynomial depending on the details of the joint distribution $p(x, y)$. This result opens up the possibility of developing a tractable heuristic algorithm that maps out the Pareto frontier, which we will explore in the remainder of this paper.

At the Pareto frontier: three vignettes

The purpose of this section is to demonstrate our algorithm and illustrate how the primal DIB frontier can be used for model selection and to provide additional insights about the data. To this end, we apply our algorithm to three different DIB optimization tasks: predicting each character from its predecessor in the English text, predicting an object from its color, and predicting the output of a group multiplication given its input. In all cases, the goal is to find a representation that retains as much predictive power as possible given a constraint on its entropy. We will describe the creation of each dataset and motivate its study. For all three tasks, we discuss the frontier discovered by our algorithm and highlight informative points on it, many of which would be missed by other methods either because they are not DMC-optimal or because they lie within convex regions of the frontier. For the task of predicting the subsequent English character, we will also compare our algorithm to existing methods including the Blahut–Arimoto algorithm, and a number of geometric clustering algorithms; we will also use this example to demonstrate the Robust Pareto Mapper (Algorithm 2).

Compressing the English Alphabet

First, we consider the problem of compressing the English alphabet with the goal of preserving the information that a character provides about the subsequent character. In this case, we collect 2 gram statistics from a large body of English text. Treating each character as a random variable, our goal is to map each English character X into a compressed version Z while retaining as much information as possible about the following character Y.

Our input dataset is a 27×27 matrix of bigram frequencies for the letters a–z and the space character, which we denote "_" in the figures below. We computed this matrix from the 100 Mb *enwiki8* Wikipedia corpus after removing all symbols except letters and the space character, removing diacritics, and making all letters lower-case.

The Pareto frontier is plotted in Fig. 5.5 and the points corresponding to some interesting clusterings on the frontier are highlighted. We find that from 2 gram frequencies alone, the DIB-optimal encodings naturally discover relevant linguistic features of the characters, such as their grouping into vowels and consonants.

The DMC-optimal encoding corresponding to a cluster size of $k = 2$ is nearly balanced (i.e., $H(Z) = \log_2 2$) and separates the space character and the vowels from most of the consonants. However, in contrast to the binary classification case of $|Y| = 2$ studied in [TW19], the DMC-optimal encodings are far from balanced (i.e., $H(Z) \approx \log_2(k)$) for larger k. We

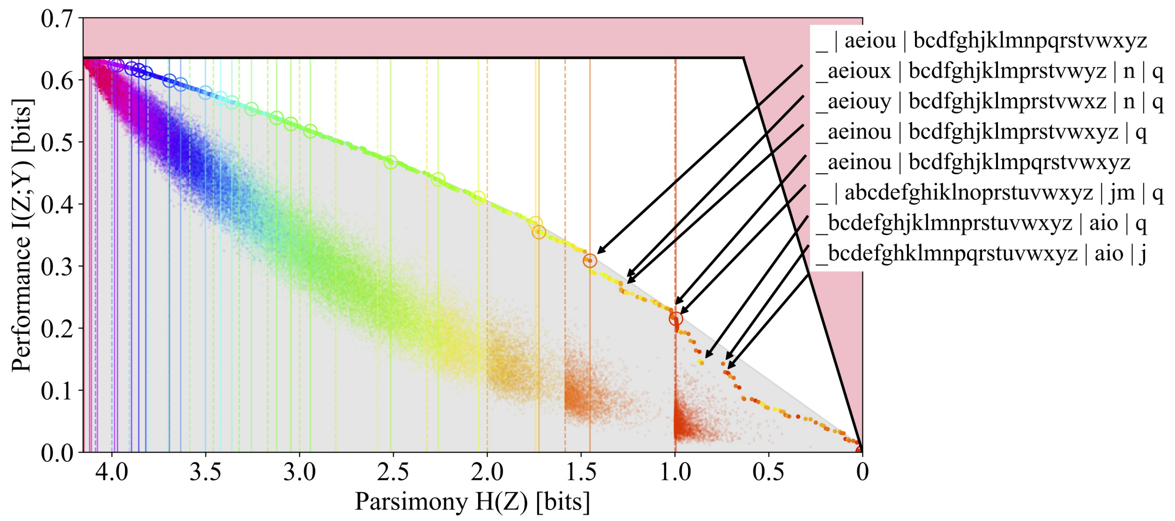


Figure 5.5: The primal DIB frontier of the English alphabet is compressed to retain information about the subsequent character. Points in the shaded gray region, indicating the convex hull, are missed by optimization of the DIB Lagrangian. Encodings corresponding to interesting features of the frontier are annotated, and DMC-optimal points are circled. Dotted vertical lines mark the location of balanced clusters (i.e., $H(Z) = \log_2 k$ for $k \in \{2, 3, \ldots\}$), and solid vertical lines correspond to the entropy of the DMC-optimal encodings. A sample of encodings drawn uniformly at random for each $|Z|$ is evaluated on the plane, illustrating the large distance from the frontier for typical points in the search space. The color indicates $|Z|$.

note that on the DIB frontier, the most prominent corners are often not the DMC-optimal points, which are circled. By looking at the corners and the DMC-optimal points near the corners, which are annotated on the figure, we discover the reason for this: distinguishing anomalous letters such as ‘q’ has an outsized effect on the overall information relative to its entropy cost. These features are missed when looking only at DMC-optimal points, because although the 2 gram statistics of ‘q’ are quite distinct (it is almost always followed by a ‘u’), it does not occur frequently enough to warrant its own cluster when our constraint is cluster count rather than entropy. In other words, ‘q‘ is quite special and noteworthy, and our Pareto plot reveals this while the traditional DMC or DIB plots do not.

The frontier is seen to reveal features at multiple scales, the most prominent corner corresponding to the encoding that separates the space character, ‘_’, from the rest of the alphabet, and the separation of the vowels from (most of the) consonants. The separation of ‘q‘ often results in a corner of a smaller scale because it is so infrequent. These corners indicate the natural scales for hierarchical clustering. We also note that a large majority of the points, including those highlighted above, are below the convex hull (denoted by the solid black line) and are therefore missed by algorithms that optimize the DIB Lagrangian.

A random sample of clusterings colored by $|Z|$ is also plotted on the DIB plane in Fig. 5.5; a sample for each value of $|Z| = \{1, \ldots, |X|\}$ is selected uniformly at random. We see that there is a large separation between the typical clustering, sampled uniformly at random, and the Pareto frontier, indicating that a pruned search based on the distance from the frontier, such as the Pareto Mapper of Algorithm 1, is likely able to successfully prune away much of the search space. A better theoretical understanding of the density could provide further insights on how the runtime scales with ϵ.

We now compare the results of the Pareto Mapper (Algorithm 1) with other clustering methods. We first use the Pareto Mapper with $\epsilon = 0$ to derive a new dataset from the original alphabet dataset (which has $|X| = 27$) by taking the $|Z| = 10$ clustering with the highest mutual information. We are able to obtain a ground truth for this new dataset with

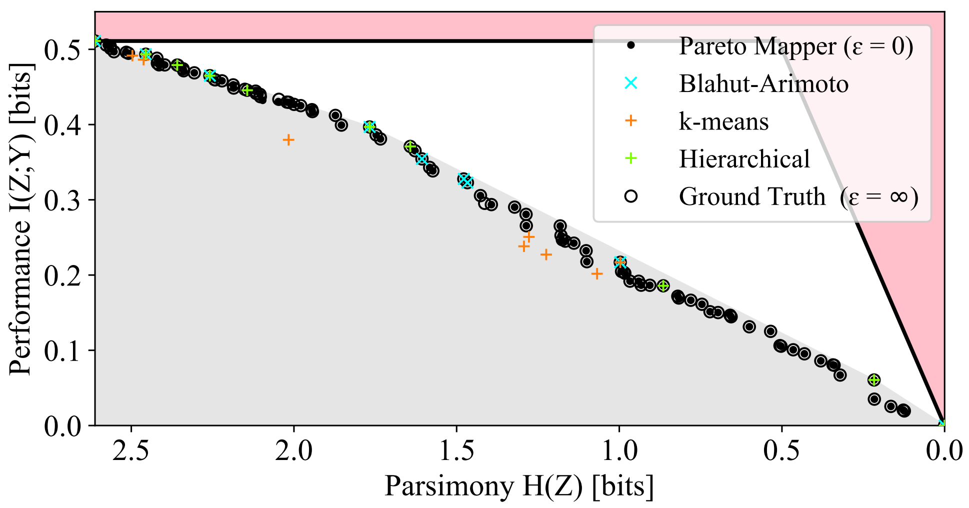


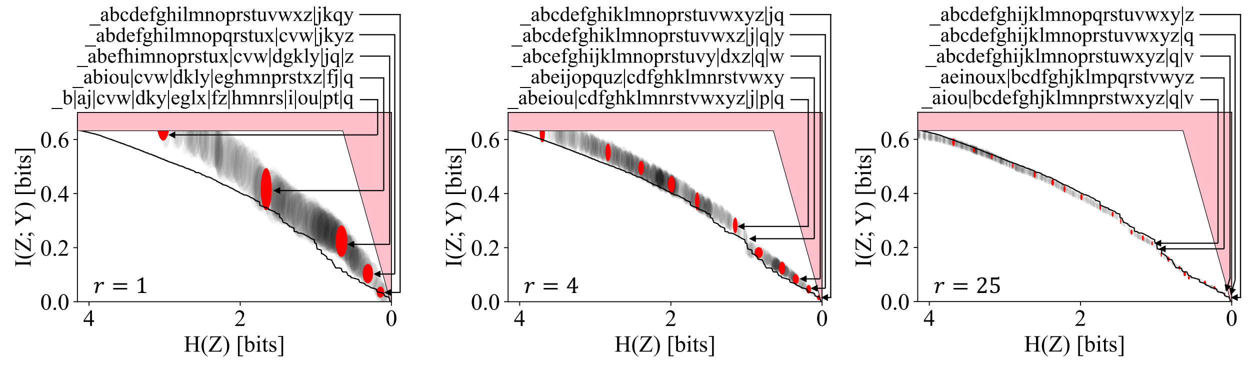
Figure 5.6: Comparison of the Pareto Mapper and other classification algorithms with ground truth for $|X| = 10$. The true Pareto frontier is calculated with a brute force search over all $B(10) = 115,975$ clusterings f.

$|X| = 10$ using a brute force search, against which we compare the other methods. These methods are compared on the DIB plane in Fig. 5.6 and in tabular form in Table 5.1. Notably, all the encodings found by the Blahut–Arimoto algorithm used in [TPB00; SS17] are DIB-optimal, but as it optimizes the DIB Lagrangian, it is unable to discover the convex portions of the frontier. We also compare our algorithm to geometric clustering methods where we assign clusters pairwise distances according to the Jensen–Shannon distances between the conditional distributions $p(Y|X = x_i)$. These methods perform poorly when compared on the DIB plane for a number of reasons: firstly, some information is lost in translation to a geometric clustering problem, since only pairwise distances are retained; secondly, the clustering algorithms are focused on minimizing the number of clusters and are therefore unable to find more than n points. Additionally, these geometric clustering algorithms, while similar in spirit, are not directly optimizing the DIB objective.

Table 5.1: Comparison of the performance of Algorithm 1 with other clustering algorithms. Here, a true positive (TP) is a point that is correctly identified as being Pareto optimal by a given method; false positives (FP) and false negatives (FN) are defined analogously.

Method	**Points**	**TP**	**FP**	**FN**	**Precision**	**Recall**
Ground truth ($\epsilon \to \infty$)	94	94	0	0	1.00	1.00
Pareto Mapper ($\epsilon = 10^{-2}$)	94	94	0	0	**1.00**	**1.00**
Pareto Mapper ($\epsilon = 0$)	91	88	3	6	0.97	0.94
Hierarchical (average)	10	7	3	87	0.70	0.07
Hierarchical (single)	10	10	**0**	84	**1.00**	0.11
Hierarchical (Ward)	10	7	3	87	0.70	0.07
k-means (JSD)	10	3	7	91	0.30	0.02
k-means (wJSD)	10	2	8	92	0.20	0.10
Blahut Arimoto	9	9	**0**	85	**1.00**	0.10

To demonstrate the Robust Pareto Mapper (Algorithm 2), we create a finite sample $\hat{n}_{XY} = s\hat{p}_{XY}$ from a multinomial distribution with parameter p_{XY} and s trials. To quantify the sample size in natural terms, we define the sampling ratio $r \equiv s/2^{H(X,Y)}$. The results of the Robust Pareto Mapper on the alphabet dataset for several sampling ratios are shown in Fig. 5.7. We note that even for relatively low sampling ratios, the algorithm is able to extract interesting information; it is able to quickly separate statistically distinct letters such as 'q' and identify groups of characters such as vowels. As the sampling ratio increases, the Robust Pareto Mapper identifies a larger number of statistically significant clusterings (marked in red) from the rest of the discovered frontier (marked in gray). It is also notable that uncertainties in the entropy are typically lowest for encodings that split X into roughly equally probably classes; that these clusters are preferred is most readily seen in the highlighted clustering with $H(Z) \approx 1$ in Fig. 5.7b. We can see from these plots that, especially for low sampling ratios, the estimated frontier often lies above that of the true p_{XY} (solid black line). This is expected, as estimators for mutual information are often biased high. Despite this,

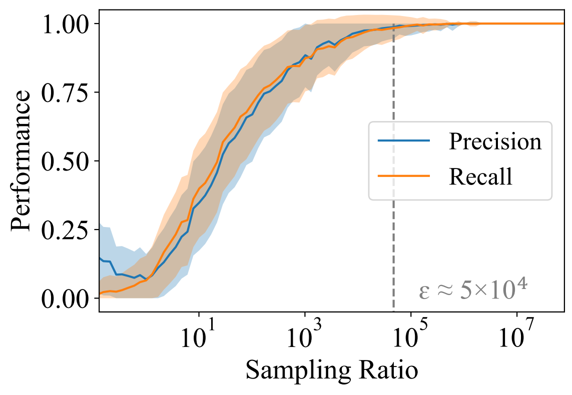


(a) (b) (c)

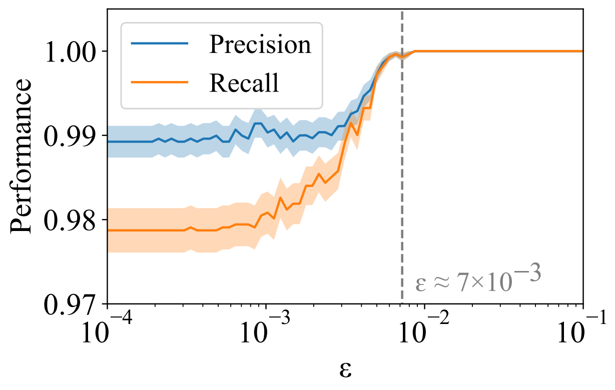
Figure 5.7: The optimal frontier discovered by the Robust Pareto Mapper at various sampling ratios. The points corresponding to robust clusterings selected by the algorithm are highlighted in red, with the rest in gray. The true frontier is shown in solid black.

the true frontier is found to lie within our estimates when the variance of the estimators is taken into account even for modest sampling ratios, as seen in the plot for $r = 4$.

Finally, we would like to comment on choosing the parameter ϵ in Algorithm 1 and Algorithm 2 when working with limited sample sizes. The uncertainty in the frontier due to finite sampling effects naturally sets a scale for choosing ϵ. Ideally, we want the two length scales—that given by ϵ, and that due to the variance in the estimators—to be comparable. This ensures that we are not wasting resources fitting sampling noise. Evaluating the performance as a function of sample size and epsilon, we see that often, sample size is the limiting factor even up to significant sampling ratios, and often, a small ϵ is often sufficient. This is demonstrated in Fig. 5.8, where it can be seen that performance is good even with small ϵ, and increasing ϵ does not result in a more accurate frontier until the sampling ratio is greater than $r \approx 5 \times 10^4$. In practice, determining the appropriate ϵ can be accomplished by selecting different holdout sets, and running the algorithm at a given ϵ in each case; when ϵ is chosen appropriately, the resulting Pareto frontier should not vary significantly between the runs.



(a)



(b)

Figure 5.8: Performance as a function of (a) sample size, and (b) ϵ. The precision and recall are measured relative to the true frontier obtained by a brute force search on the true distribution.

Naming the Colors of the Rainbow

Human languages often have a small set of colors to which words are assigned, and they remarkably often settle on similar linguistic partitions of the color space despite cultural and geographic boundaries [Two+21]. As our next example, we apply our method to the problem of optimally allocating words to colors based on the statistics of natural images. In order to cast this as a DIB-style learning problem, we consider the goal of being able to identify objects in natural images based solely on color: the variable we would like to predict, Y, is therefore the class of the object (e.g., apple or banana). The variable we would like to compress, X, is the average color of the object. The Pareto-optimal classifiers are those that, allotted limited memory for colorative adjectives, optimally draw the boundaries to accomplish the task of identifying objects. We demonstrate some success in discovering different color classes, relate it to those typically found in natural languages, and discuss shortcomings of our method.

We create a dataset derived from the COCO dataset [Lin+14], which provides a corpus of segmented natural images with 91 object classes. There are a number of challenges we immediately face in the creation of this dataset, which require us to undertake a number of preprocessing steps. Firstly, using standard RGB color values, with 8 bits per channel, leaves over 16 million color classes to cluster, which is not feasible using our technique. Secondly, RGB values contain information that is not relevant to the task at hand, as they vary with lighting and image exposure. Thus, we turn to the HSV color model and use only the hue value (since hue is a circular quantity, we use circular statistics when discussing means and variances), which we refer to as the color of the object from now on. This leaves 256 values which are further reduced by contiguous binning so that each bin has roughly equal probability in order to maximize the entropy of X. After this preprocessing, we are left with an input of size $|X| = 30$. Another challenge we face is that there are often cues in addition to average color when performing object identification such as color variations, shape, or contextual understanding of the scene; in order to obtain the cleanest results, we retain only those classes that could reasonably be identified by color alone. Specifically, for the roughly 800,000 image segments from the approximately 100,000 images we considered in the COCO dataset, we calculate the average color of each segment and keep only the 40% with the most uniform color as measured by the variance of the hue across the segment; then, looking across classes, we keep only those that are relatively uniform on the average color of its instances, keeping approximately the most uniform 20% of classes. We are left with a dataset of approximately 80,000 objects across $|Y| = 18$ classes, predictably including rather uniformly colored classes such as apples, bananas, and oranges. We chose these cutoff percentiles heuristically to maximize the predictive power of our dataset while maintaining a sizable number of examples.

The Pareto frontier for this dataset is shown in Fig. 5.9. A number of DMC-optimal points are circled, and their respective color palettes are plotted below in descending order of likelihood. First, we note that the overall amount of relevant information is quite low, with a maximum $I(X;Y) \approx 0.12$, indicating that despite our preprocessing efforts, color is not a strong predictor of object class. Unlike the other Pareto frontiers considered, there are a few prominent corners in this frontier, which is a sign that there is no clear number of

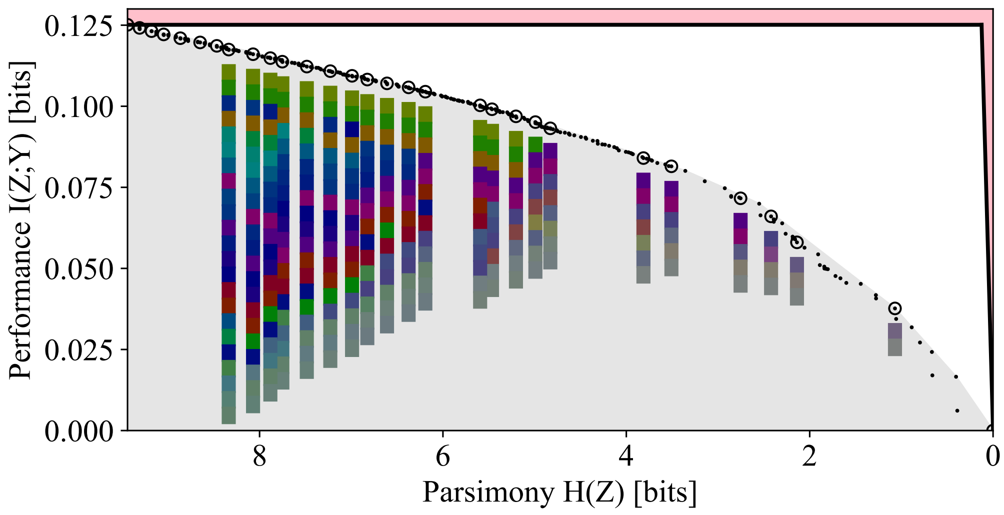


Figure 5.9: Pareto frontier of color data. A representative color patch for each cluster is shown below select points sorted by likelihood. The saturation of the patch represents the likelihood-weighted variance of the colors mapped to the class.

colors to best resolve the spectrum. For the first few DMC-optimal clusterings, the colors fall broadly into reddish-purples, greens, and blues. This is somewhat consistent with the observation that languages with limited major color terms often settle for one describing warm colors and one describing colder colors [Two+21].

Overall, the results are not conclusive. We will address a few issues with our method and discuss how it might be improved. Firstly, as noted by [Two+21], the colors present in human languages often reflect a communicative need and therefore should be expected to depend strongly on both the statistics of the images considered and also the prediction task at hand. Since the COCO dataset was not designed for the purpose of learning colors, classes had color outliers, despite our preprocessing efforts, which reduced the classification accuracy by color alone. Using color as a predictor of the variety of an apple or as a predictor of the ripeness of a banana might yield better results (see Fig. 5.10); indeed, these tasks might be more reflective of the communicative requirements under which some human languages developed [Two+21]. Due to the scarcity of relevant datasets, we have not attempted to address these subtleties. Another issue, more fundamental to the DIB algorithm, is that DIB is not well suited for the compression of domains of a continuous nature. The DIB trade-off naturally

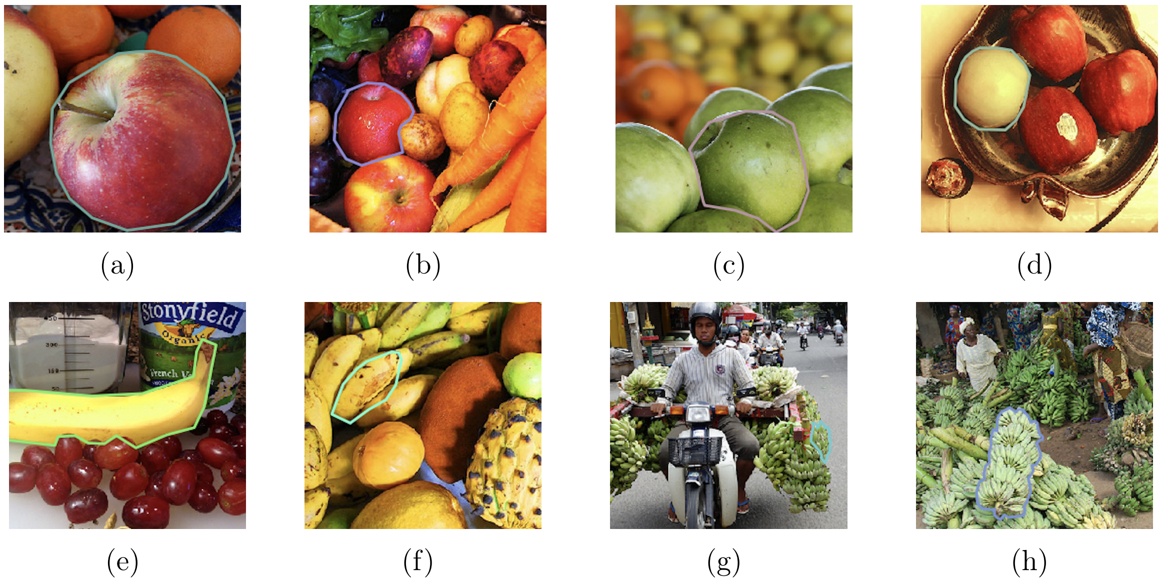


Figure 5.10: Examples of correctly (a, b) and incorrectly (c, d) identified apples; and correctly (e, f) and incorrectly (g, h) identified bananas from the filtered COCO dataset using the best discovered five-bin clustering.

favors a discrete domain, X, without a measure of similarity between objects in X. Unlike the other examples considered, the space of colors is inherently continuous: there is some notion of similarity between different hues. One weaknesses of the DIB trade-off is that it does not respect this natural notion of closeness and it is as likely to map distant hues together as it is ones that are close together. This is undesirable in the case of the color dataset, as we would ideally like to map contiguous portions of the color space to the same output. Other objectives, such as the IB or a multidimensional generalization of [TW19], may be more suitable in cases where the domain is of a continuous nature.

Symmetric Compression of Groups

For our final example, we turn our attention to a group–theoretic toy example illustrating a variation on the compression algorithm so far considered which we call "symmetric compression." We consider a triplet of random variables (X_1, X_2, Y), each taking on values in the set G with the special property that G forms a group under the binary group operation '$\cdot$'. We could apply Algorithm 1 directly to this problem by setting $X = (X_1, X_2)$, but this is not ideal, as it does not make use of the structure we know the data to have and as a result needlessly expands our search space. Instead, we make the slight modification, detailed in Appendix C.3, where we apply the same clustering to both inputs, $Z = (f(A), f(B))$. We would like to discover an encoding f that trades off the entropy of the encoding with the ability to predict Y from $(f(X_1), f(X_2))$. We expect that the DIB frontier encodes information about the subgroups of the group G, but we also expect to find points on the frontier corresponding to near-subgroups of G.

We consider two distributions. The first consists of the sixteen integers that are coprime to 40, i.e., $\{1, 3, 7, 9, 11, 13, 17, 19, 21, 23, 27, 29, 31, 33, 37, 39\}$, which for a multiplicative group module 40 denoted $(\mathbb{Z}/40\mathbb{Z})^\times$. The second is the Pauli group G_1, whose elements are the sixteen 2×2 matrices generated by the Pauli matrices under matrix multiplication: they are the identity matrix I and Pauli matrices X, Y, Z, each multiplied by ± 1 and $\pm i$. These groups are chosen as they both have order 16 but are otherwise quite different; notably, $(\mathbb{Z}/40\mathbb{Z})^\times$ is abelian while G_1 is not. The joint probability distribution is defined as follows for each group G: we take (X_1, X_2) to be distributed uniformly over G^2 and $Y = X_1 \cdot X_2$. The distribution $p_{X_1 X_2 Y}$ is given as input to the symmetric Pareto Mapper (Algorithm 6).

The resultant frontiers are shown in Fig. 5.11. As expected, the subgroups are readily identified in both cases, as seen the in circled points on the frontier with entropy $H(Z) = 1$, $H(Z) = 2$, and $H(Z) = 3$, corresponding to subgroups of size 2, 4, and 8, respectively. In this example, we also see that the clusterings corresponding to the subgroups saturate the feasibility bound of $I(Z; Y) = H(Z)$, indicating that at these points, all the information captured in Z is relevant to Y. At these points, the encoding effectively identifies a subgroup $H \leq G$ and retains information only about which of the $|G|/|H|$ cosets an element belongs to; as it retains the identity of the cosets of X_1 and X_2 in Z_1 and Z_2, it is able to identify the coset of the output Y, thereby specifying Y to $\log_2 \frac{|G|}{|H|}$ bits. These clearly desirable solutions show up prominently in the primal DIB frontier, yet their prominence is not evident on the frontier of the Lagrangian DIB—notably having zero kink angle as defined by [SS19].

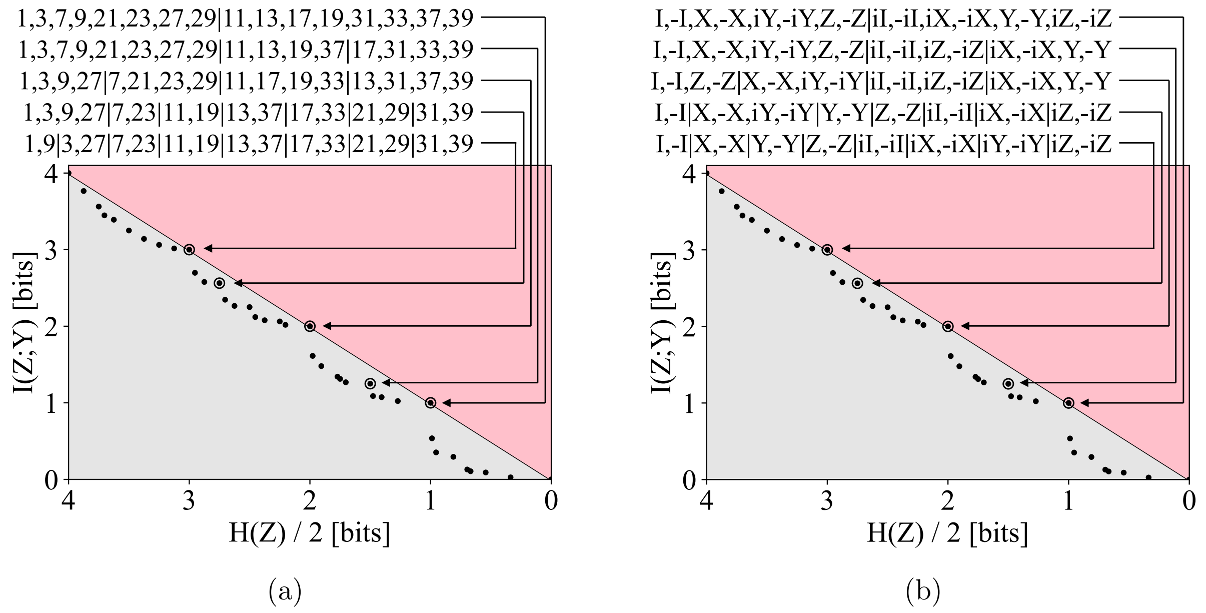


Figure 5.11: Discovered frontier of the (a) $(\mathbb{Z}/40\mathbb{Z})^\times$ group and (b) the non-abelian Pauli group. Both groups have identical frontiers despite having different group structures.

In addition to the points corresponding to identified subgroups, a number of intermediary points have also been highlighted showing 'near-subgroups', where, allotted a slightly larger entropy budget, the encoder can further split cosets apart in such a way that partial information is retained. Interestingly, despite being very different groups, they have identical Pareto frontiers. This is because they both have subgroups of the same cardinality, and the entropy and relevant information of these encodings is agnostic to the group theoretic details and concerns itself only with the ability to predict the result of the group operation.

5.1.4 Concluding remarks

We have presented the Pareto Mapper algorithm, which computes the optimal trade-off between parsimony and performance for lossy data compression. By applying it to examples involving linguistics, image colors and group theory, we have demonstrated the richness of the DIB Pareto frontier that customarily lies hidden beneath the convex hull. Our English alphabet example revealed features at multiple scales and examples of what the frontier structure reveals about the data, and we demonstrated a modification to our algorithm that can aid model selection given significant sampling noise. Notably, we showed how the prominence of a point on the primal frontier can be a sharper tool for model selection than existing measures on the Lagrangian DIB frontier; for example, for our group theory examples, it outperformed the kink-angle method for model selection, which only gave kink angles of zero.

Our result helps shed light on recently observed phases transitions. Recent work has shown that learning phase transitions can occur when optimizing two-objective trade-offs including the (D)IB [AS18; Fis20; Wu+20; NS21] and β-VAEs [RV18]. In these cases, it is found that the performance of the learned representation makes discontinuous jumps as the trade-off parameter β is varied continuously. Such phase transitions can be readily understood in terms of the primal Pareto frontier of the trade-off: methods that optimize the Lagrangian DIB are only able to capture solutions on the convex hull of objective plane; as the Pareto frontier is largely convex, methods that optimize the Lagrangian exhibit will discontinuous jumps when the trade-off parameter β (which corresponds to the slope of a tangent to the frontier) is varied. This is analogous to the way first-order phase transitions in statistical physics arise, where it is the closely related Legendre–Fenchel dual that is minimized.

We would like to emphasize that, going beyond the IB framework, our basic method (Section 5.1.2) is generally applicable to a large class of two-objective optimization problems, including general clustering problems. Specifically, our method can be adapted for two-objective trade-offs with the following properties: a discrete search space; a frontier that, for typical datasets, grows polynomially with the input size $|X|$; and a notion of relatedness between objects in the search space (e.g., for the DIB problem, new encodings can be derived from existing ones by merging its output classes), which allows for an agglomerative search. The modification (Section 5.1.2) can also be adapted given suitable estimators for other two-objective trade-offs.

Outlook

There are many opportunities to further improve our results both conceptually and practically. To overcome the limitations we highlighted with our image color dataset, it will be interesting to generalize our work and [TW19] to compressing continuous variables potentially with trade-offs such as the IB. While our evidence for the polynomial scaling of the size of the Pareto frontier is likewise applicable to other trade-offs of this sort, the runtime of our algorithm depends heavily on how quickly the search space can be pruned away and therefore is not guaranteed to be polynomial. Here, there is ample opportunity to tighten our analysis of the algorithmic complexity of finding the DIB frontier and on the scaling of generic Pareto frontiers.

Proofs aside, it will also be interesting to optimize the algorithm runtime beyond simply showing that it is polynomial. Although we have demonstrated the polynomial scaling of our algorithm for realistic datasets, the polynomial is of a high degree for our implementation, placing limits of $|X| \leq 50$ in practice. There are fundamental lower bounds on the runtime set by the scaling of the Pareto set, which we have shown in Fig. 5.4b to be approximately $O(n^{2.1})$ for realistic datasets; however, there is likely to be some room for reducing the runtime by sampling clusterings from a better distribution. Another opportunity for improvement is increasing the speed at which a given point can be evaluated on the objective plane, which is evidenced by the gap between the runtime, approximately $O(n^{5.0})$, and the number of points searched, $O(n^{3.0})$ (Fig. 5.3a).

While our method is only applicable to trade-offs over discrete search spaces, the Pareto frontier over continuous search spaces can also fail to be (strictly) concave. For example, the inability for the Lagrangian formulation of the (D)IB to explore all points on the trade-off has previously been studied in [KTV18]. They propose a modification to the (D)IB Lagrangian that allows for the exploration of parts of the frontier that are not strictly concave. An interesting direction for future work is to study whether a similar modification to the Lagrangian can be used to discover the convex portions of similar trade-offs, including those over discrete spaces. Another direction for future work is to compare the primal DIB frontier with solutions to the IB; while solutions to the DIB Lagrangian often perform well on IB plane [SS17], it is an open question whether the solutions to the primal DIB perform favorably. Finally, as pointed out by a helpful reviewer, the dual problem corresponding to the primal problem of Eq. (5.1), being a convex optimization problem, is also an interesting direction for future study.

We would also like to note that Pareto-pruned agglomerative search is a generic strategy for mapping the Pareto frontiers of two-objective optimization problems over discrete domains. The Pareto Mapper algorithm can also be extended to work in multi-objective settings given an appropriate implementation Pareto set in higher dimensions. We conjecture that the poly-logarithmic scaling of the Pareto set holds in higher dimensions as well. Extending this work to multi-objective optimization problems is another interesting direction for future work.

In summary, multi-objective optimization problems over discrete search spaces arise naturally in many fields from machine learning [SB04; TZ15; Ale+17; SS19; SS17; CS18; Sax+19] to thermodynamics [Sti20] and neuroscience [BM10]. There will therefore be a multitude of interesting use cases for further improved techniques that map these Pareto frontiers in full detail, including concave parts that reveal interesting structure in the data.

5.2 Negative fault-tolerance results

The previous Chapters have focused on positive fault-tolerance results. These results are based on constructive proofs following the highly structured von Neumann construction for fault-tolerance. One may wonder, however, whether other fault-tolerance schemes may exist. Of course, the space of possible circuit designs is vast; and furthermore, the analysis of probabilistic circuits is complicated by the intractability of calculating marginal distributions—formal constructive results as in [EP98; ES03] as well as our extension in Section 2.1 cir-

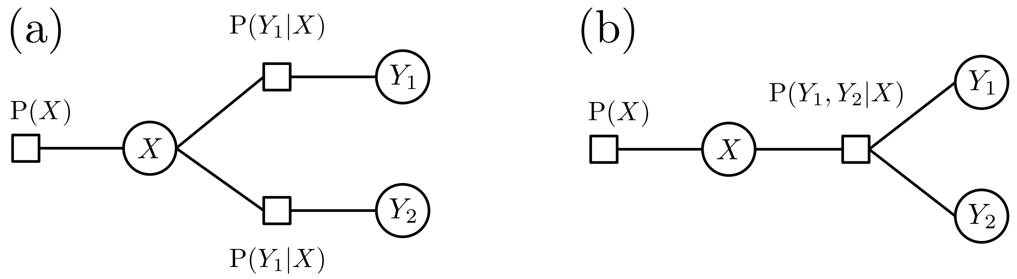


Figure 5.12: Factor graph representation of (a) formulas and (b) circuits for input signal X and wires Y_1 and Y_2 such that no path exists between Y_1 and Y_2.

cumvent this issue by working with formulas. Because of this, all known upper bounds are argued indirectly by placing a bound on the propagation of signals at individual wires or gates [Pip88; EP98; ES99; ES03; Ung07]. We will review these results and discuss opportunities in this Section.

It is tempting to adopt the picture of information as a fluid that is flows from the inputs of a circuit to its outputs with noise corresponding to the leaking of the fluid—here, we take "information" to mean a general measure of correlation with an input signal X. Taking this view, redundantly encoding signals and operating on them in a redundant manner, fault-tolerant constructions allow us to build better effective pipes and junctures; and the fault-tolerance threshold is precisely when the leaking of information can no longer be overcome by increased redundancy. There are three main things that need to be quantified to turn this intuition into a proof:

(P1) Quantify the amount of fluid, i.e. information about signal X, present in a pipe.

(P2) Quantify the rate at which the pipes leak; and

(P3) Quantify the amount of fluid required at the output to achieve a prescribed error rate.

This is a reasonable description in the case of formulas in particular, where the fluid combines at junctures as expected. For instance, consider the quantification of information by mutual information. Let X be some signal and let Y_1 and Y_2 be random variables which are inputs to a gate. Recalling the mutual information chain rule [CT06, Theorem 2.5.2], we find

$$I(Y_1, Y_2; X) = I(Y_1; X) + I(Y_2; X|Y_1), \tag{5.5}$$

$$= I(Y_1; X) + I(Y_2; X) - I(Y_1; Y_2; X), \tag{5.6}$$

where $I(Y_1; Y_2; X) \equiv I(Y_2; X) - I(Y_2; X|Y_1)$ is sometimes called the interaction information. For formulas, Y_1 and Y_2 are independent conditioned on X—see factor graph representation of Fig. 5.12a—so that $I(Y_1; Y_2; X) \geq 0$ and $I(Y_1, Y_2; X) \leq I(Y_1; X) + I(Y_2; X)$ as expected for a fluid. Information, however, is a strange fluid in the sense that mutual information $I(Y_1; Y_2; X)$ can be negative when Y_1 and Y_2 jointly contain more information about X than they do independently: the canonical example being that of random variables satisfying the relationship $Y_2 = X \oplus Y_1$. Similar behavior also occurs in other measures of information in circuits where random variables have the general factorization of Fig. 5.12b. For this reason, we must modify the picture of information combining at a juncture. A key observation of Evans and Schulman was the replacement of (P1) with the alternative [ES99]:

(P1') Quantify the amount of fluid, i.e. information about signal X, present in *any set* of pipes.

Instead of adopting the intuition of the fluid being localized to any particular pipe, (P1') treats the fluid as a quantity that exists in a collection of random variables. It turns out that this weaker picture is sufficient to prove negative fault-tolerance results [ES99].

In the remainder of this Section, we review existing negative results and discuss possibilities for new monotones and their implications for fault-tolerant computation and beyond. In our review, we use the modern treatment of the noisy circuit as a directed graphical model and interpret results in the language of strong data processing inequalities (SDPIs) [PW17]. Adopting the notation of [PW17], let $G = (\mathcal{V}, \mathcal{E})$ denote the directed acyclic graph (DAG) corresponding to a noisy circuit with q-ary random variables Y_v at each vertex $v \in \mathcal{V}$ (for most of our review $q = 2$). Each vertex corresponds either to the output of a gate, or an input or output of the circuit. Further, let $\mathrm{pa}(v)$ denote the parents of v. In general G admits a topological ordering of vertices corresponding to dependencies in the computation; for vertices $V_1, V_2 \in \mathcal{V}$ we use $V_1 > V_2$ to denote the fact that V_1 does not depend on V_2 in the sense that there is no path from V_1 to V_2.

In Section 5.2.1, we provide a review of negative fault-tolerance results based on the contraction of information at the wires which follows the prescription above. For formulas, these results use (P1), (P2), and (P3); they can also be adopted to circuits using the weaker (P1'). However, for both formulas and circuits, bounds based on information contraction at the wires are unable to provide tight negative bounds on fault-tolerance thresholds.

In Section 5.2.2, we review sharp negative fault-tolerance results which require analysis that is tailored to a set of gates. Here, we will introduce the alternative (P2') to capture information leaking at the gates. We interpret the negative results of [HW91; ES03] through the three-step process of (P1), (P2'), and (P3); the same can be done for our large alphabet positive results (Section 2.1) from which we can derive negative results which are tight for our specific choice of denoising gates. We also discuss negative fault-tolerance results based on the contraction of the lesser known *extractable information* [Ung10] and discuss their relationship with the tight negative results of [HW91; ES03]. Unfortunately, as these methods are based on (P1), their applicability is limited to formula-based computation.

Finally, we discuss opportunities to build upon these existing results with an eye towards tight negative results for fault-tolerant circuit-based computation in Section 5.2.3.

5.2.1 Information contraction at the wires

First, we provide a brief review of negative fault-tolerance by Evans and Schulman [ES99] where the information content is quantified by the mutual information and a negative fault-tolerance result is derived via (P1'), (P2), and (P3). Recall that in our model, computation is performed by noisy gates whose outputs are corrupted by binary symmetric noise.

The first part of the recipe (P1') involves the combination of information in a set of random variables. In particular, a bound is placed on the information between the source X and a set of wires $V \subset \mathcal{V}$ [ES99]. Using this approach, Evans and Schulman derive the following bound [ES99, Lemma 2]

$$I(V;X) \leq \sum_{\pi \text{ from } X \text{ to } W} \eta^{|\pi|}, \tag{5.7}$$

where the sum is over all paths π from X to $V \subset \mathcal{V}$ in G, $|\pi|$ denotes its length, and η denotes the rate at which mutual information decays at each wire defined more precisely below. The proof of Eq. (5.7) for the mutual information contraction coefficient relies on the data processing inequality [CT06, Theorem 2.8.1] and chain rule [CT06, Theorem 2.5.2] properties of mutual information. A related bound appears in [Pip88, Eq. 2] for formulas which takes the view of (P1).

Continuing on to (P2), Evans and Schulman demonstration that mutual information strictly decreases in the noisy binary symmetric channel [ES99]. Let X, Y and Z be Boolean random variables admitting factorization $X \to Y \to Z$. If $Z = \mathrm{BSC}_\epsilon(Y)$, where BSC_ϵ represents the binary symmetric channel with error parameter ϵ, then

$$I(X;Z) < \eta I(X;Y), \tag{5.8}$$

where for $\epsilon > 0 \implies \eta < 1$. The quantity η is known as the mutual information contraction coefficient of the channel $P_{Z|Y}$ [PW17]; and results of the form Eq. (5.8) are known as *strong data processing inequalities*. For the binary symmetric channel with error rate ϵ, Pippenger's proved $\eta = (1-2\epsilon)$ [Pip88] which was later tightened to $\eta = (1-2\epsilon)^2$ by Evans and Schulman [ES99, Theorem 1].

Finally, (P3) is obtained using Fano's inequality thereby obtaining the desired negative fault-tolerance result [Pip88; ES99]. In particular, consider the case of a circuit that depends essentially on one argument (i.e. there exists an assignment of all other inputs such that the output depends only on the unfixed argument). Let X denote the essential input variable and Z denote the circuit output fixing all other inputs. In this case, we can view the computation tree as a noisy communication channel and a direct application of Fano's inequality [CT06, Theorem 2.10.1] yields

$$1 + H_2(\delta) \leq I(F_m; X), \tag{5.9}$$

where H_2 is the binary entropy. Combining Eqs. (5.7) to (5.9) gives a bound on the output error rate as a function of the input error rate. Bounding the fan-in results in a bound on the threshold error rate.

Bounds that are in essence applications of Eqs. (5.7) and (5.8) for different noisy channels and different network topologies have proven remarkably useful for a number of different applications including applications to phase transitions in Ising models, percolation, and cellular automata among others [PW17; Mak19]. An unfortunate consequence of the generalizability of these information theoretic monotones is that they have so far been unable to produce tight bounds for fault-tolerance thresholds for any specific set of gates. To understand the limitations of these mutual-information-based monotones, it is worth reviewing the closely related problem of broadcasting on trees. In this problem, a signal X exists at the root of a tree and at each time step is sent to vertices of increasing depth via independent noisy channels; the goal is to reconstruct X from the signals at depth n. As $n \to \infty$, the probability that the reconstruction is correct has a sharp transition in the noise level. In this case of Boolean random variables subject to binary symmetric error of strength on a $(k + 1)$-regular tree ϵ, Evans et al. [Eva+00] proved a threshold of

$$\epsilon < \frac{1}{2} - \frac{1}{2\sqrt{k}}. \tag{5.10}$$

This is the same bound found by Evans and Schulman for the case of reliable computation in k-input gates [ES99]. Furthermore, Eq. (5.10) is sharp for the problem of broadcasting on trees [Eva+00]. Understanding the difference between these two problems in instructive for grasping the limitations of wire-based, i.e. those based on (P2), contraction bounds. The task of broadcasting on trees is simpler than that of reliable computation since it concerns itself only with the propagation of the signal whereas the computation task requires that signals can be generically manipulated: therefore the threshold for broadcasting on trees upper bounds that of reliable computation. In this sense, the wire-based contraction of Eq. (5.8) does not capture the processing that occurs at the gate; the broadcasting tree consists only of fan-in 1 gates that do not perform non-trivial processing. In fact, the bound of [ES99] is agnostic to the particular gate set which proves to be a limitation if tight upper bounds on fault-tolerance are to be desired.

5.2.2 Information contraction at the gates

As previously observed, negative fault-tolerance results will have to take into account the noisy processing being performed at each gate. Instead of quantifying the loss of information at the wires as in (P2), we should update the first part of the prescription to

(P2') Quantify the rate at which the junctures leak.

Here, information does not necessarily mean mutual information, as it is generally not true that mutual information will provide the tightest bounds. We use the term *extractable information*[2] to refer to any reasonable measure of correlations that provides a tight negative fault-tolerance bound; in particular, the signals should become indistinguishable as extractable information goes to zero so that we may define (P3) in analogy with Eq. (5.9). All known measures to date localize information to specific wires and therefore only apply to formulas. The general recipe for the negative fault-tolerance results discussed in this Section is (P1'), (P2), and (P3).

First, we review the tight negative results for reliable formula-based Boolean computation with gates of odd fan-in $k \geq 3$. The special case of $k = 3$ was first shown by Hajek and

[2]The term extractable information was first coined by Unger [Ung10].

Weller [HW91] and subsequently generalized to all odd $k \geq 3$ by Evans and Schulman [ES03]. We recast their negative result into the language of [Ung07]. Let $Y_1, \ldots, Y_k$ be the inputs to some gate and W be its noisy output. In this case, the information that any wire Y carries about X is quantified by the extractable information (P1), a possible definition of which is

$$\mathcal{EI}_{\mathrm{ES}}(Y;X) = |\Pr[Y=0|X=0] - \Pr[Y=0|X=1]|, \tag{5.11}$$

$$= \left\| P_{Y|X=0} - P_{Y|X=1} \right\|_{\mathrm{TV}}, \tag{5.12}$$

where X is some input signal, and $\|\cdot\|_{\mathrm{TV}}$ denotes the total variation (TV) distance. Starting with (P1'), the goal is to bound the extractable information contraction at each gate, i.e. to find θ such that

$$\mathcal{EI}(W;X) \leq \theta \max_{j\in[k]} \mathcal{EI}(Y_j;X). \tag{5.13}$$

Note that for a proper definition of θ we also need to maximize over distributions of X. The essence of the proof of the negative result in [HW91; ES03] is that above the threshold error rate, we have the strict inequality $\theta < 1$ and the extractable information decays geometrically with formula depth. Furthermore, this error rate is precisely the denoising threshold where the two stable fixed-points merge. By assumption, the extractable information admits a bound analogous to Fano's inequality for mutual information thereby giving (P3), giving a tight negative fault-tolerance result.

A more involved example is the original result of Unger [Ung07] which provides a tight upper bound on reliable formula-based Boolean computation with gates of $k = 2$ using a different definition of extractable information (P1)

$$\mathcal{EI}_{\mathrm{Ung}}(Y;X) = q(\Pr[Y=0]) \times \left\| P_{Y|X=0} - P_{Y|X=1} \right\|_{\mathrm{TV}}, \tag{5.14}$$

where

$$q(a) = \left(\frac{29}{2} + 2\sqrt{7}\right)\left(a - \frac{1}{2}\right)^4 + \left(\frac{5\sqrt{7}}{2} - \frac{13}{4}\right)\left(a - \frac{1}{2}\right)^2 - \frac{\sqrt{7}}{2} + \frac{73}{32}, \tag{5.15}$$

is known as the potential function [Ung07, Eq. 3]. Unger needed to update the definition of extractable information in the case of $k = 2$ as Eq. (5.13) does not provide a tight bound in this case. We emphasize that extractable information is not unique, in fact Unger subsequently shows that

$$q(a) = \frac{1}{(1-a)a + (11 - 4\sqrt{7})/18}, \tag{5.16}$$

[Ung10, Eq. 3] likewise provides a tight upper bound for the $k = 2$ case. Again it is possible to show (P3) thereby proving the tight negative result.

Note that Eq. (5.13) can also be used to prove a negative fault-tolerance result in our models of larger alphabet fault-tolerance. For example, the ultimate saddle-node bifurcation in Section 2.1.3 is precisely the point where $\theta < 1$. In the large alphabet case, though this bound is tight for the q-ary majority gate, it is not clear whether computation can be performed up to this point. As discussed in Section 2.1.3, the proof of fault-tolerance only works up to the point of the transcritical bifurcation which is strictly below the saddle-node bifurcation for $q \geq 3$.

Another related approach is that by Polyanskiy and Wu [PW17], who have developed a machinery for bounding the contraction coefficient η of a Bayesian network rather than a single channel. For a set of wires $V \subset \mathcal{V}$, the addition of a new wire $W \in \mathcal{V}$ such that $W > V$ yields the following bound on the contraction coefficient of the channel between X and $V \cup \{W\}$ [PW17, Theorem 5]:

$$\eta(P_{V,W|X}) \leq \eta_W \eta(P_{V,\mathrm{pa}(W)|X}) + (1 - \eta_W)\eta(P_{V|X}), \tag{5.17}$$

where $\eta_W \equiv \eta(P_{W\,|\,\mathrm{pa}(W)})$. This effectively combines (P1') and (P2'), with the earlier result by Evans and Schulman (Eq. (5.7)) [ES99, Lemma 2] appears as a corollary of [PW17, Theorem 5]. Notably, an exact analog of Eq. (5.17) holds for TV [PW17, Theorem 8] which does not depend on the mutual information chain rule. The application of Eq. (5.17) (or its equivalent for TV) for a specific set of gates could provide new insights for fault-tolerance upper bounds.

Finally, it is also worth mentioning the result of [MMP19] on the problem of broadcasting on random DAGs with noisy wires (rather than noisy gates). Unlike the broadcasting on trees problem which we've previously discussed, the gates in this case necessarily handle fan-in $k > 1$ and therefore need to perform non-trivial processing. When this processing is done by majority gates with fan-in $k \geq 3$, Makur et al. [MMP19, Theorem 1] prove that the threshold coincides with the Evans and Schulman result for reliable computation by formulas with the same fan-in [ES03]; the same authors [MMP19, Theorem 2] show that for fan-in 2 by alternating AND and OR gates the broadcasting on DAGs threshold coincides with the Evans and Pippenger [EP98] result reliable computation by formulas with the same fan-in. The fact that the fixed-point structures coincide with those for fault-tolerant computation despite slight differences in assumptions is interesting; however, a negative result for broadcasting on random DAGs with logarithmic width remains an open problem [MMP19] (the bound on fault-tolerant formulas gives a weaker negative result for DAGs with exponential width [MMP19, Proposition 1, Part 2]).

5.2.3 Possible extensions and implications

We have developed a three-part template for the proof of negative fault-tolerance results. In this Section, we have discussed the limitations of (P1) and (P2), and existing results have been classified based on their approach (a summary is provided in Table 5.2). The template has been left deliberately vague since it is not clear what information measure will yield the best results. For example, while mutual information is a powerful measure of

	(P2)	(P2')
(P1)	[Pip88]	[EP98; ES03; Ung07]
(P1')	[ES99]	?

Table 5.2: An overview of negative fault-tolerance results based on whether information is localized to specific wires (P1) or a set of wires (P1'); and whether information contraction is bounded at the wires (P2) or at the gates (P2'). Results based on (P1) are applicable only to formulas, and results based on (P2) do not provide tight bounds. Improvements to existing upper bounds on reliable computation with circuits may be found in the bottom right quadrant.

correlations with many useful properties (e.g. the data processing inequality [CT06, Theorem 2.8.1]), its generality can also be a weakness. In particular, correlations that are captured by mutual information may not be accessible to organisms composed of noisy components subject to fan-in constraints. In these cases, we need a different approach to quantifying useful correlations. The use of a measure of correlation in deriving tight upper bounds on fault-tolerance is a meaningful way to select a measure among the vast space of candidates.

Now we summarize some potentially promising applications for our template for negative fault-tolerance results.

- One potential test for such results is in the bounding of Boolean fault-tolerance thresholds for even fan-in $k > 2$ where the threshold error rate (or even the existence of such a threshold) is unknown [ES03]. In particular, it may be instructive to develop a class of Unger-like extractable information for even $k > 2$. Though the current results based on extractable information use (P1) and are therefore limited to formulas, the method is not obviously incompatible with the more general (P1').

- Very little is known about fault-tolerance in larger alphabet models. For those that are based on q-ary majority gates as in Section 2.1.3, it is unclear whether reliable computation is possible beyond the transcritical bifurcation. An answer to this problem may also improve understanding of problem of broadcasting on trees for larger alphabets and the Potts model on a tree for which the Evans and Schulman [ES99] bound is not tight [Eva+00].

- For large alphabet models which operate over a smaller logical alphabet as in Section 2.1.4, existing measures of information are inadequate since they treat all alphabet elements equally. For example, one may define an extractable information quantity from the Lyapunov function of Eq. (2.54) for the case of Boolean computation over symmetrically noisy ternary gates. Might there be some systematic way to obtain Lyapunov functions of this sort for general gates? Again, the current analysis is limited to formulas but the approach is not obviously incompatible (P1').

- In addition to constraints on fan-in, certain applications may limit the set of available gates. For example, in large alphabet computation subject to q-ary symmetric noise (Section 2.1), though using gates with a logical Boolean alphabet as in Section 2.1.4 could lead to larger fault-tolerance thresholds, there may be a physical motivation for considering q-ary majority gates instead . In such cases it is particularly important for the measure of information to be tailored to the gate set at hand.

- The error models considered have been relatively simple so far. In addition to tailoring measure of information towards a specific gate set, we may also consider more complicated error models such as those in which the error parameter itself can vary [Ung10] or can be gate dependent [SWH20].

- The problem of broadcasting on random DAGs may prove to be a simpler setting to test new techniques. In particular, an upper bound on the broadcasting threshold for DAGs of logarithmic width remains an open problem [MMP19] and may have a bearing on fault-tolerance upper bounds.

While the correct measure of information is application dependent, it is likely that progress towards these open problems following our fairly generic template will require (P1'), (P2'), and (P3).

Chapter 6

Concluding remarks

We conclude by reflecting on the definition of the organism in Section 6.1, and by providing an outlook in Section 6.2.

6.1 Reflections on the *organism*

The term "organism" appears prominently in this thesis primarily as an allusion to von Neumann's lecture [Neu56] from which this thesis draws. As such, it is worth examining what von Neumann meant by the term. From his vantage point in the 1950s, von Neumann motivates his lecture by observing the recent (for him) synthesis of constructive logics and idealized models of the nervous systems, i.e. electrical networks: citing work by Turing [Tur36] on the former, and McCulloch–Pitts [MP43] on the latter. The concept of a computer, however, was still taking shape. It was likely not until the decade prior, with the construction of ENIAC in the 1940s, that computer came to refer to the general purpose digital variety with which we are familiar [Wei61]. From our present point of view, it is no longer controversial to describe the nervous systems of animals as performing computation. Indeed, the term has become so familiar that many physicists would not take issue with describing the universe itself, or any subset thereof, as performing a large (quantum) computation. Deutsch provided an exposition of this belief commonly called the *strong Church–Turing thesis* asserting that "every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means" [Deu85]. von Neumann, however, presented his work seven decades ago, without the benefit of these more modern developments.

The organism according to von Neumann [Neu56]—

> The terminology used in the following is taken from several fields of science; neurology, electrical engineering, and mathematics furnish most of the words. No attempt is made to be systematic in the application of terms, but it is hoped that the meaning will be clear in every case. [...] Thus, in speaking of a neuron we don't mean the animal organ, but rather one of the basic components of our network which resembles an animal neuron only superficially, and which might equally well have been called an electrical relay.

In essence von Neumann's "organ" is what today might be called a logic gate. It is possible that, had von Neumann given his lecture in modern times, he may have used the word "computer" in place of "organism." However, von Neumann makes a point of not providing a precise definition of the "organism." Following in his footsteps, neither shall we. Instead, we discuss several viable definitions of an organism: as an unconventional model of computation, as an open system, and as a local factor graph. While the differences are largely semantic, they provide different perspectives into fault-tolerance.

The organism as an unconventional model of computation— Though the word computation has become—at least in some communities—synonymous with information processing, the practicalities are that the overwhelming majority of practical general purpose electronic computers are based on Boolean signals manipulated by gates with bounded fan-in. This more restrictive and practical definition of a computer is what we term a "conventional computer." Our primary reason for using the term "organism" is to draw a contrast with this conventional definition of a computer. For the purposes of this thesis, it may be most accurate to define an organism as "anything that performs information processing using noisy components and is not a Boolean circuit," i.e. an unconventional model of computation.

The organism as open system capable of general information processing— An alternative interpretation, suggested by the author's advisor, is that of an organism as an open system. While computation typically assumes that components work noiselessly in the sense that it has perfect control over its microstates, the organism exists in a thermal bath and must contend with noise. In order to successfully accomplish its information processing task, the organism must both redundantly encode and manipulate its logical degrees of freedom. By this definition, computers as such are idealizations: both desktop computers and mammalian brains are organisms since they operate as open systems at finite temperature. Taking this view, the amount of redundancy and therefore the scale of such organisms may then be seen as a consequence of the requirement for fault-tolerance; in this case, the size of the organism can be viewed as being proportional to the size of a repetition code, e.g. the number of atoms in a transistor, or the number of neurons employed in the grid code. This is clear in the case of the size of the transistors in the integrated circuitry for example where we are fast approaching a size limit for transistors, i.e. due to thermal and quantum fluctuations, or manufacturing errors [Sha+08; Sha+18]. The size of biological organisms and their components may in part be due to a selection pressure towards fault-tolerance as well. For example, synaptic noise levels impact the optimal parameters of the grid code.

The organism as a local factor graph— What is essential to both von Neumann's fault-tolerant organism and the fault-tolerant organisms presented in this thesis, is the fact that they are composed of simple basic components subject to local noise; where simplicity is taken here to be largely synonymous with having bounded fan-in and fan-out. The framework of bounded degree probabilistic graphical models captures these essential features, namely that the full joint distribution of the signals admits a local factorization (in this case, where noise occurs at the gates, a local factor graph requires both bounded fan-in and fan-out). We represent these graphical models using factor graphs: a representative example is shown in Fig. 6.1a along with a Boolean circuit (potentially subject to independent gate-wise noise) in

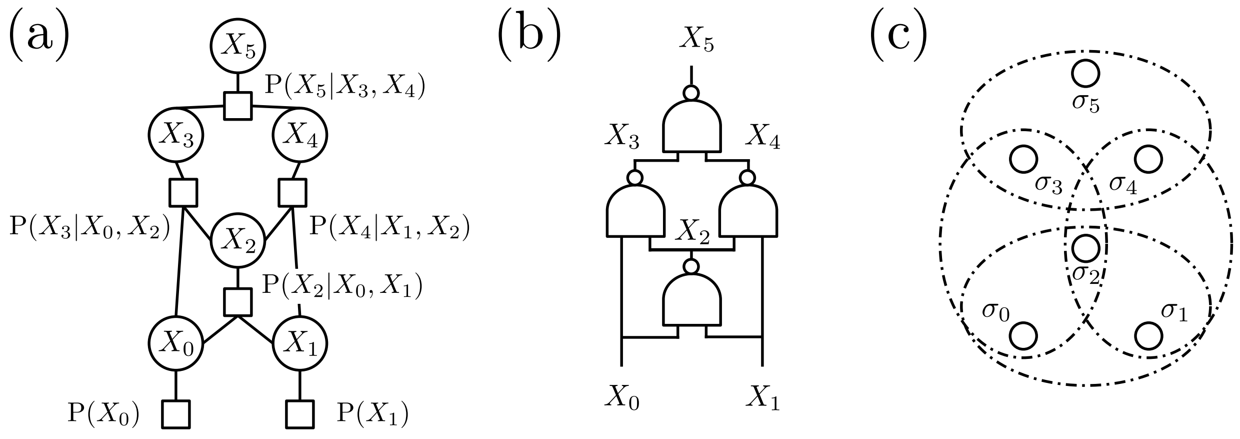


Figure 6.1: Different ways in which the factorization of Eq. (6.1) may arise. Shown are the (a) factor graph, (b) a circuit, and (c) a spin system with dash-dotted lines indicating coupling via local Hamiltonian terms.

Fig. 6.1b. Both examples correspond to joint distributions with the following factorization:

$$\Pr(\boldsymbol{X}) = \Pr(X_5|X_3,X_4)\Pr(X_4|X_1,X_2)\Pr(X_3|X_0,X_2)\Pr(X_2|X_0,X_1)\Pr(X_1)\Pr(X_0). \quad (6.1)$$

The factor graph representation makes it easy to visualize the signals and their interactions. For example, the factorization of Eq. (6.1) can arise from the following Hamiltonian with Ising degrees of freedom $\sigma_i = \pm 1$,

$$H(\boldsymbol{\sigma}) = H_{\mathrm{g}}(\sigma_3,\sigma_4,\sigma_5) + H_{\mathrm{g}}(\sigma_1,\sigma_2,\sigma_4) + H_{\mathrm{g}}(\sigma_0,\sigma_2,\sigma_3) + H_{\mathrm{g}}(\sigma_0,\sigma_1,\sigma_2) + h_0\sigma_0 + h_1\sigma_1, \quad (6.2)$$

where H_{g} represents, in this case, a 3-local gate Hamiltonian, and h_0 and h_1 represent external input fields. A representation of this Hamiltonian is depicted in Fig. 6.1c. Given such a factor graph, fault-tolerance is then the ability to maintain and manipulate information across the nodes.

This final representation, as a local factor graph, is the most general and captures essence of fault-tolerance as property of large systems composed of locally interacting components, i.e. the thermodynamic limit. It is for this reason that fault-tolerance results and their techniques have found such broad application in the study of phase transitions in Ising models, percolation, and noisy cellular automata among others [PW17; Mak19]. It describes an interesting phase of matter, one that not only demonstrates long-range correlations, but is also capable of performing general information processing. We conclude by discussing some takeaways from the work described in the results presented in this thesis.

6.2 Outlook

We summarize key takeaways below and point towards new research directions:

Native hardware-level fault-tolerant design— While there is a comparatively definitive understanding of fault-tolerance in Boolean-circuit-based computation subject to bit-flip errors, there is little understanding of how to design fault-tolerant systems in other models. It is clear from the examples of fault-tolerance in unconventional classical models of computation that the specifics of the constructions and their limitations depend crucially on the noise model and primitives assumed (Chapter 2). Since all practical systems we use to perform information processing come with their own idiosyncrasies, so too must fault-tolerance be tailored. We emphasize that this is not merely in the choice of a suitable error correcting code—a fault-tolerant design must also take into account the implementation of the noisy error correcting procedure as well as the implementation of the logical primitives. Furthermore, as with the design of an error corrected quantum subroutine (Chapter 3), we have freedom to choose the noisy units with which our computation is synthesized. Though most fault-tolerant constructions assume the application of error correction after every computation step, if error rates are sufficiently low, it may be more economical to treat the effective errors at the level of the subroutine.

Resource efficiency as a design criterion— Though the choice between non-fault-tolerant and fault-tolerant designs is clear in certain cases—in favor of non-fault-tolerant design in classical computers, and in favor of fault-tolerance in quantum computers—the choice may not be as clear generically. In our toy model (Chapter 4), we showed that whether fault-tolerance was preferred depended crucially on the resource–reliability trade-offs of the computational primitives and the required level of reliability. As heavy-tailed power-law error distributions are common in many systems, our results indicate that fault-tolerant design may provide resource benefits in many cases. In addition to informing the design of systems that perform information processing, this result might also suggest the emergence of fault-tolerance in naturally occurring information processing systems, e.g. as a result of selection pressures towards energy efficiency in biological organisms.

New measures of information— The power of mutual information as a general measure of correlations can be a limitation when one is trying to quantify the correlations that are accessible to an organism with constrained processing power. A promising direction is the development of new measures of information that are tailored to a particular model of computation. However, the space of candidate measures of information is vast. The fault-tolerance phase transition occurs when the circuit is no longer able to process information and therefore information becomes in essence inaccessible; it seems natural to require that a measure of information tailored to a noisy model of computation be capable of capturing this fault-tolerant phase. In this sense the fault-tolerance threshold may be used to select an appropriate measure of information. A more thorough discussion of this point can be found in Section 5.2.

Clearly this thesis does not provide an overarching theory of a fault-tolerant organism, nor does it does not attempt to point at what such a theory may look like. However, it is the author's hope that its components will motivate the further study in this rich field of inquiry.

Appendix A

Appendix for Chapter 2

A.1 Comparison of repetition for discrete versus analog fault-tolerance

While the repetition code is sufficient to arrive at digital fault-tolerance when subject to digital errors, such as bit flips or synaptic failure, it is insufficient for analog computation in the presence of additive Gaussian noise. Key to this is the $O(\text{polylog}(1/\epsilon))$ scaling with respect to the desired output error rate ϵ in the definition of fault-tolerance. For Boolean (and more generally discrete) random variables, suffering from i.i.d. bit-flip errors at a rate $p < 1/2$, a repetition code of size M reduces errors exponentially as $\sim p^M$. Given a target error rate ϵ, it is sufficient to choose

$$M \sim \frac{\log \frac{1}{\epsilon}}{\log \frac{1}{p}}. \tag{A.1}$$

For a circuit of N gates, an overall error of ϵ could be achieved by demanding individual gate errors ϵ/N as per the union bound. Inserting this desired error rate into Eq. (A.1), and using results of the concatenation scheme described in Section 2.2.2, we find that this translates to the desired $O(N\text{polylog}(N/\epsilon))$ scaling in the definition of fault-tolerance, so long as the error rate is below a threshold p_0 that is dependent on the details of the error correcting circuit.

For analog variables, the repetition code does not suppress errors strongly enough to achieve this scaling. For additive Gaussian noise with standard deviation σ, a repetition code of size M suppresses errors not exponentially in M, but only as $\sim \sigma/\sqrt{M}$. For a target standard deviation ϵ, the code size is required to scale as

$$M \sim \left(\frac{\sigma}{\epsilon}\right)^2. \tag{A.2}$$

Analog computation using the repetition code would require an asymptotic lower bound of $\Omega(\mathrm{poly}(1/\epsilon))$ resources, and thus does not meet our definition of fault-tolerance. In order to achieve analog fault-tolerance, we must make use of a stronger error correction code, such as the grid code utillzed in this work.

A.2 Detailed analysis of reliability in the presence of Gaussian noise and synaptic failure

In this Appendix, we expand on the analysis in Section 2.2.4 for the fully general case that takes into account both Gaussian errors and synaptic failure.

With the analysis for Gaussian failures worked out in Section 2.2.4, we proceed to consider the effect of synaptic failure for each possible type of synapse in the logical neuron of Fig. 2.9a. The goal is to find an upper bound on the probability that the logical NAND fails, corresponding to a lower bound on the threshold for synaptic failure.

First, considering the synapses from the decoder neurons x_i to the new logical phases ϕ'_j (i.e., the final layer of Fig. 2.9a), a failed synapse may originate from the correct decoder neuron or an incorrect decoder neuron. We ignore the failed synapse from an incorrect decoding, consistent with upper-bounding the failure probability. If the correct decoding fails, the encoded phase may not fire. In the application of logical weights to the logical phase of the next neuron (i.e., the first layer of Fig. 2.9a), the synapse with a logical weight a_i may similarly fail. The two phenomena of a correct decoder synapse failing and a logical weight synapse failing produce the same outcome: an input phase $\theta_i^{(1,2)}$ may fail. The effect of only a single input phase (e.g. $\theta_i^{(1)}$) failing is different from the effect of both input phases failing (i.e. $\theta_i^{(1)}$ and $\theta_i^{(2)}$). If one input phase fails, the logical phase ϕ_i assumes a uniformly random value from 0 to 1. This has no impact on $f_{\text{NAND}}(k \neq k^*)$, but it reduces the mean of $f_{\text{NAND}}(k = k^*)$ by removing one of the moduli and requires adjustment of the standard deviation by the inclusion of a random phase. If both input phases fail, the logical phase does not fire. Hence, one of the moduli is removed from both $f_{\text{NAND}}(k \neq k^*)$ and $f_{\text{NAND}}(k = k^*)$. In total, $4Mp(1-p)$ single input phases are expected to fail and $2Mp^2$ double input phases are expected to fail.

Next, consider the synapses into and out of the $\sin 2\pi\phi_i$ and $\cos 2\pi\phi_i$ neurons. Here, we also find two cases: if there is a failure of a single sine or cosine, the original distribution must be compensated by the remaining sine or cosine of the phase; if there is a failure of both, the modulus is removed entirely. In expectation, $2Mp(1-p)$ failures are expected for the former effect (for each of sine and cosine), and $2Mp^2$ failures are expected for the latter. By adding each of the failure modes independently, we place an upper bound on logical failure due to double-counting failures that happen sequentially in the network.

To obtain $f'_{\text{NAND}}(k \neq k^*)$ and $f'_{\text{NAND}}(k = k^*)$, we repeat the noisy logical neuron analysis of Eqs. (2.77) and (2.78) for the neural NAND construction including possibility of synaptic failure detailed above. We make the same assumptions as for Eqs. (2.84) and (2.86), namely logical weights $a_i = 1$ and $S = 3$ decoder neurons as per the neural NAND gate construction. Assuming a large number of moduli $M \gg 1$ and applying the central limit theorem, we obtain

$$f'_{\text{NAND}}(k=k^*) =$$
$$\mathcal{N}\left(M \cdot \frac{e^{-6\pi^2\sigma^2}\operatorname{erf}\left(\sqrt{2}\pi\sigma\right)^6}{2^9\pi^3\sigma^6} \cdot (2(p-3)p+1), \sqrt{\sigma^2 - \frac{1}{2}M((2p(p+1)-1)\left(2\sigma^2+1\right)-\zeta')}\right), \tag{A.3}$$

where

$$\zeta' = 2^{-18}\pi^{-6}\sigma^{-12}\left[(p(3p-7)+1)e^{-24\pi^2\sigma^2}\left(e^{12\pi^2\sigma^2}\operatorname{erf}\left(\sqrt{2}\pi\sigma\right)^{12} - 4\pi^3\sigma^6\operatorname{erf}\left(2\sqrt{2}\pi\sigma\right)^6\right)\right], \tag{A.4}$$

and

$$f'_{\text{NAND}}(k \neq k^*) = \mathcal{N}\left(0, \sqrt{\sigma^2 - \frac{1}{2}M(2p(p+1)-1)\left(2\sigma^2+1\right)}\right). \tag{A.5}$$

Given Eqs. (A.3) and (A.5), we may evaluate the probability of successful decoding. As explained in Section 2.2.4, we use a threshold non-linearity for decoding which is more biologically plausible than the alternative winner-take-all dynamics due to its locality. Choosing a threshold value of c, a correct decoding then requires the correct neuron sampled from $f'_{\text{NAND}}(k=k^*)$ (Eq. (A.3)) to exceed c and the two incorrect neurons sampled from $f'_{\text{NAND}}(k \neq k^*)$ (Eq. (A.5)) to lie below c, i.e.

$$1 - \epsilon(c;\sigma,p) \equiv \Pr[f'_{\text{NAND}}(k=k^*) > c] \times \Pr[f'_{\text{NAND}}(k \neq k^*) < c]^2. \tag{A.6}$$

Evaluated explicitly, we have

$$\begin{aligned}
&1-\epsilon(c;\sigma,p) = \\
&\frac{1}{8}\left(\operatorname{erf}\left(\frac{c}{\sqrt{2\sigma^2 - M(2p(p+1)-1)\left(2\sigma^2+1\right)}}\right)+1\right)^2 \times \\
&\quad \operatorname{erfc}\Bigg\{\left(2^9\pi^3 c\sigma^6 - e^{-6\pi^2\sigma^2} M(2(p-3)p+1)\operatorname{erf}\left(\sqrt{2}\pi\sigma\right)^6\right) \\
&\quad \Big/ \Bigg[-\Bigg(2M(p(3p-7)+1)e^{-24\pi^2\sigma^2}\Bigg(e^{12\pi^2\sigma^2}\times \\
&\qquad\qquad \operatorname{erf}\left(\sqrt{2}\pi\sigma\right)^{12} - 4\pi^3\sigma^6\operatorname{erf}\left(2\sqrt{2}\pi\sigma\right)^6\Bigg)\Bigg) \\
&\quad - 2^{18}\pi^6\sigma^{12}\left(M(2p(p+1)-1)\left(2\sigma^2+1\right)-2\sigma^2\right)\Bigg]^{1/2}\Bigg\},
\end{aligned} \tag{A.7}$$

where the decoding step activation function cutoff c is obtained by maximizing the probability of success over all possible values of c. This results in the logical error rate $\epsilon(\sigma,p) \equiv \min_c \epsilon(c;\sigma,p)$.

A.3 Comparison of discrete and analog thresholds for formula-based computation

Finally, we analyze a simple analog model of Boolean formula-based fault-tolerant computation which establishes a baseline for comparing discrete and analog thresholds. This is independent of neural computation, but rather serves as a pedagogical comparison of digital and analog computation that allows analog Gaussian noise to be translated into the well-known failure probability thresholds of [EP98] using 2-input NAND gates. Accordingly, the thresholds found below do not correspond to fault-tolerant neural network thresholds.

For the following analysis, we consider a binary alphabet encoded into real-valued signals $\{-1,1\}$ corresponding to the Boolean 0 and 1 respectively. For convenience we define the Boolean NAND to operate on our alphabet, and therefore it can be viewed as a function

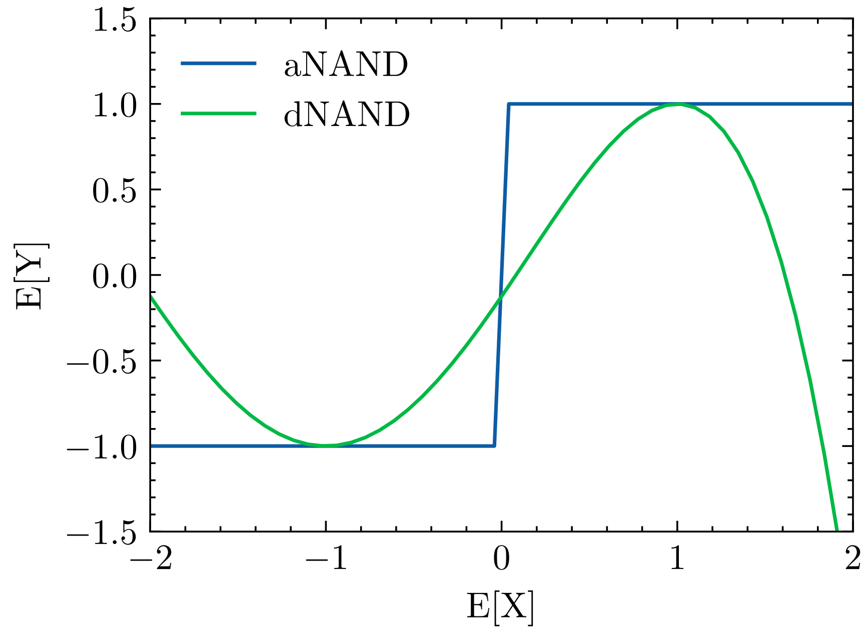


Figure A.1: A comparison of the denoiser response derived from the two proposed analog NAND gates.

NAND : $\{-1, 1\} \times \{-1, 1\} \to \{-1, 1\}$. There is considerable freedom in defining the analog NAND gate. We will choose to analyse the noise thresholds of two representative candidates, dNAND and aNAND defined below, and argue that these are extremal in the sense that all other reasonable definitions of the analog NAND result thresholds intermediate to the ones presented.

First, consider the dNAND defined as follows:

$$\mathrm{dNAND}_\sigma(x, y) \equiv \textsc{nand}(\mathrm{sgn}(x), \mathrm{sgn}(y)) + \xi, \tag{A.8}$$

where NAND is the noiseless Boolean NAND defined above, and sgn is the sign function. One can verify that a formula composed of dNAND gates works exactly a discrete NAND with an effective error rate $\epsilon = \Pr[\xi \geq 1]$. Therefore denoising is successful as long as

$$\frac{1}{2}\left[1 - \mathrm{erf}\left(\frac{1}{\sigma\sqrt{2}}\right)\right] < \epsilon_0, \tag{A.9}$$

where $\epsilon_0 = \frac{3-\sqrt{7}}{4}$ denotes the threshold of [EP98]. Numerically, we find a corresponding analog threshold of $\sigma < \sigma_d^*$ for $\sigma_d^* \approx 0.7409$.

For comparison, consider the aNAND gate defined as follows:

$$\mathrm{aNAND}_\sigma(x, y) \equiv \frac{1 - x - y - xy}{2} + \xi, \tag{A.10}$$

where $\xi \sim \mathcal{N}(0, \sigma^2)$. Note that for $\sigma = 0$ the analog NAND gate behaves as a noiseless NAND. In this notation, the balanced depth-2 binary tree denoiser used by [EP98] is written

$$\mathrm{Denoise}_\sigma(x_1, x_2, x_3, x_4) = \mathrm{aNAND}_\sigma(\mathrm{aNAND}_\sigma(x_1, x_2), \mathrm{aNAND}_\sigma(x_3, x_4)). \tag{A.11}$$

We would like to analyze the behavior of the denoiser for real-valued i.i.d. random variables X_i with $\mathbb{E}[X_i] = x \in \{-1, 1\}$ and $\mathrm{Var}(X_i) = \alpha^2$. Define $Y \equiv \mathrm{Denoise}_\sigma(X_1, X_2, X_3, X_4)$; note that $\mathbb{E}[Y]$ is a quartic polynomial in x plotted in Fig. A.1. One finds that the denoising operation is unbiased (i.e. $\mathbb{E}[Y] = x$) and has variance that depends on x: for $x = -1$,

$$\mathrm{Var}(Y) = \frac{1}{64}\left(\alpha^8 + 8\alpha^4\left(\sigma^2 + 4\right) + 16\sigma^2\left(\sigma^2 + 12\right)\right), \tag{A.12}$$

and for $x = +1$,

$$\mathrm{Var}(Y) = \frac{1}{64}\left(\alpha^4 + 8\alpha^2 + 4\sigma^2\right)^2 + \sigma^2. \tag{A.13}$$

Denoising is successful if $\mathrm{Var}(Y) < \alpha^2$ for $x \in \{-1, 1\}$. As in the discrete case, for $\sigma < \sigma_a'$, the denoiser has two fixed points, with the lower one being stable. At the denoising threshold, a saddle-node bifurcation occurs and denosing is no longer possible. Numerically, we find the denoising threshold for the aNAND to be at $\sigma_a' \approx 0.3929$. For fault-tolerant computation, we require not only that denoising can be done successfully, but that at least one step of computation can be applied between denoising stages without exceeding the capacity of our denoiser. In terms of the denoising fixed points, we require that a single aNAND gate with input variances near the lower fixed point produces a resulting value that below the upper fixed point. The critical case is that where aNAND is given inputs of opposite value. Numerically we find that computation is possible below $\sigma_a^* \approx 0.3385$.

As in the discrete case, computation below threshold is possible with minimal overhead. Since we are assuming a formula-based model of computation, the overhead in question corresponds to an increase in depth. To arrive a rough estimate of the overhead required for denoising, note that the distance between the upper and lower fixed points for gate noise σ below threshold is $\Theta(\sigma_a^{*2} - \sigma^2)$; additionally, each denoising step reduces the variance by a factor $1 - \Theta(1)$. Therefore, we find a modest depth increase by factor

$$L_a = \Theta\left(\frac{1}{\sigma_a^{*2} - \sigma^2} \log\left(\frac{1}{\sigma_a^{*2} - \sigma^2}\right)\right), \tag{A.14}$$

which establishes a fault-tolerance theorem for Boolean formulas composed of noisy analog NAND gates and is a similar form to the factor found by Evans and Pippenger [EP98] for the case of computation with discrete NAND gates.

Appendix B

Appendix for Chapter 3

B.1 Additional results related to the canonical expansion

In this Appendix, we demonstrate a class of operators, namely linear combinations of noiseless even-length QSP unitaries, that admit the canonical expansion of Section 3.2.4. This class includes the diagrammatic components of QSP error channels Kraus operators studied in this paper.

Remark 40 (QSP operators of even length exhibit canonical expansion)**.** Let $U_0(\theta) = \mathrm{QSP}(\theta; \vec{\phi})$ be a noiseless QSP unitary of length-$2d$, then $U_0(\theta)$ admits a canonical expansion at zeroth order in ϵ. Additionally, $\mathcal{P}_j^{(\sigma,k)} = 0$ for all $k > 0$ and for $k = 0$ with $\sigma \in \{0, x, y, z\}$, and $j \geq d$.

Proof. A QSP unitary of even length $2d$ can be written in the form Eq. (3.3) with polynomials $P, Q \in \mathbb{C}[a]$ where P has degree at most $2d$ and is of even parity in $\cos\theta$; and Q has degree at most $2d-1$ and is of odd parity [Gil+19]. Writing the polynomials as

$$P(\cos\theta) = \sum_{j=0}^{d} p_{2j} \cos^{2j}(\theta), \tag{B.1a}$$

$$Q(\cos\theta) = \sum_{j=0}^{d-1} q_{2j+1} \cos^{2j+1}(\theta), \tag{B.1b}$$

we have

$$\begin{aligned}
w_0(\theta) &= \Re[P(\cos\theta)], \\
&= \cos^2\theta \sum_{j=-1}^{d-1} \Re[p_{2j}] \cos^{2j}(\theta),
\end{aligned} \tag{B.2a}$$

$$\begin{aligned}
x_0(\theta) &= \sin\theta \Re[Q(\cos\theta)], \\
&= \sin(2\theta) \sum_{j=0}^{d-1} \frac{1}{2} \Re[q_{2j+1}] \cos^{2j}(\theta),
\end{aligned} \tag{B.2b}$$

$$\begin{aligned}
y_0(\theta) &= -\sin\theta \Im[Q(\cos\theta)], \\
&= -\sin(2\theta) \sum_{j=0}^{d-1} \frac{1}{2} \Im[q_{2j+1}] \cos^{2j}(\theta),
\end{aligned} \tag{B.2c}$$

$$\begin{aligned}
z_0(\theta) &= \Im[P(\cos\theta)], \\
&= \cos^2\theta \sum_{j=-1}^{d-1} \Im[p_{2j}] \cos^{2j}(\theta).
\end{aligned} \tag{B.2d}$$

This is of the desired form. □

Since the transformation from an operator to its canonical profile is linear, we have the following corollary:

Corollary 41 (Linear combinations of QSP operators of even length exhibit canonical expansion)**.** If an operator A can be decomposed into

$$A = \sum_i \gamma_i U_i, \tag{B.3}$$

for $\gamma_i \in \mathbb{R}$ and QSP unitaries U_i of even length (i.e. A can be written as a linear combination of QSP unitaries of even length), then it admits a canonical expansion.

Next we show how the canonical expansion is transformed under Z-rotation and conjugation. It will often be convenient to represent a canonical profile $\mathcal{P}$ in vector form. Assuming

$\mathcal{P}_j^{(k)} = 0$ for all $j \geq d$, we can write the entire canonical profile as a vector in $\mathbb{R}^{4(d+1)}$,

$$\vec{\mathcal{P}}^{(k)} \equiv \begin{pmatrix} \vec{\mathcal{P}}_{d-1}^{(k)} \\ \vdots \\ \vec{\mathcal{P}}_0^{(k)} \\ \vec{\mathcal{P}}_{-1}^{(k)} \end{pmatrix}, \tag{B.4}$$

where the vector $\vec{\mathcal{P}}_j^{(k)} \equiv (\mathcal{P}^{(0,k)}, \mathcal{P}_j^{(x,k)}, \mathcal{P}_j^{(y,k)}, \mathcal{P}_j^{(z,k)})$.

Remark 42 (Canonical expansion of Z-rotation)**.** Let U_ϵ be an operator admitting canonical expansion with vector form $\vec{\mathcal{P}}^{(k)}$ at order $k \geq 0$. Then $V_\epsilon = e^{i\chi_0 Z} U_\epsilon e^{i\chi_1 Z}$ for $\chi_0, \chi_1 \in \mathbb{R}$ has canonical profile at order k

$$\vec{\mathcal{P}}^{(k)'} = O_z(\chi_0,\chi_1)\vec{\mathcal{P}}^{(k)} \equiv \begin{pmatrix} O_z(\chi_0,\chi_1) & & & & \\ & O_z(\chi_0,\chi_1) & & & \\ & & O_z(\chi_0,\chi_1) & & \\ & & & \ddots & \\ & & & & O_z(\chi_0,\chi_1) \end{pmatrix} \begin{pmatrix} \vec{\mathcal{P}}_{d-1}^{(k)} \\ \vdots \\ \vec{\mathcal{P}}_0^{(k)} \\ \vec{\mathcal{P}}_{-1}^{(k)} \end{pmatrix}, \tag{B.5}$$

$$\text{where} \quad O_z(\chi_0,\chi_1) \equiv \begin{pmatrix} \cos(\chi_0+\chi_1) & 0 & 0 & -\sin(\chi_0+\chi_1) \\ 0 & \cos(\chi_0-\chi_1) & \sin(\chi_0-\chi_1) & 0 \\ 0 & -\sin(\chi_0-\chi_1) & \cos(\chi_0-\chi_1) & 0 \\ \sin(\chi_0+\chi_1) & 0 & 0 & \cos(\chi_0+\chi_1) \end{pmatrix}. \tag{B.6}$$

We note several useful properties of the conjugation operation in the following remarks:

Remark 43 (Recurrence under conjugation, first-order)**.** Given an unbiased operator U_ϵ with functions w, x, y, z in its canonical expansion, the corresponding functions of conjugated operator $U'_\epsilon = \mathcal{C}_{m,n,\eta} U_\epsilon$ are given by

$$\begin{pmatrix} w' \\ x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos^2(\theta) & 0 & 0 & 0 \\ 0 & \cos(2\eta) & -\cos(2\theta)\sin(2\eta) & \cos^2(\theta)\sin(2\eta) \\ 0 & \sin(2\eta) & \cos(2\theta)\cos(2\eta) & -\cos^2(\theta)\cos(2\eta) \\ 0 & 0 & 4\sin^2(\theta) & \cos(2\theta) \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix}. \tag{B.7}$$

From Eq. (B.7), we can write the recurrence of the canonical profiles using vector form of Eq. (B.4). Suppose there exists some $d \geq 0$ such that the canonical profile of U_ϵ satisfies $\mathcal{P}_j^{(\sigma,1)} = 0$ for all $j \geq d$, then recurrence of the canonical profile of U'_ϵ satisfies

$$\begin{pmatrix} \vec{\mathcal{P}'}^{(1)}_d \\ \vec{\mathcal{P}'}^{(1)}_{d-1} \\ \vdots \\ \vec{\mathcal{P}'}^{(1)}_0 \\ \vec{\mathcal{P}'}^{(1)}_{-1} \end{pmatrix} = \begin{pmatrix} A(\eta) & & & & \\ B(\eta) & A(\eta) & & & \\ & B(\eta) & A(\eta) & & \\ & & & \ddots & \\ & & & B(\eta) & A(\eta) \\ & & & & B(\eta) \end{pmatrix} \begin{pmatrix} \vec{\mathcal{P}}^{(1)}_{d-1} \\ \vdots \\ \vec{\mathcal{P}}^{(1)}_0 \\ \vec{\mathcal{P}}^{(1)}_{-1} \end{pmatrix}, \tag{B.8}$$

where the matrix on the right-hand side is a block bidiagonal matrix of size $4(d+2) \times 4(d+1)$

with blocks

$$A(\eta) \equiv \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -2\sin 2\eta & \sin 2\eta \\ 0 & 0 & 2\cos 2\eta & \cos 2\eta \\ 0 & 0 & -4 & 2 \end{pmatrix}, \tag{B.9a}$$

$$\text{and} \quad B(\eta) \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2\eta & \sin 2\eta & 0 \\ 0 & \sin 2\eta & -\cos 2\eta & 0 \\ 0 & 0 & 4 & -1 \end{pmatrix}. \tag{B.9b}$$

Remark 44 (Linearity of conjugation)**.** Note that conjugation is linear. For operators U, V and coefficients $\alpha, \beta \in \mathbb{R}$,

$$\mathcal{C}_{m,n,\eta}(\alpha U + \beta V) = \alpha \mathcal{C}_{m,n,\eta} U + \beta \mathcal{C}_{m,n,\eta} V. \tag{B.10}$$

Similarly, we can write the effect of anti-conjugation operation on the canonical profile:

Remark 45 (Recurrence under anti-conjugation, first-order)**.** We can construct the inverse to the conjugation operation using Corollary 20.

$$\mathcal{C}^{-1}_{n,\eta}\mathrm{QSP}(\theta; \vec{\phi}) \equiv e^{i\pi(n+\frac{1}{2})Z} W e^{i\eta Z} \mathrm{QSP}(\theta; \vec{\phi}) e^{-i(\eta+\frac{\pi}{2})Z} W e^{i\frac{\pi}{2}Z}, \tag{B.11}$$

Let U_ϵ be a degree-d operator with canonical expansion $\mathcal{P}$, the canonical expansion of

$\mathcal{C}_{n,\eta}^{-1}U_\epsilon$ is given by $\mathcal{P}'$ of the form Eq. (B.8) with blocks

$$A(\eta) \equiv \begin{pmatrix} 0 & -4\sin 2\eta & -4\cos 2\eta & -2 \\ 0 & -2\sin 2\eta & 2\cos 2\eta & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{B.12a}$$

$$\text{and} \quad B(\eta) \equiv \begin{pmatrix} 0 & -4\sin 2\eta & 4\cos 2\eta & 1 \\ 0 & \sin 2\eta & -\cos 2\eta & 0 \\ 0 & -\cos 2\eta & -\sin 2\eta & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{B.12b}$$

Remark 46 (Canonical profile of product of unbiased operators, leading-order)**.** If U_ϵ, V_ϵ are both unbiased to order $k \geq 1$ with canonical profiles $\mathcal{P}$ and $\mathcal{Q}$ respectively, then $W_\epsilon = U_\epsilon V_\epsilon$ is also unbiased to order k with an canonical profile $\mathcal{R}$ satisfying

$$\mathcal{R}_j^{(\sigma,k)} = \mathcal{P}_j^{(\sigma,k)} + \mathcal{Q}_j^{(\sigma,k)}, \qquad \sigma \in \{x, y\}, \tag{B.13a}$$

$$\mathcal{R}_j^{(\sigma,1)} = \mathcal{P}_j^{(\sigma,1)} + \mathcal{Q}_j^{(\sigma,1)}, \qquad \sigma \in \{0, z\}, \tag{B.13b}$$

for all j.

Remark 47 (Equivalence of expansions of unbiased operators to leading-order)**.** Two different expansions in ϵ have been presented: the expansion in Remark 46 is performed in the exponent (i.e. the expansion is in the Hermitian generator of the unitary operator), while the canonical expansion is of the unitary operator itself. These expansions do not yield the same expansion coefficients in general; however, for unbiased operators, the coefficients are identical to the leading-order for $k \geq 1$:

$$e^{i\epsilon(z+O(\epsilon))Z+i\epsilon^k[(x+O(\epsilon))X+(y+O(\epsilon))Y]} = (1+O(\epsilon^2))I+i\epsilon(z+O(\epsilon))Z+i\epsilon^k[(x+O(\epsilon))X+(y+O(\epsilon))Y]. \tag{B.14}$$

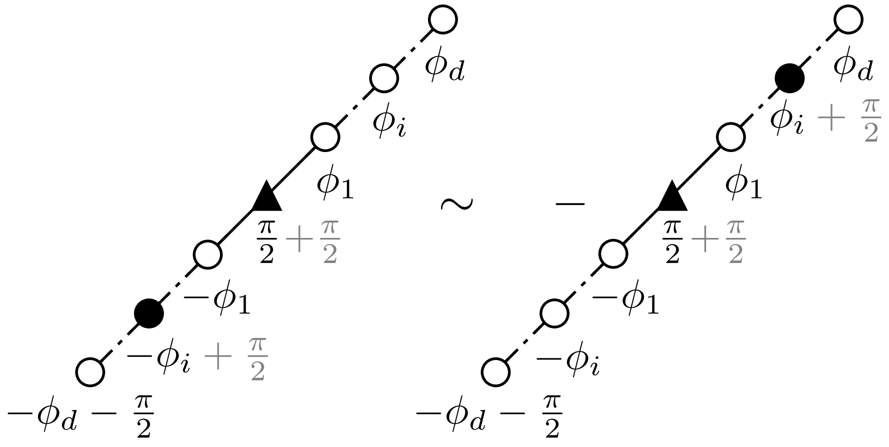


Figure B.1: Diagrammatic representations of Remark 48 i.e. the XY-equivalence of conjugation by $\pi/2$.

As all of our analysis will be done recursively on the leading order, we will use these forms interchangeably.

B.2 Proof of Theorems 4 and 5: Higher-order Component-wise Recovery

In this Appendix, we generalize the first-order component-wise recovery construction of Section 3.4.4 to all orders in k. First, we make the following observation which is useful for simplifying higher-order operators:

Remark 48 ($\pi/2$-rotation identity)**.** Let V' be a length-$2r$ QSP operator parameterized by phases

$$\begin{gathered}(-\phi_d-\pi/2,\dots,-\phi_i+\pi/2,\dots,-\phi_{d-r+1},\pi,\\ \phi_{d-r+1}+b_{d-r+1}\pi/2,\dots,\phi_i,\dots,\phi_d+b_d\pi/2),\end{gathered} \tag{B.15}$$

with $\phi_i\in\mathbb{R}$ and $b_i\in\{0,1\}$. Using Remark 42 and Remark 43, we find that

$$V'\sim -V, \tag{B.16}$$

for QSP operator V of the form Definition 49 parameterized by phases

$$\begin{aligned}(-\phi_d - \pi/2, \ldots, -\phi_i, \ldots, -\phi_{d-r+1}, \pi, \\ \phi_{d-r+1} + b_{d-r+1}\pi/2, \ldots, \phi_i + \pi/2, \ldots, \phi_d + b_d\pi/2).\end{aligned} \tag{B.17}$$

This is shown diagrammatically for an example QSP in Fig. B.1.

The preceding remark motivates the following nomenclature useful in the analysis of the Kraus operators of error and recovery channels to higher order in ϵ:

Definition 49 (Error component)**.** Let U be length-d QSP operator parameterized by real phases $(\phi_0, \ldots, \phi_d)$. Then a length-$2r$ QSP V with $1 \le r \le d$ is said to be an *error component of QSP U* if it can be written in the following form:

$$\begin{aligned}V = \mathrm{QSP}(\theta;(-\phi_d - \pi/2, \ldots, -\phi_{d-r+1}, \pi, \\ \phi_{d-r+1} + b_{d-r+1}\pi/2, \ldots, \phi_d + b_d\pi/2)),\end{aligned} \tag{B.18}$$

where $b_i \in \{0, 1\}$. Furthermore, it is assumed that no ϕ_i for $i < d$ is a half-integer multiple of π; otherwise, we can perform elision to simplify the diagram. This is a generalization of Definition 16.

Definition 50 (Standard form, higher-order)**.** To simplify the analysis of error and recovery operators at any order, we generalize Definition 15 by writing the contribution at each order as an XY-equivalent linear combination of diagrams of the form Definition 49. Note that the first-order analysis presented in Section 3.4.1 is already in this form. For all higher-orders, this can be accomplished using repeated application of Remark 48 and the identity $e^{i\pi Z} = -I$, keeping track of factors of -1.

Diagrams to order $k = 3$ are shown for a generic length-3 QSP in Fig. B.2.

Remark 51 (Constructing higher-order recovery sequences)**.** Let R_ϵ and $\bar{R}_\epsilon$ be k^{th}-order

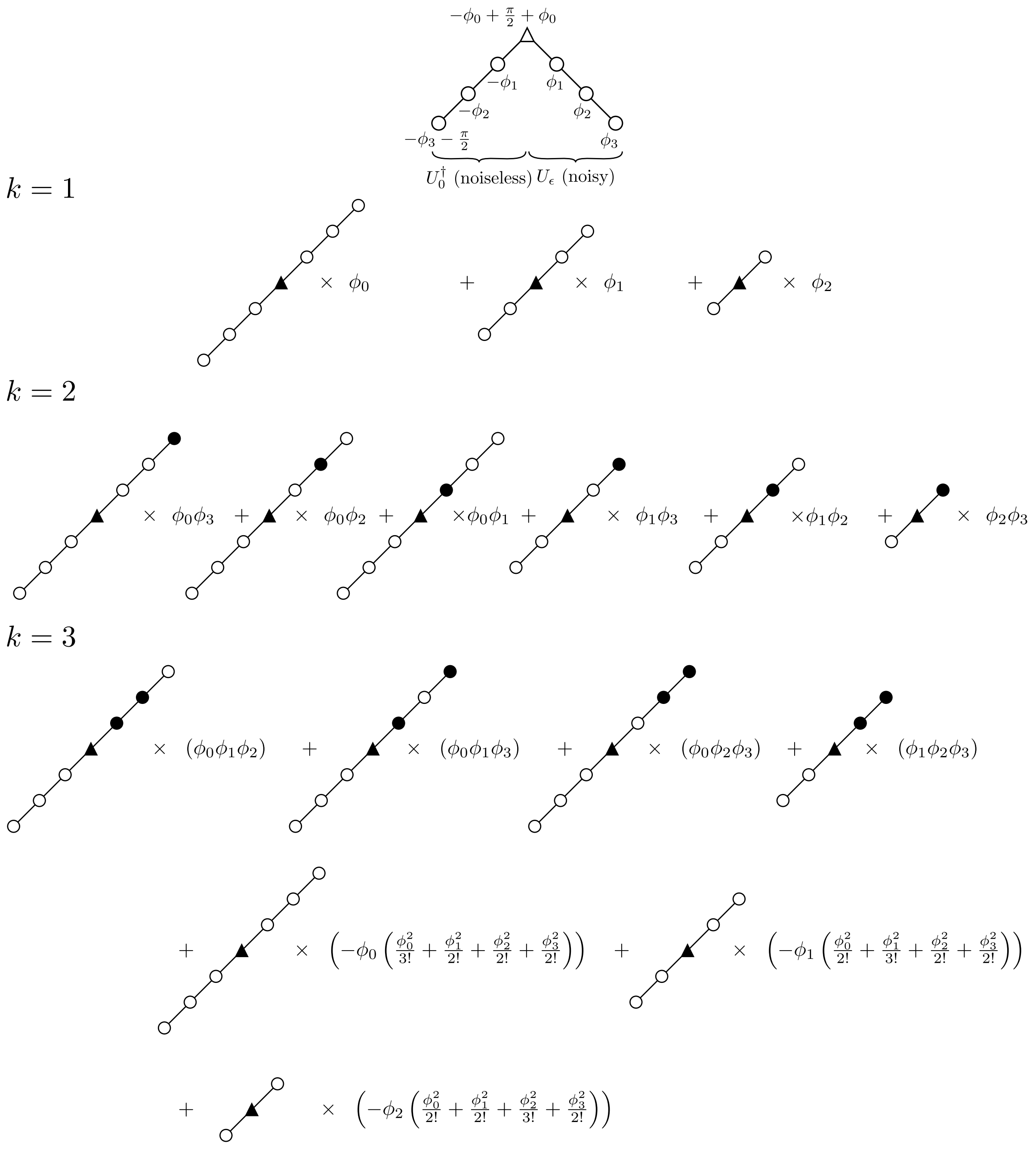

$-\phi_0 + \frac{\pi}{2} + \phi_0$
$-\phi_1$
ϕ_1
$-\phi_2$
ϕ_2
$-\phi_3 - \frac{\pi}{2}$
ϕ_3
$U_0^\dagger$ (noiseless)
U_ϵ (noisy)
$k = 1$
$\times\ \phi_0$
$+$
$\times\ \phi_1$
$+$
$\times\ \phi_2$
$k = 2$
$\times\ \phi_0\phi_3$
$+$
$\times\ \phi_0\phi_2$
$+$
$\times\phi_0\phi_1$
$+$
$\times\ \phi_1\phi_3$
$+$
$\times\phi_1\phi_2$
$+$
$\times\ \phi_2\phi_3$
$k = 3$
$\times\ (\phi_0\phi_1\phi_2)$
$+$
$\times\ (\phi_0\phi_1\phi_3)$
$+$
$\times\ (\phi_0\phi_2\phi_3)$
$+$
$\times\ (\phi_1\phi_2\phi_3)$
$+$
$\times\ \left(-\phi_0\left(\frac{\phi_0^2}{3!} + \frac{\phi_1^2}{2!} + \frac{\phi_2^2}{2!} + \frac{\phi_3^2}{2!}\right)\right)$
$+$
$\times\ \left(-\phi_1\left(\frac{\phi_0^2}{2!} + \frac{\phi_1^2}{3!} + \frac{\phi_2^2}{2!} + \frac{\phi_3^2}{2!}\right)\right)$
$+$
$\times\ \left(-\phi_2\left(\frac{\phi_0^2}{2!} + \frac{\phi_1^2}{2!} + \frac{\phi_2^2}{3!} + \frac{\phi_3^2}{2!}\right)\right)$

Figure B.2: Diagrammatic representation of error terms for a representative length-3 QSP parameterized by $\phi_i \in \mathbb{R}$ at (a) order-1, (b) order-2, and c) order-3. To save space, we have omitted phase labels in the expansions. Diagrams are understood to be in the form of Definition 49 with open circles denoting $b_i = 0$ and filled circles denoting $b_i = 1$.

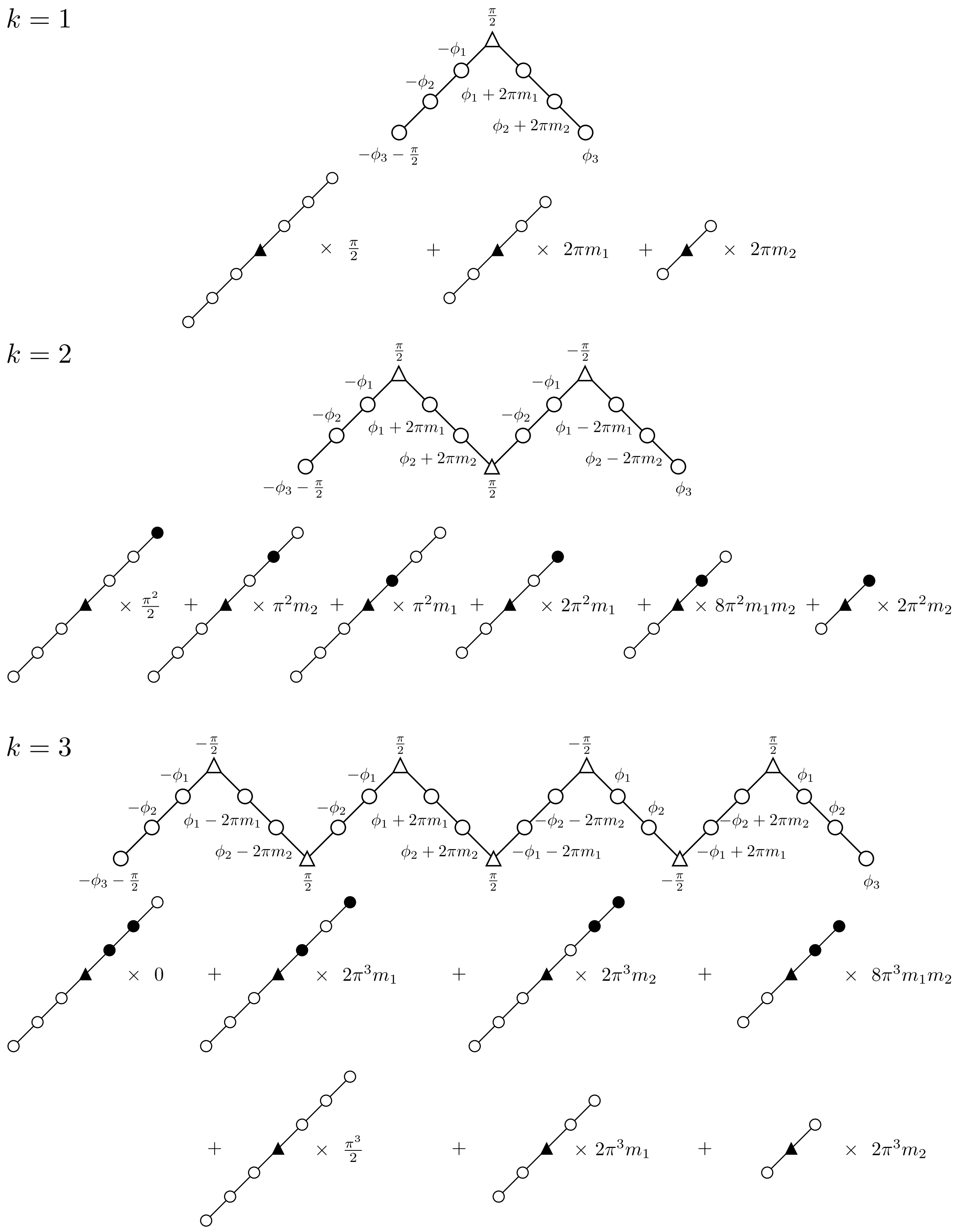


Figure B.3: Diagrammatic analysis of expansions for representative recovery diagrams of (a) order-1, (b) order-2, and (c) order-3. As in Fig. B.3, phase labels are omitted and diagrams are understood to be in the form of Definition 49 with open circles denoting $b_i = 0$ and filled circles denoting $b_i = 1$.

unbiased sequences with canonical profiles $\mathcal{R}$ and $\bar{\mathcal{R}}$ respectively with

$$\mathcal{R}_j^{(\sigma,k)} = -\bar{\mathcal{R}}_j^{(\sigma,k)} \qquad (\sigma = x, y, z), \tag{B.19}$$

for all j, and

$$\mathcal{R}_j^{(z,k)} = \bar{\mathcal{R}}_j^{(z,k')} = 0, \tag{B.20}$$

for all j for $k' < k$.

Then the operator $S_\epsilon \equiv R_\epsilon \bar{R}_\epsilon$ with canonical profile $\mathcal{S}$ is an order-$(k+1)$ unbiased sequence with

$$\mathcal{S}_j^{(\sigma,k+1)} = \mathcal{R}_j^{(\sigma,k)} + \bar{\mathcal{R}}_j^{(\sigma,k)} \quad (\sigma = x, y, z), \tag{B.21}$$

for all j.

We can make use of the above observation to generate general higher-order recovery sequences. For instance, to create a second-order recovery sequence, we may combine two first-order sequences. In general, this requires careful choice of m_i and n. Example recovery sequences up to order-3 are shown in Fig. B.3.

Lemma 52 (Higher-order component-wise recovery)**.** Let $U_\epsilon^{(k-1)}$ be a noisy QSP unitary XY recovered to order-$(k-1)$,

$$U_\epsilon^{(k-1)} = U_0 e^{i(\chi + O(\epsilon))Z + \epsilon^k((x+O(\epsilon))X + (y+O(\epsilon))Y)}. \tag{B.22}$$

If $U_\epsilon^{(0)}$ is of length-d and with c unique phases (up to factors of 2π). Then, a recovery sequence $R^{(k)}$ of length $\Theta(2^k c^k d)$ exists to recover up to order-k,

$$U_\epsilon^{(k-1)} R^{(k)} = U_0 e^{i(\chi + O(\epsilon))Z + \epsilon^k((x+O(\epsilon))X + (y+O(\epsilon))Y)}. \tag{B.23}$$

Proof. The result for higher-orders is similar to that for first-order recovery. The units of recovery at order-k can be constructed using Remark 51 and are of length $\Theta(2^k d)$ (example recovery units are shown in Fig. B.3).

As in the first-order case, we can correct groups of diagrams scaled by the same real coefficient using each recovery unit. In general, each recovery unit may need to be repeated a constant number of times with different values of m_i and n in order to attain the desired integral coefficients.

The main factor bounding recovery length is the number of distinct groups with different real coefficients. The real coefficients at order-k consist of the k-tuples of the c distinct phases, of which there are $\binom{c}{k} = \Theta(c^k)$. Note that again we consider phases equivalent if they differ only by an integer multiple of 2π as these require only a constant number of additional recovery units.

Thus the recovery to order-k can be accomplished using a recovery sequence of length $\Theta(2^k c^k d)$. □

We are now ready to prove Theorem 5, restated below, in full generality.

Theorem 5 (Upper bound on recovery length). *Given any noisy QSP operator $U_\epsilon(\theta)$ of length d with c distinct phases (up to factors of 2π) and an integer $k \geq 1$, there exists a recovery sequence $R_\epsilon(\theta)$ satisfying*

$$|\langle 0|U_\epsilon R_\epsilon|0\rangle|^2 = |\langle 0|U_0|0\rangle|^2 + O(\epsilon^{k+1}), \tag{3.46}$$

for all θ. Furthermore, there exists a QSP operator satisfying the above with length at most $O(2^k c^{k(k+1)/2} d)$.

Proof. Let U_ϵ be a length-d QSP sequence with c unique phases with error operator E_ϵ. We perform recovery order-by-order so that we can make use of the additive property of leading-order terms (Remark 46). At each order, from Eq. (3.45), it suffices to append a sequence of relatively negative XY-equivalent diagrams. Recovery to first-order has been shown in Section 3.4.4 with a sequence $R_\epsilon^{(1)}$ of length $\Theta(cd)$.

After appending the first-order recovery sequence, we have $U_\epsilon R_\epsilon^{(1)}$, a length $\Theta(cd)$ QSP sequence. The key to showing the desired scaling is to notice this sequence has large phase redundancy: namely, the length $\Theta(cd)$ QSP sequence is parameterized by the same phases, except for an additional $\Theta(c)$ new distinct phases used in the counter-rotation. Thus, by Lemma 52, recovery to second-order can be accomplished using a recovery sequence of length $\Theta(2^2 \times c^2 \times cd) = \Theta(2^2 c^3 d)$. Recovery to order-$k$ can be accomplished using a recovery sequence a factor $\Theta(c^k)$ longer than the previous order. Overall this construction requires a length $O(2^k c^{k(k+1)/2} d)$ sequence, thus proving the Theorem. □

We have Theorem 4, restated below, as a corollary.

Theorem 4 (Recoverability)**.** *Given any noisy QSP operator $U_\epsilon(\theta)$ of length d and an integer $k \geq 1$, there exists a recovery sequence $R_\epsilon(\theta)$ satisfying*

$$|\langle 0|U_\epsilon R_\epsilon|0\rangle|^2 = |\langle 0|U_0|0\rangle|^2 + O(\epsilon^{k+1}), \tag{3.44}$$

for all θ.

Finally we note that the scaling of the recovery length in Theorem 5 shows that QSPs with fewer unique phases require fewer resources for recovery. As an example, we provide an explicit choice of recovery phases for a QSP with a single unique phase.

Remark 53 (Example for QSP with single unique phase.)**.** Let U_ϵ be a length-d QSP with a single unique phase ϕ (e.g. Grover search). Using this construction, we can construct a first-order recovery operator by choosing phases

$$\begin{aligned}\vec{\eta}_1 = (&-\phi - \pi - \delta, -\phi, \ldots, -\phi, \pi/2,\\ &\phi + 2\pi m, \ldots, \phi + 2\pi m, \phi + \pi/2 + \delta),\end{aligned} \tag{B.24a}$$

$$\begin{aligned}\vec{\eta}_2 = (&-\phi - 4n\pi - \delta, -\phi, \ldots, -\phi, 4n\pi - \pi/2,\\ &\phi + 2\pi m, \ldots, \phi + 2\pi m, \phi + \pi/2 + \delta),\end{aligned} \tag{B.24b}$$

$$\begin{aligned}\vec{\eta}_3 = (&-\phi - \pi + \delta, -\phi, \ldots, -\phi, \pi/2,\\ &\phi + 2\pi m, \ldots, \phi + 2\pi m, \phi + \pi/2 - \delta),\end{aligned} \tag{B.24c}$$

$$\begin{aligned}\vec{\eta}_4 = (&-\phi - 4n\pi + \delta, -\phi, \ldots, -\phi, 4n\pi - \pi/2,\\ &\phi + 2\pi m, \ldots, \phi + 2\pi m, \phi + \pi/2 - \delta),\end{aligned} \tag{B.24d}$$

with appropriate $m, n \in \mathbb{Z}$ and δ depending on ϕ. Note crucially that each "$\ldots$" hides only $\Theta(d)$ phases.

Letting $V_\epsilon^{(1)} = \text{QSP}(\theta; \vec{\eta}_1), \ldots, V_\epsilon^{(4)} = \text{QSP}(\theta; \vec{\eta}_1)$, $V_\epsilon^{(1)} V_\epsilon^{(2)} V_\epsilon^{(3)} V_\epsilon^{(4)}$ is a first-order recovery sequence for U_ϵ.

B.3 Alternate Proof of Theorem 4: Degree-wise Recovery

In this Appendix, we present an alternate recovery construction for the coherent error model in Section 3.4. Though this construction is exponentially less efficient, generating a length $\Theta(2^k d^{2^k})$ sequence, we find that it is sufficiently different to be worth discussion. Furthermore, despite being asymptotically worse, it can produce shorter recovery sequences in practice due to the large constants hidden in Lemma 52. Whereas the construction presented in performs recovery component-wise, here we perform recovery degree-wise.

B.3.1 First-order

We now describe our construction of the recovery sequence that corrects the first-order error in a QSP sequence. Given a faulty QSP sequence U_ϵ, let E_ϵ be its error operator and $\mathcal{P}$ its error profile, it suffices by Eq. (3.45) to construct a recovery sequence R_ϵ with an error profile $\mathcal{R}$ satisfying

$$\mathcal{R}_j^{(\sigma,1)} = -\mathcal{P}_j^{(\sigma,1)}, \tag{B.25}$$

for $\sigma = x, y$ and for all j.

The construction of R_ϵ is recursive. In the first iteration, we construct R_ϵ that satisfies Eq. (B.25) only at $j_{\max}$, the largest j such that $\mathcal{P}_j^{(\sigma,1)} \neq 0$ ($\sigma = x, y$). At the end of this iteration, the appended QSP sequence has a modified error profile $\mathcal{P}'$ such that $\mathcal{P}_j'^{(\sigma,0)} = 0$ for all $j \geq j_{\max}$, resulting in a lower $j_{\max}$ for the next iteration. Repeating this procedure until $\mathcal{P}_j^{(\sigma,0)} = 0$ for all j, we arrive at the desired recovery sequence.

The building block for our recovery sequence is the conjugations in Eq. (B.34). The following Lemma gives the error profile for QSP sequence that results from the conjugations.

Lemma 54 (Top-degree recovery term, first-order)**.** Let R be the length-$(2d)$ QSP sequence resulting from d conjugations in Eq. (B.34) with $m_1 = \cdots = m_d = 0$ and $n_1 = \cdots = n_d = n$. Let $\mathcal{R}$ be the error profile of R_ϵ. We have

$$\begin{pmatrix} \mathcal{R}_{d-1}^{(x,1)} \\ \mathcal{R}_{d-1}^{(y,1)} \\ \mathcal{R}_{d-1}^{(z,1)} \end{pmatrix} = \pi 2^{2d-3}(2n+1) \prod_{j=1}^{d-1} \cos^2(\eta_j) \begin{pmatrix} \sin(2\eta_d) \\ -\cos(2\eta_d) \\ 2 \end{pmatrix}. \tag{B.26}$$

Proof. We prove Lemma 54 by induction. For $d = 1$, the error profile of the corresponding length-2 QSP satisfies Eq. (B.26):

$$\begin{pmatrix} \mathcal{R}_0^{(x,1)} \\ \mathcal{R}_0^{(y,1)} \\ \mathcal{R}_0^{(z,1)} \end{pmatrix} = \pi 2^{-1}(2n+1) \begin{pmatrix} \sin(2\eta_1) \\ -\cos(2\eta_1) \\ 2 \end{pmatrix}. \tag{B.27}$$

Suppose Lemma 54 holds for all length-$2d$ sequences. We shall prove that it also holds for all length-$(2d+2)$ sequences.

Let R' be a length-$(2d+2)$ sequence satisfying the assumptions of Lemma 54 and R be a length-$2d$ sequence satisfying

$$R' = \mathcal{C}_{0,n,\eta_{d+1}} R. \tag{B.28}$$

Let $\mathcal{R}'$ and $\mathcal{R}$ be the error profiles of R'_ϵ and R_ϵ, respectively. Using Eq. (B.7), we have

$$\mathcal{R}_d'^{(x,0)} = \sin(2\eta_{d+1})(\mathcal{R}_{d-1}^{(z,0)} - 2\mathcal{R}_{d-1}^{(y,0)}), \tag{B.29a}$$

$$\mathcal{R}_d'^{(y,0)} = -\cos(2\eta_{d+1})(\mathcal{R}_{d-1}^{(z,0)} - 2\mathcal{R}_{d-1}^{(y,0)}), \tag{B.29b}$$

$$\mathcal{R}_d'^{(z,0)} = 2(\mathcal{R}_{d-1}^{(z,0)} - 2\mathcal{R}_{d-1}^{(y,0)}). \tag{B.29c}$$

Applying Lemma 54 on $\mathcal{R}$, we have

$$\mathcal{R}_{d-1}^{(z,0)} - 2\mathcal{R}_{d-1}^{(y,0)} = \pi 2^{2d-1}(2n+1)\prod_{j=1}^{d} \cos^2(\eta_j). \tag{B.30}$$

Therefore,

$$\begin{pmatrix} \mathcal{R}_d'^{(x,1)} \\ \mathcal{R}_d'^{(y,1)} \\ \mathcal{R}_d'^{(z,1)} \end{pmatrix} = \pi 2^{2d-1}(2n+1)\prod_{j=1}^{d} \cos^2(\eta_j) \begin{pmatrix} \sin(2\eta_{d+1}) \\ -\cos(2\eta_{d+1}) \\ 2 \end{pmatrix}, \tag{B.31}$$

and Lemma 54 holds for R'. By induction, Lemma 54 holds for length-$2d$ QSP sequences for all $d \geq 1$. □

Also, recall from Remark 43 that $R_j^{(x,1)} = R_j^{(y,1)} = R_j^{(z,1)} = 0$ for all $j \geq d$.

Lemma 55 (First-order degree-wise recovery)**.** Let U be a length-d QSP sequence, E_ϵ be the error operator for U, and $\mathcal{P}$ its error profile. Let $j_{\max}$ be the largest j such that either $\mathcal{P}_j^{(x,1)} \neq 0$ or $\mathcal{P}_j^{(y,1)} \neq 1$. There exists an unbiased recovery sequence R such that the error profile $\mathcal{P}'$ of $U_\epsilon R_\epsilon$ satisfies

$$\mathcal{P}'^{(x,1)}_j = \mathcal{P}'^{(y,1)}_j = 0, \tag{B.32}$$

for all $j \geq j_{\max}$. In addition, the length of the recovery sequence is at most $2(j_{\max}+1)$ if $j_{\max} \geq 1$ and at most 4 if $j_{\max} = 0$.

Proof. First, we consider $j_{\max} \geq 1$. Let n be the smallest integer such that

$$\sqrt{(\mathcal{P}_{j_{\max}}^{(x,1)})^2 + (\mathcal{P}_{j_{\max}}^{(y,1)})^2} \leq \pi 2^{2j_{\max}-1}\left(n + \frac{1}{2}\right). \tag{B.33}$$

Let R be the length-$(2j_{\max}+2)$ QSP sequence in the form

$$\mathcal{C}_{m_d,n,\eta_d} \ldots \mathcal{C}_{m_1,n,\eta_1} I, \tag{B.34}$$

with $m_1 = \cdots = m_{j_{\max}+1} = 0$, $n_1 = \cdots = n_{j_{\max}+1} = n$ and $\mathcal{R}$ be the error profile of R_ϵ. By Lemma 54, we have

$$\begin{pmatrix} \mathcal{R}_{j_{\max}}^{(x,1)} \\ \mathcal{R}_{j_{\max}}^{(y,1)} \end{pmatrix} = \pi 2^{2j_{\max}-1}\left(n + \frac{1}{2}\right) \times \prod_{j=1}^{j_{\max}} \cos^2(\eta_j) \begin{pmatrix} \sin(2\eta_{j_{\max}+1}) \\ -\cos(2\eta_{j_{\max}+1}) \end{pmatrix}. \tag{B.35}$$

Next, we choose $\eta_1 = \cdots = \eta_{j_{\max}-1} = 0$,

$$\eta_{j_{\max}} = \cos^{-1}\left(\frac{\sqrt{(\mathcal{P}_{j_{\max}}^{(x,1)})^2 + (\mathcal{P}_{j_{\max}}^{(y,1)})^2}}{\pi 2^{2j_{\max}-1}\left(n+\frac{1}{2}\right)}\right)^{1/2} \tag{B.36}$$

$$\eta_{j_{\max}+1} = -\frac{1}{2}\tan^{-1}\left(\frac{\mathcal{P}_{j_{\max}}^{(x,1)}}{\mathcal{P}_{j_{\max}}^{(y,1)}}\right) \leq 0. \tag{B.37}$$

Substituting these phase angles into Eq. (B.35), we obtain

$$\begin{pmatrix} \mathcal{R}_{j_{\max}}^{(x,1)} \\ \mathcal{R}_{j_{\max}}^{(y,1)} \end{pmatrix} = - \begin{pmatrix} \mathcal{P}_{j_{\max}}^{(x,1)} \\ \mathcal{P}_{j_{\max}}^{(y,1)} \end{pmatrix}. \tag{B.38}$$

Thus by Remark 46 we have the Lemma for $j_{\max} \geq 1$.

Finally, we consider the case $j_{\max} = 0$. Recall that for $j_{\max} \geq 1$, we have a continuous control over the magnitude of the error profile provided by $\eta_{j_{\max}}$. For $j_{\max} = 0$, we use the counter-rotation trick of Eq. (3.49). Accordingly, we choose $\eta = -\tan^{-1}\left(\mathcal{P}_0^{(x,1)}/\mathcal{P}_0^{(y,1)}\right) \leq 0$ and

$$\delta\eta = \frac{1}{2}\cos^{-1}\left(\frac{\sqrt{(\mathcal{P}_0^{(x,1)})^2 + (\mathcal{P}_0^{(y,1)})^2}}{\pi\left(n+\frac{1}{2}\right)}\right) \tag{B.39}$$

to arrive at Eq. (B.38) for $j_{\max} = 0$. This concludes the proof of the Lemma. □

Repeatedly applying Lemma 55, we incrementally lower $j_{\max}$. When $\mathcal{P}_j^{(x,0)} = \mathcal{P}_j^{(y,0)} = 0$ for all $j \geq 0$, we arrive at Theorem 4 for $k = 1$. Since the length of the recovery sequence in each iteration of Lemma 54 is $2(j_{\max}+1)$ and $j_{\max}$ is initially at most $d-1$, the total length of the recovery sequence is at most

$$4 + \sum_{j_{\max}=1}^{d-1} 2(j_{\max}+1) = d^2 + d + 2. \tag{B.40}$$

Therefore, for a length-d QSP U_ϵ, there exists recovery sequence R_ϵ of length d^2+d+2 satisfying Theorem 4 for $k = 1$.

B.3.2 Higher-order

We now generalize to higher-orders.

First, we provide an explicit recursive construction of higher order unbiased sequences.

Lemma 56 (Top-degree recovery term, higher-order)**.** For all $k \geq 1$, there exists a k^{th}-order unbiased QSP sequence of length-$(2^k d)$ R_ϵ, parameterized by $\eta_1, \ldots, \eta_d \in [-\pi, \pi)$ and $n \in \mathbb{Z}$ with an error profile $\mathcal{R}$ satisfying

$$\begin{pmatrix} \mathcal{R}_{d-1}^{(x,k)} \\ \mathcal{R}_{d-1}^{(y,k)} \end{pmatrix} = \pi^k 2^{2d-3}(2n+1)^k \prod_{j=1}^{d-1} \cos^2(\eta_j) \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^{k-1} \begin{pmatrix} \sin(2\eta_d) \\ -\cos(2\eta_d) \end{pmatrix}, \tag{B.41}$$

and $\mathcal{R}_j^{(x,k-1)} = \mathcal{R}_j^{(y,k-1)} = 0$ for all $j \geq d$.

Proof. We will provide a recursive construction for a length-$(2^k d)$ QSP satisfying the Lemma.

Let R_ϵ be the length-$2d$ recovery sequence parameterized by $n \in \mathbb{Z}$ and $\eta_1, \ldots, \eta_d$ as in Lemma 54 and $\mathcal{R}$ its error profile. From the Lemma, we have

$$\begin{pmatrix} \mathcal{R}_{d-1}^{(x,1)} \\ \mathcal{R}_{d-1}^{(y,1)} \end{pmatrix} = \pi 2^{2d-3}(2n+1) \prod_{j=1}^{d-1} \cos^2(\eta_j) \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^{0} \begin{pmatrix} \sin(2\eta_d) \\ -\cos(2\eta_d) \end{pmatrix}, \tag{B.42}$$

and $\mathcal{R}_j^{(x,0)} = \mathcal{R}_j^{(y,0)} = 0$ for all $j \geq d$. So R satisfies the Lemma for $k = 1$.

Suppose the Lemma holds for $k \geq 1$ and let R_ϵ be the k^{th}-order unbiased QSP sequence satisfying the Lemma. We define $\bar{R}_\epsilon$ to be the QSP sequence identical to R_ϵ except that $\eta_d \mapsto \eta_d + \frac{\pi}{2}$ and $n \mapsto -(n+1)$. Let $\mathcal{R}$ and $\bar{\mathcal{R}}$ be their respective error profiles. Using Remark 51, one can show that

$$\mathcal{R}_j^{(\sigma,k)} = \bar{\mathcal{R}}_j^{(\sigma,k)} \qquad (\sigma = x, y), \tag{B.43a}$$

$$\mathcal{R}_j^{(z,k)} = -\bar{\mathcal{R}}_j^{(z,k)}, \tag{B.43b}$$

for all j and $k \geq 1$.

Thus we can construct a sequence S using Remark 51 that is unbiased to order $k+1$ as follow:

$$S_\epsilon \equiv e^{-i\pi(n+\frac{1}{2})(1+\epsilon)Z} R_\epsilon e^{i\pi(n+\frac{1}{2})(1+\epsilon)Z} \bar{R}_\epsilon, \tag{B.44}$$

which is a length-($2^{k+1}d$) QSP unitary. By the result of Remark 51, the error profile $\mathcal{S}$ of S_ϵ satisfies

$$\begin{pmatrix} \mathcal{S}_{d-1}^{(x,k)} \\ \mathcal{S}_{d-1}^{(y,k)} \end{pmatrix} = \pi(2n+1) \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} \mathcal{R}_{d-1}^{(y,k-1)} \\ \mathcal{R}_{d-1}^{(x,k-1)} \end{pmatrix}, \tag{B.45}$$

$$= \pi^k 2^{2d-3}(2n+1)^k \prod_{j=1}^{d-1} \cos^2(\eta_j) \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^{k-1} \begin{pmatrix} \sin(2\eta_d) \\ -\cos(2\eta_d) \end{pmatrix}. \tag{B.46}$$

Therefore the Lemma holds for $k+1$ and, by induction, it holds for all k. □

Lemma 57 (Higher-order degree-wise recovery)**.** Let U be a length-d QSP sequence, E_ϵ be its error operator, and $\mathcal{P}$ its error profile. Suppose E_ϵ is unbiased to order-k, that is $\mathcal{P}_j^{(x,k')} = \mathcal{P}_j^{(y,k')} = 0$ for all $j \geq 0$ and $k' < k$. Let $j_{\max}$ be the largest j such that either $\mathcal{P}_j^{(x,k)} \neq 0$ or $\mathcal{P}_j^{(y,k)} \neq 0$. There exists an unbiased recovery sequence R such that the error profile $\mathcal{P}'$ of $U_\epsilon R_\epsilon$ satisfies

$$\mathcal{P}_j'^{(x,k)} = \mathcal{P}_j'^{(y,k)} = 0, \tag{B.47}$$

for all $j \geq j_{\max}$ and $k \geq 0$. In addition, the length of the recovery sequence is at most $2^{k+1}(j_{\max}+1)$ if $j_{\max} \geq 1$ and at most 2^{k+2} if $j_{\max} = 0$.

Proof. The proof of the Lemma is nearly identical to that of Lemma 55.

First, consider $k \geq 1$ and $j_{\max} \geq 1$. Let n be the smallest integer such that

$$\sqrt{(\mathcal{P}_{j_{\max}}^{(x,k)})^2 + (\mathcal{P}_{j_{\max}}^{(y,k)})^2} \leq \pi^k 2^{2j_{\max}-1}(2n+1)^k. \tag{B.48}$$

Let R be the kth-order unbiased length-($2^{k+1}(j_{\max}+1)$) QSP sequence parameterized by $\eta_1, \ldots, \eta_{j_{\max}+1} \in [-\pi, \pi)$ and $n \in \mathbb{Z}$, that satisfies Lemma 56 and $\mathcal{R}$ be its error profile.

From Lemma 56, we have

$$\begin{pmatrix} \mathcal{R}^{(x,k)}_{j_{\max}} \\ \mathcal{R}^{(y,k)}_{j_{\max}} \end{pmatrix} = \pi^k 2^{2j_{\max}-1}(2n+1)^k \prod_{j=1}^{j_{\max}} \cos^2(\eta_j) \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}^{k-1} \begin{pmatrix} \sin(2\eta_{j_{\max}+1}) \\ -\cos(2\eta_{j_{\max}+1}) \end{pmatrix}, \tag{B.49}$$

Next, we choose $\eta_1 = \cdots = \eta_{j_{\max}-1} = 0$, n from Eq. (B.48), and

$$\eta_{j_{\max}} = \cos^{-1}\left(\frac{\sqrt{(\mathcal{P}^{(x,k)}_{j_{\max}})^2 + (\mathcal{P}^{(y,k)}_{j_{\max}})^2}}{\pi^k 2^{2j_{\max}-1}(2n+1)^k} \right)^{1/2}, \tag{B.50}$$

$$\eta_{j_{\max}+1} = -\frac{1}{2}\tan^{-1}\left(\frac{\mathcal{P}^{(x,k)}_{j_{\max}}}{\mathcal{P}^{(y,k)}_{j_{\max}}} \right) - \frac{3\pi k}{4}. \tag{B.51}$$

Substituting these phase angles into Eq. (B.49), we obtain

$$\begin{pmatrix} \mathcal{R}^{(x,k)}_{j_{\max}} \\ \mathcal{R}^{(y,k)}_{j_{\max}} \end{pmatrix} = - \begin{pmatrix} \mathcal{P}^{(x,k)}_{j_{\max}} \\ \mathcal{P}^{(y,k)}_{j_{\max}} \end{pmatrix}. \tag{B.52}$$

From Remark 46, we have Lemma 57 for $j_{\max} \geq 1$.

Finally, for the case $j_{\max} = 0$ we again use the counter-rotation trick of Eq. (3.49) setting $\eta_1 = \eta \pm \delta\eta$. We choose $\eta = -\tan^{-1}\left(\mathcal{P}^{(x,0)}_0 / \mathcal{P}^{(y,0)}_0\right) - \frac{3\pi k}{4}$ and

$$\delta\eta = \frac{1}{2}\cos^{-1}\left(\frac{2\sqrt{(\mathcal{P}^{(x,k)}_0)^2 + (\mathcal{P}^{(y,k)}_0)^2}}{\pi^k (2n+1)^k} \right), \tag{B.53}$$

to arrive at Eq. (B.52) for $j_{\max} = 0$. This concludes the proof of Lemma 57. □

This provides an alternate proof for Theorem 4 which is restated below for convenience.

Theorem 4 (Recoverability)**.** *Given any noisy QSP operator $U_\epsilon(\theta)$ of length d and an integer $k \geq 1$, there exists a recovery sequence $R_\epsilon(\theta)$ satisfying*

$$|\langle 0|U_\epsilon R_\epsilon|0\rangle|^2 = |\langle 0|U_0|0\rangle|^2 + O(\epsilon^{k+1}), \tag{3.44}$$

for all θ.

Proof. To prove Theorem 4 in full generality for $k \geq 1$, we repeatedly apply Lemma 57.

Let $R_\epsilon^{(k')}$ be the recovery operator accumulated from such repeated applications for order-k'. We start by constructing $R_\epsilon^{(1)}$ such that $U_0^\dagger U_\epsilon R_\epsilon^{(1)}$ is unbiased to order-2. We then increment k', repeating the process up to $k' = k+1$ to obtain a $(k+1)^{\text{th}}$-order unbiased operator $U_0^\dagger U_\epsilon R_\epsilon^{(1)} \dots R_\epsilon^{(k)}$.

We can therefore write

$$U_\epsilon R_\epsilon^{(1)} \dots R_\epsilon^{(k)} \equiv U_0 e^{i\epsilon(z+O(\epsilon))Z + i\epsilon^{k+1}[(x+O(\epsilon))X+(y+O(\epsilon))Y]}, \tag{B.54}$$

as required by Eq. (3.45) thus providing an alternate proof of Theorem 4. □

Recalling that for $k = 1$, the length of the recovery operation using this construction is $d^2 + d + 2 = \Theta(2^1 d^{2^1})$ (Eq. (B.40)), we note that given a noisy QSP corrected to order k of length-d_k with $d_k = \Theta(2^k d^{2^k})$, we can perform correction to the $(k+1)^{\text{th}}$-order using at most d_k applications of Lemma 56 for each $0 \leq j_{\max} \leq d_k - 1$. From Lemma 57, each application adds length $2^k(j_{\max}+1)$ for $j_{\max} > 0$ and 2^{k+1} for $j_{\max} = 0$. The overall length of the resulting sequence therefore has length d_k satisfying

$$d_{k+1} = d_k + 2^{k+2} + \sum_{j_{\max}=1}^{d_k} 2^{k+1}(j_{\max}+1) = \Theta(2^{k+1} d^{2^{k+1}}). \tag{B.55}$$

The main reason for the exponentially worse performance as compared with the construction in Appendix B.2 is the fact that we do not make use of the phase redundancy in the recovered operators, and thus recovery at each order is performed de novo.

Appendix C

Appendix for Chapter 5

C.1 Proof of Pareto Set Scaling Theorem

As discussed in Section 5.1.3, the performance of our algorithm depends on the size of the Pareto frontier. In the paper, we provide experimental evidence for the polynomial scaling of the DIB Pareto frontier of a variety of datasets. In this appendix, we will prove Theorem 7, which provides sufficient conditions for the sparsity of the Pareto frontier and apply it to a number of examples.

As in Section 5.1.3, let $S = \{(U_i, V_i)\}_{i=1}^N$ be a sample of N i.i.d. bivariate random variables having joint cumulative distribution $F_{UV}(u, v)$. Further, let $R_{S,U}(U_i)$ and $R_{S,V}(V_i)$ be the marginal rank statistics of U and V, respectively, with respect to S; that is, U_i is the $R_{S,U}(U_i)^{\text{th}}$ smallest U-value in S and likewise for V. Ties can be broken arbitrarily. We will often drop the subscripts on $R_{S,U}$ and $R_{S,V}$ when the context is obvious.

Definition 58. Given a permutation $\sigma : [N] \to [N]$ where $[N] \equiv \{1, \dots, N\}$, we call i a *sequential minimum* if $j < i \Rightarrow \sigma(j) > \sigma(i)$.

We would now like to show that the marginal rank statistics S are sufficient for determining membership in Pareto(S), which we formalize in Lemma 59.

Lemma 59. Let $\sigma_U(i) = R(U_i)$ and $\sigma_V(i) = R(V_i)$. An element $(U_i, V_i) \in S$ is maximal if and only if its rank, i, is a sequential minimum of $\sigma_U \circ \sigma_V$.

Proof. ($\Longrightarrow$) Assume $(U_i, V_{\sigma(i)}) \in S$ is maximal. For any other point $(u_j, v_{\sigma(j)}) \in S$, $i \neq j$, if $j < i \Rightarrow u_i < u_j$, then $v_{\sigma(i)} > v_{\sigma(j)}$ by definition of maximality, which implies $\sigma(j) > \sigma(i)$, showing that i is a sequential minimum of σ.

($\Longleftarrow$) For $(u_i, v_{\sigma(i)}) \in S$ such that i is a sequential minimum of σ. For any other point $(u_j, v_{\sigma(j)}) \in S$, $i \neq j$, either $i < j \Rightarrow u_i > u_j$ showing that $(u_i, v_{\sigma(i)})$ is maximal, or $j > i \Rightarrow \sigma(j) > \sigma(i)$ by definition of a sequential minimum, which implies $v_{\sigma(i)} > v_{\sigma(j)}$ showing that $(u_i, v_{\sigma(i)})$ is maximal. □

Corollary 60. Membership in the Pareto set is invariant under strictly monotonic transformations of U or V.

Proof. Strictly monotonic transformations leave the rank statistics unchanged and therefore also do not affect membership in the Pareto set by Lemma 59. □

We now turn to the main result of this Appendix: the proof of Theorem 7, which is restated here for convenience.

Theorem 7. *Let $S = \{(U_i, V_i)\}_{i=1}^N$ be a set of bivariate random variables drawn i.i.d. from a distribution with Lipschitz continuous CDF $F(u, v)$, and invertible marginal CDFs F_U, F_V. Define the region*

$$R_N \equiv \left\{(u, v) \in [0, 1] \times [0, 1] : u + v - C(u, v) \geq e^{-\frac{1}{N}}\right\}, \tag{5.3}$$

where $C(u, v)$ denotes the copula of (U_i, V_i), which is the function that satisfies $F(u, v) = C(F_U(u), F_V(v))$.

Then, if the Lebesgue measure of this region $\lambda(R_N) = \Theta\left(\frac{\ell(N)}{N}\right)$ as $N \to \infty$, we have $\mathbb{E}[|\operatorname{Pareto}(S)|] = \Theta(\ell(N))$.

Proof. Since the marginal CDFs are invertible by assumption and therefore strictly monotonic, Corollary 60 allows us to consider instead $U_i' = F_U(U_i)$ and $V_i' = F_V(V_i)$ with the promise that $\text{Pareto}(S') = \text{Pareto}(S)$ where $S' \equiv \{(U_i', V_i'\}$. Note that $F_{U'}(u') = u'$ and $F_{V'}(v') = v'$, and therefore without loss of generality, we can assume F_U and F_V are uniform distributions over the interval $[0, 1]$ dropping the prime notation. This allows us to identify the copula with the joint CDF $C(F_U(u), F_V(v)) = C(u, v) = F(u, v)$.

Let $\mathbf{1}_A(x)$ denote the indicator function of a set A: taking the value 1 for $x \in A$ and 0 otherwise. Then, $\mathbb{E}_S\left[|\text{Pareto}(S)|\right] = \mathbb{E}_S\left[\sum_{i=1}^N \mathbf{1}_{\text{Pareto}(S)}(U_i, V_i)\right]$. Making use of the linearity of expectation and noting that (U_i, V_i) are drawn i.i.d., we can write

$$\mathbb{E}_S\left[|\text{Pareto}(S)|\right] = N\mathbb{E}_S\left[\mathbf{1}_{\text{Pareto}(S)}(U_1, V_1)\right]. \tag{C.1}$$

Note that $\mathbb{E}\left[\mathbf{1}_{\text{Pareto}(S)}(u, v)\right] = (1 - \Pr[U > u, V > v])^N = (u + v - C(u, v))^N$, which follows from the definition of Pareto optimality. For convenience, we define $\hat{C}(u, v) \equiv u+v-C(u, v)$ yielding

$$\mathbb{E}_S\left[|\text{Pareto}(S)|\right] = \int_0^1 \int_0^1 N f(u, v) \hat{C}(u, v)^{N-1} du dv. \tag{C.2}$$

Take $f_{\max}$ to be the maximum value f achieves over the domain, we are guaranteed $f_{\max} < \infty$ as C is Lipschitz by assumption. Therefore

$$\mathbb{E}_S\left[|\text{Pareto}(S)|\right] \leq N f_{max} \int_0^1 \int_0^1 \hat{C}(u, v)^{N-1} du dv. \tag{C.3}$$

Now, define $\hat{C}_N$, which is equal to $\hat{C}$ in the region R_N and 0 otherwise. We also define the region

$$R_N' \equiv \left\{(u, v) \in [0, 1] \times [0, 1] : e^{-\frac{1+2\log N}{N}} \leq \hat{C}(u, v) < e^{-\frac{1}{N}}\right\}. \tag{C.4}$$

We now split the integral over $[0,1]^2$ into three disjoint parts

$$\begin{aligned}\int_0^1 \int_0^1 \hat{C}(u,v)^{N-1} dudv = \\ \int_{R_N} \hat{C}_N(u,v)^{N-1} dudv + \int_{R'_N} \hat{C}(u,v)^{N-1} dudv + \int_{[0,1]^2 \setminus R_N \cup R'_N} \hat{C}(u,v)^{N-1} dudv.\end{aligned} \tag{C.5}$$

The integrand of the final term is bounded by $e^{-\log(N)+O(1)} = O(N^{-1})$ and $\lambda([0,1]^2 \setminus R_N \cup R'_N) = \Theta(1)$; therefore, this term goes to 0 as $N \to \infty$. Now, we turn to the middle term on the right-hand side. Since C is 2-non-decreasing and Lipschitz, we have that the measure of the set $\lambda(R'_N) = \Theta\left(e^{-\frac{1}{N}} - e^{-\frac{1+2\log N}{N}}\right) = \Theta\left(\frac{\log N}{N}\right)$, $\hat{C}(u,v) < e^{-\frac{1}{N}}$ in the region R'_N by definition, and therefore, the second term goes to 0 as $N \to \infty$. Since there is always at least one point on the Pareto frontier, the first term must be $\Omega(1)$, and the integral is dominated by the portion over R_N. Equivalently,

$$\int_0^1 \int_0^1 \hat{C}(u,v)^{N-1} dudv \sim \int_0^1 \int_0^1 \hat{C}_N(u,v)^{N-1} dudv. \tag{C.6}$$

Further,

$$\int_0^1 \int_0^1 N C_N(u,v)^{N-1} dudv \leq N \int_0^1 \int_0^1 \mathbf{1}_{R_N}(u,v) dudv = N\lambda(R_N) = \ell(N). \tag{C.7}$$

Following the chain of inequalities and asymptotic equivalences, we arrive at the desired result $\mathbb{E}_S[|\operatorname{Pareto}(S)|] = \Theta(\ell(N))$. $\square$

We now apply Theorem 7 to a few illustrative examples.

The Fréchet–Hoeffding copulae, W and M, are extremal in the sense that, written in *two* dimensions, any copula C must satisfy $W(u,v) \leq C(u,v) \leq M(u,v)$, $\forall (u,v) \in [0,1]^2$; where $W(u,v) = \max(u+v-1, 0)$ and $M(u,v) = \min(u,v)$. W and M correspond to complete negative and positive monotonic dependence, respectively.

Example 61 (Fréchet–Hoeffding lower bound)**.** First, let us consider the scaling of the

Pareto of a distribution with extremal copula $W(u, v)$. In this case, we note that the region $[0,1]^2 \setminus R_N$ is the triangle with vertices at $\{(0,0),(0,e^{-1/N}),(e^{-1/N},0)\}$, and therefore $\lambda(R_N) = 1 - \frac{1}{2}\exp^{-\frac{2}{N}}$. For large N, $\lambda(R_N) = \frac{1}{2} + O(N^{-1})$. We see that this satisfies the conditions for Theorem 7 with $\ell(N) = N$, giving $\mathbb{E}_S\left[|\operatorname{Pareto}(S)|\right] = \Theta(N)$ as expected for a distribution with complete negative monotonic dependence.

Example 62 (Fréchet–Hoeffding upper bound)**.** First, let us consider the scaling of the Pareto of a distribution with extremal copula $M(u, v)$. In this case, we note that the region $[0,1]^2 \setminus R_N$ is the region $[0, e^{-1/N}]$, and therefore, $\lambda(R_N) = 1 - \exp^{-\frac{2}{N}}$. For large N, $\lambda(R_N) = \frac{2}{N} + O(N^{-2})$. We see that this satisfies the conditions for Theorem 7 with $\ell(N) = 1$, giving $\mathbb{E}_S\left[|\operatorname{Pareto}(S)|\right] = \Theta(1)$ as expected for a distribution with complete positive monotonic dependence.

Example 63 (Independent random variables)**.** Next, let us consider the case of independent random variables with copula $C(u, v) = uv$. Note that the level curves in this case $u + v - C(u, v) = e^{-\frac{1}{N}}$ are given by $v = \frac{e^{-\frac{1}{N}} - u}{1-u}$. We can then integrate to find the area of the region R_N

$$\lambda(R_N) = 1 - \int_0^{e^{-\frac{1}{N}}} \frac{e^{-\frac{1}{N}} - u}{1-u} du = e^{-1/n}\left(1 - e^{1/n}\right)\left(\log\left(1 - e^{-1/n}\right) - 1\right). \tag{C.8}$$

Expanding for large N, we find that $\lambda(R_N) = \frac{\log N}{N} + O(N^{-1})$. We see that this satisfies the conditions for Theorem 7 with $\ell(N) = \log N$, giving $\mathbb{E}_S\left[|\operatorname{Pareto}(S)|\right] = \Theta(\log N)$.

Theorem 7 provides a useful tool to pin down the scaling of the size of the Pareto set. Due to the relatively quick decay of the additional terms in Eq. (C.5), we find that scaling estimates using the region R_N are quite accurate even for modest N. However, its applicability is limited, as it requires that we either have an analytic expression for the copula or are otherwise able to estimate the copula to precision $1/N$. In particular, we are not able to prove any bounds for the DIB frontier, which is the case $U = H(Z)$, and $V = I(Z;Y)$. We suspect that for most realistic datasets, including points on the DIB plane, that $\ell(N) = \text{polylog}(N)$, which implies that the scaling of the Pareto set is likewise $\Theta(\text{polylog}(N))$. Since we are interested in the large N behavior, we are hopeful that more general results can be found through the study of extreme-value copulas, which we leave for future work.

C.2 Auxiliary Functions

In this appendix, we provide the pseudocode for the important auxiliary functions used in Algorithms 1 and 2. The Pareto set data structure is a list of point structures. A point structure, p, contains fields for both objectives p.x, p.y, and optional fields for storing the uncertainties p.dx, p.dy and clustering function p.f. As a list, the Pareto set P for point p, and index i, also supports the functions SIZE(P) returning the number of elements in P, INSERT(p, i, P) for inserting point p at index i, and REMOVE(i, P) for removing the entry at index i. Additionally, since the Pareto set P is maintained in sorted order by its first index, we can find the correct index at which to insert a new point in logarithmic time: for a point p and Pareto set P, this is written FIND_INDEX($p.x, P$) in the pseudocode of Algorithms Algorithms 3 to 5.

Algorithm 3 Check if a point is Pareto optimal

```
Input: Point on objective plane p, and Pareto Set P
Output: TRUE if and only if p is Pareto optimal in P
1: procedure IS_PARETO(p, P)
2:     i = FIND_INDEX(p.x, P)                    ▷ Return correct value to insert p in P
       return SIZE(P) = 0  or  i = SIZE(P) or P[i + 1].y < p.y
```

C.3 The Symmetric Pareto Mapper

In this appendix, we consider one way that Algorithm 1 can be modified to accommodate an additional structure in the dataset. The full pseudocode is provided in Algorithm 6 with the key difference occurring on line 12. This modification amounts to a redefining of the

Algorithm 4 Add point to Pareto Set

Input: Point on objective plane p, and Pareto Set P

Output: Updated Pareto Set P

```
1: procedure PARETO_ADD($p, P$)
2:     if IS_PARETO($p, P$) then                              ▷ Insert only if Pareto optimal
3:         $i$ = FIND_INDEX($p.x, P$)
4:         $P \leftarrow$ INSERT($p, i, P$)                        ▷ Insert point into correct location
5:         while $i <$ SIZE($P$) and $p$.y $> P$[i + 1].y do        ▷ Remove dominated points
6:             REMOVE($i + 1, P$)
7:             $i = i + 1$
    return P
```

Algorithm 5 Calculate distance to Pareto frontier

Input: Point on objective plane p, and Pareto Set P

Output: Distance to Pareto frontier (defined to be zero if Pareto optimal)

```
1: procedure PARETO_DISTANCE($p, P$)
2:     if IS_PARETO($p, P$) then return 0  ▷ Distance defined to be zero if point is Pareto optimal
3:     $i$ = FIND_INDEX($p.x, P$)
4:     $d = P$[i].y $- p$.y                                        ▷ Check top boundary
5:     while $P$[i].x $- p$.x $< d$ do
6:         if $i + 1 <$ SIZE($P$) and $P$[i].y $> p$.y then
7:             $q$ = POINT($x = P$[i].x$, y = P$[i + 1].y)
8:             $d$ = MINIMUM(DISTANCE($p, q$)$, d$)                    ▷ Check corners
9:         else
10:            $d$ = MINIMUM($P$[i].x $- p$.x$, d$)                  ▷ Check right boundary
     return $d$
```

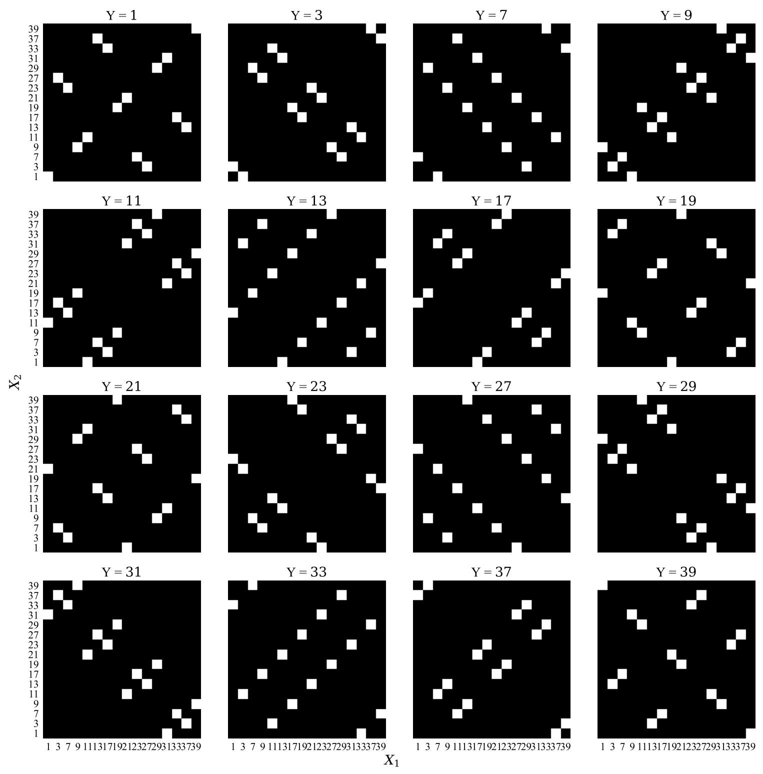
compressed variable $Z = (f(X_1), f(X_2))$. We would like to discover an encoding f that trades off the entropy of the encoding with the ability to predict Y from $(f(X_1), f(X_2))$. This corresponds to the following graphical model:

$$\begin{tikzcd}
Z_1 & X_1 \arrow[l, "f"'] \arrow[r] & Y \\
Z_2 & X_2 \arrow[l, "f"] \arrow[ur] &
\end{tikzcd}$$

Algorithm 6 Symmetric Pareto Mapper

Input: Joint distribution $A, B, C \sim p_{ABC}$, and search parameter ε
Output: Approximate Pareto frontier P

```
 1: procedure SYMMETRIC_PARETO_MAPPER($p_{ABC}, \varepsilon$)
 2:     Pareto set $P = \emptyset$                                   ▷ Initialize maintained Pareto set
 3:     Queue $Q = \emptyset$                                        ▷ Initialize search queue
 4:     Point $p = (\mathrm{x} = -\mathrm{H}(p_{X_1X_2})/2, \mathrm{y} = \mathrm{I}(p_{X_1X_2;Y}), \mathrm{f} = \mathrm{id})$     ▷ Evaluate trivial clustering
 5:     $P \leftarrow$ INSERT($p, P$)
 6:     $Q \leftarrow$ ENQUEUE(id, $Q$)    ▷ Start with identity clustering id : $[n] \to [n]$ where $n = |X|$
 7:     while $Q$ is not $\emptyset$ do
 8:         $f =$ DEQUEUE($Q$)
 9:         $n = |\mathrm{range}(f)|$
10:         for $0 < i < j < n$ do                        ▷ Loop over all pairs of output clusters of $f$
11:             $f' = c_{i,j} \circ f$                               ▷ Merge clusters $i, j$ output $f$
12:             Point $p = \mathrm{Point}(\mathrm{x} = -\mathrm{H}(p_{f'(X_1)f'(X_2)})/2, \mathrm{y} = \mathrm{I}(p_{f'(X_1)f'(X_2);Y}), \mathrm{f} = f')$
13:             $d =$ PARETO_DISTANCE($p, P$)
14:             $P \leftarrow$ PARETO_ADD($p, P$)  ▷ Keep track of point and clustering in Pareto set
15:             with probability $e^{-d/\varepsilon}$, $Q \leftarrow$ ENQUEUE($f', Q$)
16:
        return $P$
```


(a)

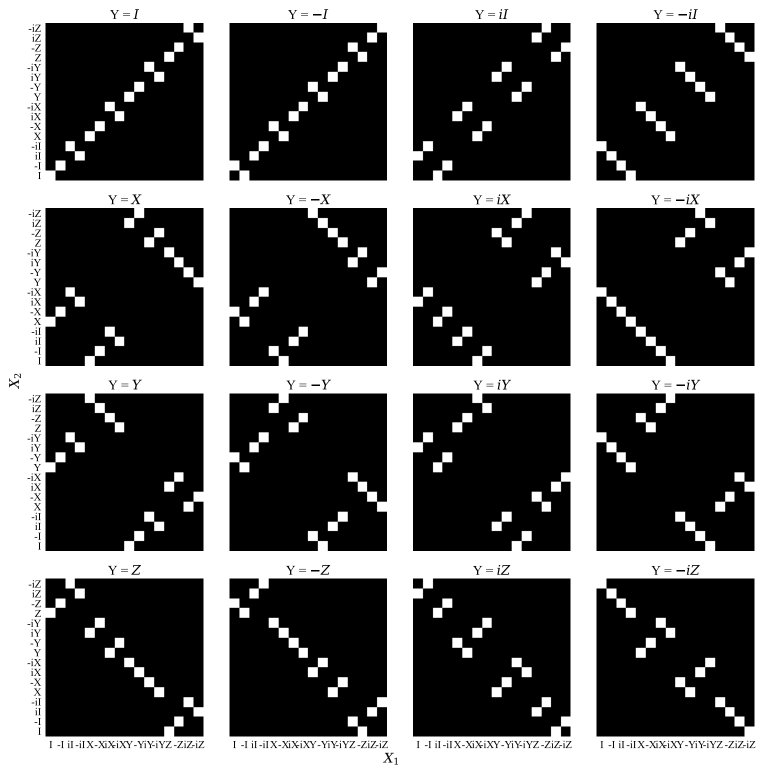

(b)

Figure C.1: Joint distribution $p_{X_1,X_2;Y}$ for the (a) $(\mathbb{Z}/40\mathbb{Z})^{\times}$ group and (b) the Pauli group.

References

[AS18] Alessandro Achille and Stefano Soatto. "Emergence of invariance and disentanglement in deep representations". In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 1947–1980.

[AB97] Dorit Aharonov and Michael Ben-Or. "Fault-tolerant quantum computation with constant error". In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. 1997, pp. 176–188.

[AL18] Tameem Albash and Daniel A Lidar. "Adiabatic quantum computation". In: *Reviews of Modern Physics* 90.1 (2018), p. 015002.

[Alb+18] Victor V. Albert et al. "Performance and structure of single-mode bosonic codes". In: *Physical Review A* 97 (3 Mar. 2018), p. 032346. DOI: 10.1103/PhysRevA.97.032346.

[AF18] Alexander A Alemi and Ian Fischer. "TherML: thermodynamics of machine learning". In: *arXiv preprint arXiv:1807.04162* (2018).

[Ale+17] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. "Deep variational information bottleneck". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[And+04] Periklis Andritsos, Panayiotis Tsaparas, Renée J. Miller, and Kenneth C. Sevcik. "LIMBO: scalable clustering of categorical data". In: *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology*. Heraklion, Crete, Greece, 2004, pp. 123–146. ISBN: 9783540212003. DOI: 10.1007/978-3-540-24741-8_9.

[Awa+15] Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali K Sinop. "The hardness of approximation of euclidean k-Means". In: *31st International Symposium on Computational Geometry (SoCG 2015)*. 2015.

[Ban+05] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, Joydeep Ghosh, and John Lafferty. "Clustering with Bregman divergences." In: *Journal of Machine Learning Research* 6.10 (2005).

[Bat19] Gérard Battail. "Error-correcting codes and information in biology". In: *BioSystems* 184 (2019), p. 103987.

[Ber+15] Dominic W Berry, Andrew M Childs, Richard Cleve, Robin Kothari, and Rolando D Somma. “Simulating hamiltonian dynamics with a truncated Taylor series”. In: *Physical Review Letters* 114.9 (2015), p. 090502.

[BRS17] Alex Bocharov, Martin Roetteler, and Krysta M. Svore. “Factoring with qutrits: Shor’s algorithm on ternary and metaplectic quantum architectures”. In: *Physical Review A* 96 (1 July 2017), p. 012306. DOI: 10.1103/PhysRevA.96.012306.

[Bri+09] Hans J Briegel, David E Browne, Wolfgang Dür, Robert Raussendorf, and Maarten Van den Nest. “Measurement-based quantum computation”. In: *Nature Physics* 5.1 (2009), pp. 19–26.

[BHC04] Kenneth R Brown, Aram W Harrow, and Isaac L Chuang. “Arbitrarily accurate composite pulse sequences”. In: *Physical Review A* 70.5 (2004), p. 052318.

[Bro+20] Tom Brown et al. “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[BM10] Lars Buesing and Wolfgang Maass. “A spiking neuron as information bottleneck”. In: *Neural Computation* 22.8 (2010), pp. 1961–1992.

[CTV17] Earl T Campbell, Barbara M Terhal, and Christophe Vuillot. “Roads towards fault-tolerant universal quantum computation”. In: *Nature* 549.7671 (2017), pp. 172–179.

[CV20a] Daan Camps and Roel Van Beeumen. “Approximate quantum circuit synthesis using block encodings”. In: *Physical Review A* 102.5 (2020), p. 052411.

[CV22] Daan Camps and Roel Van Beeumen. “FABLE: Fast Approximate Quantum Circuits for Block-Encodings”. In: *arXiv preprint arXiv:2205.00081* (2022).

[CR20] Rui Chao and Ben W. Reichardt. “Flag Fault-Tolerant Error Correction for any Stabilizer Code”. In: *PRX Quantum* 1 (1 Sept. 2020), p. 010302. DOI: 10.1103/PRXQuantum.1.010302.

[CV16] Avhishek Chatterjee and Lav R. Varshney. “Energy-reliability limits in nanoscale circuits”. In: *2016 Information Theory and Applications Workshop (ITA)*. 2016, pp. 1–6. DOI: 10.1109/ITA.2016.7888169.

[CV20b] Avhishek Chatterjee and Lav R. Varshney. “Energy-Reliability Limits in Nanoscale Feedforward Neural Networks and Formulas”. In: *IEEE Journal on Selected Areas in Information Theory* 1.1 (2020), pp. 250–266. DOI: 10.1109/JSAIT.2020.2981889.

[CS18] Pratik Chaudhari and Stefano Soatto. “Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks”. In: *2018 Information Theory and Applications Workshop (ITA)*. IEEE. 2018, pp. 1–10.

[CGW13] Andrew M Childs, David Gosset, and Zak Webb. “Universal computation by multiparticle quantum walk”. In: *Science* 339.6121 (2013), pp. 791–794.

[Chi+21] Andrew M Childs, Yuan Su, Minh C Tran, Nathan Wiebe, and Shuchen Zhu. "Theory of trotter error with commutator scaling". In: *Physical Review X* 11.1 (2021), p. 011020.

[CW12] Andrew M Childs and Nathan Wiebe. "Hamiltonian simulation using linear combinations of unitary operations". In: *Quantum Information & Computation* 12.11-12 (2012), pp. 901–924.

[Coh+99] Neil Cohen, TS Sriram, Norm Leland, David Moyer, Steve Butler, and Robert Flatley. "Soft error considerations for deep-submicron CMOS circuit applications". In: *International Electron Devices Meeting 1999. Technical Digest (Cat. No. 99CH36318)*. IEEE. 1999, pp. 315–318.

[Con+19] Tom Conte, Erik DeBenedictis, Natesh Ganesh, Todd Hylton, John Paul Strachan, R Stanley Williams, Alexander Alemi, Lee Altenberg, Gavin Crooks, James Crutchfield, et al. "Thermodynamic computing". In: *arXiv preprint arXiv:1911.01968* (2019).

[Cop02] Don Coppersmith. "An approximate Fourier transform useful in quantum factoring". In: *arXiv preprint quant-ph/0201067* (2002).

[CT06] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. 2nd ed. Wiley, 2006. ISBN: 9780471748816.

[Deu85] David Deutsch. "Quantum theory, the Church–Turing principle and the universal quantum computer". In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (1985), pp. 97–117.

[Deu89] David Elieser Deutsch. "Quantum computational networks". In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 425.1868 (1989), pp. 73–90.

[Dub99] Elena Dubrova. "Multiple-valued logic in VLSI: challenges and opportunities". In: *Proceedings of NORCHIP*. Vol. 99. 1999. 1999, pp. 340–350.

[EG17] El Mahdi El Mhamdi and Rachid Guerraoui. "When Neurons Fail". In: *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2017, pp. 1028–1037.

[EFR74] George Epstein, Gideon Frieder, and David C. Rine. "The development of multiple-valued logic as related to computer science". In: *Computer* 7.9 (1974), pp. 20–32. DOI: 10.1109/MC.1974.6323304.

[Ess+16] Steven K. Esser et al. "Convolutional networks for fast, energy-efficient neuromorphic computing". In: *Proceedings of the National Academy of Sciences* 113.41 (2016), pp. 11441–11446. ISSN: 0027-8424. DOI: 10.1073/pnas.1604850113.

[EP98] W. Evans and N. Pippenger. "On the maximum tolerable noise for reliable computation by formulas". In: *IEEE Transactions on Information Theory* 44.3 (1998). ISSN: 0018-9448. DOI: 10.1109/18.669417.

[ES99] W.S. Evans and L.J. Schulman. "Signal propagation and noisy circuits". In: *IEEE Transactions on Information Theory* 45.7 (1999), pp. 2367–2373. ISSN: 0018-9448. DOI: 10.1109/18.796377.

[ES03] W.S. Evans and L.J. Schulman. “On the maximum tolerable noise of k-input gates for reliable computation by formulas”. In: *IEEE Transactions on Information Theory* 49.11 (2003), pp. 3094–3098. DOI: 10.1109/TIT.2003.818405.

[Eva+00] William Evans, Claire Kenyon, Yuval Peres, and Leonard J Schulman. “Broadcasting on trees and the Ising model”. In: *The Annals of Applied Probability* 10.2 (2000), pp. 410–433.

[Far+01] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. “A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem”. In: *Science* 292.5516 (2001), pp. 472–475.

[Far+00] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. “Quantum computation by adiabatic evolution”. In: *arXiv preprint quant-ph/0001106* (2000).

[Fed89] T. Feder. “Reliable computation by networks in the presence of noise”. In: *IEEE Transactions on Information Theory* 35.3 (1989), pp. 569–571. ISSN: 0018-9448. DOI: 10.1109/18.30978.

[Fey85] Richard P Feynman. “Quantum mechanical computers”. In: *Optics News* 11.2 (1985), pp. 11–20.

[FBB08] Ila R Fiete, Yoram Burak, and Ted Brookings. “What grid cells convey about rat location”. In: *Journal of Neuroscience* 28.27 (2008), pp. 6858–6871.

[Fis20] Ian Fischer. “The conditional entropy bottleneck”. In: *Entropy* 22.9 (2020), p. 999. DOI: 10.3390/e22090999.

[For81] Donald R Forsdyke. “Are introns in-series error-detecting sequences?” In: *Journal of theoretical biology* 93.4 (1981), pp. 861–866.

[FT82] Edward Fredkin and Tommaso Toffoli. “Conservative logic”. In: *International Journal of Theoretical Physics* 21.3-4 (1982), pp. 219–253.

[Gad+06] Matthew J Gadlage, Paul H Eaton, Joseph M Benedetto, Marty Carts, Vivian Zhu, and Thomas L Turflinger. “Digital device error rate trends in advanced CMOS technologies”. In: *IEEE Transactions on Nuclear Science* 53.6 (2006), pp. 3466–3471.

[GQF05] J.B. Gao, Yan Qi, and J.A.B. Fortes. “Bifurcations and fundamental error bounds for fault-tolerant computations”. In: *IEEE Transactions on Nanotechnology* 4.4 (2005), pp. 395–402. DOI: 10.1109/TNANO.2005.851289.

[Gil+19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. “Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 193–204.

[Gob65] T. Goblick. “Theoretical limitations on the transmission of data from analog sources”. In: *IEEE Transactions on Information Theory* 11.4 (1965), pp. 558–567. DOI: 10.1109/TIT.1965.1053821.

[Got22] Daniel Gottesman. "Opportunities and Challenges in Fault-Tolerant Quantum Computation". In: *arXiv preprint arXiv:2210.15844* (2022). arXiv: 2210.15844 [quant-ph].

[Gro96] Lov K. Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on theory of computing*. 1996, pp. 212–219.

[Gro97] Lov K. Grover. "Quantum mechanics helps in searching for a needle in a haystack". In: *Physical Review Letters* 79.2 (1997), p. 325.

[Gro98] Lov K. Grover. "Quantum computers can search rapidly by using almost any transformation". In: *Physical Review Letters* 80.19 (1998), p. 4329.

[Haf+05] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. "Microstructure of a spatial map in the entorhinal cortex". In: *Nature* 436.7052 (2005), pp. 801–806.

[HW91] B. Hajek and T. Weller. "On the maximum tolerable noise for reliable computation by formulas". In: *IEEE Transactions on Information Theory* 37.2 (1991), pp. 388–391. ISSN: 0018-9448. DOI: 10.1109/18.75261.

[HWD17] S. Hassanpour, D. Wuebben, and A. Dekorsy. "Overview and investigation of algorithms for the information bottleneck method". In: *SCC 2017; 11th International ITG Conference on Systems, Communications and Coding*. Feb. 2017, pp. 1–6.

[HSM93] Neal A. Hessler, Aneil M. Shirke, and Roberto Malinow. "The probability of transmitter release at a mammalian central synapse". In: *Nature* 366.6455 (Dec. 1993), pp. 569–572. ISSN: 1476-4687. DOI: 10.1038/366569a0.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080.

[IIW15] Marat Ibragimov, Rustam Ibragimov, and Johan Walden. *Heavy-tailed Distributions and Robustness in Economics and Finance*. Vol. 214. Springer, 2015.

[Imp04] François Impens. "Fine-grained fault-tolerance: Reliability as a fungible resource". MA thesis. Massachusetts Institute of Technology, 2004.

[Ind+11] Giacomo Indiveri et al. "Neuromorphic Silicon Neuron Circuits". In: *Frontiers in Neuroscience* 5 (2011), p. 73. ISSN: 1662-453X. DOI: 10.3389/fnins.2011.00073.

[JKC21] Sae Byeok Jo, Joohoon Kang, and Jeong Ho Cho. "Recent advances on multivalued logic gates: a materials perspective". In: *Advanced Science* 8.8 (2021), p. 2004216.

[JEP+21] John Jumper, Richard Evans, Alexander Pritzel, et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[Kem+08] Julia Kempe, Oded Regev, Falk Unger, and Ronald De Wolf. "Upper bounds on the noise threshold for fault-tolerant quantum computing". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2008, pp. 845–856.

[Kir+08] Andy Kirou, Błażej Ruszczycki, Markus Walser, and Neil F Johnson. "Computational modeling of collective human behavior: The example of financial markets". In: *Computational Science–ICCS 2008: 8th International Conference, Kraków, Poland, June 23-25, 2008, Proceedings, Part I 8*. Springer. 2008, pp. 33–41.

[Kit03] A.Yu. Kitaev. "Fault-tolerant quantum computation by anyons". In: *Annals of Physics* 303.1 (Jan. 2003), pp. 2–30. ISSN: 0003-4916. DOI: 10.1016/s0003-4916(02)00018-0.

[KL97] Emanuel Knill and Raymond Laflamme. "Theory of quantum error-correcting codes". In: *Physical Review A* 55.2 (1997), p. 900.

[KLZ98] Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. "Resilient quantum computation: error models and thresholds". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (Jan. 1998), pp. 365–384. ISSN: 1471-2946. DOI: 10.1098/rspa.1998.0166.

[KTV18] Artemy Kolchinsky, Brendan D Tracey, and Steven Van Kuyk. "Caveats for information bottleneck in deterministic scenarios". In: *International Conference on Learning Representations*. 2018.

[KSG04] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. "Estimating mutual information". In: *Physical Review E* 69.6 (2004), p. 066138.

[KY14] B. M. Kurkoski and H. Yagi. "Quantization of binary-input discrete memoryless channels". In: *IEEE Transactions on Information Theory* 60.8 (Aug. 2014), pp. 4544–4552. ISSN: 1557-9654. DOI: 10.1109/TIT.2014.2327016.

[Lan61] Rolf Landauer. "Irreversibility and heat generation in the computing process". In: *IBM Journal of Research and Development* 5.3 (1961), pp. 183–191.

[Lau06] Dietlinde Lau. *Function algebras on finite sets: Basic course on many-valued logic and clone theory*. Springer Science & Business Media, 2006.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[Lev86] Malcolm H Levitt. "Composite pulses". In: *Progress in Nuclear Magnetic Resonance Spectroscopy* 18.2 (1986), pp. 61–122.

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft COCO: common objects in context". In: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755.

[Lin+20] Xi Lin, Zhiyuan Yang, Qingfu Zhang, and Sam Kwong. "Controllable pareto multi-task learning". In: *arXiv preprint arXiv:2010.06313* (2020).

[Liu+19] Tao Liu, Wujie Wen, Lei Jiang, Yanzhi Wang, Chengmo Yang, and Gang Quan. "A Fault-Tolerant Neural Network Architecture". In: *2019 56th ACM/IEEE Design Automation Conference (DAC)*. 2019, pp. 1–6.

[Lov+10] Neil B Lovett, Sally Cooper, Matthew Everitt, Matthew Trevers, and Viv Kendon. “Universal quantum computation using the discrete-time quantum walk”. In: *Physical Review A* 81.4 (2010), p. 042330.

[LC17] Guang Hao Low and Isaac L Chuang. “Optimal Hamiltonian simulation by quantum signal processing”. In: *Physical Review Letters* 118.1 (2017), p. 010501.

[LC19] Guang Hao Low and Isaac L Chuang. “Hamiltonian simulation by qubitization”. In: *Quantum* 3 (2019), p. 163.

[LYC14] Guang Hao Low, Theodore J Yoder, and Isaac L Chuang. “Optimal arbitrarily accurate composite pulse sequences”. In: *Physical Review A* 89.2 (2014), p. 022341.

[LYC16] Guang Hao Low, Theodore J Yoder, and Isaac L Chuang. “Methodology of resonant equiangular composite quantum gates”. In: *Physical Review X* 6.4 (2016), p. 041067.

[Mak19] Anuran Makur. “Information contraction and decomposition”. PhD thesis. Massachusetts Institute of Technology, 2019.

[MMP19] Anuran Makur, Elchanan Mossel, and Yury Polyanskiy. “Broadcasting on random directed acyclic graphs”. In: *IEEE Transactions on Information Theory* 66.2 (2019), pp. 780–812.

[Mar+23] John M Martyn, Yuan Liu, Zachary E Chin, and Isaac L Chuang. “Efficient fully-coherent quantum signal processing algorithms for real-time dynamics simulation”. In: *The Journal of Chemical Physics* 158.2 (2023), p. 024106.

[Mar+21] John M Martyn, Zane M Rossi, Andrew K Tan, and Isaac L Chuang. “Grand unification of quantum algorithms”. In: *PRX Quantum* 2.4 (2021), p. 040203.

[MFC23] Trevor McCourt, Ila R Fiete, and Isaac L Chuang. “Noisy dynamical systems evolve error correcting codes and modularity”. In: *arXiv preprint arXiv:2303.14448* (2023).

[MP43] Warren S. McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The Bulletin of Mathematical Biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259.

[McM23] Peter L. McMahon. “The physics of optical computing”. In: *Nature Reviews Physics* 5.12 (Oct. 2023), pp. 717–734. ISSN: 2522-5820. DOI: 10.1038/s42254-023-00645-5. URL: http://dx.doi.org/10.1038/s42254-023-00645-5.

[MON89] M. Morisue, K. Oochi, and M. Nishizawa. “A novel ternary logic circuit using Josephson junction”. In: *IEEE Transactions on Magnetics* 25.2 (1989), pp. 845–848. DOI: 10.1109/20.92418.

[Nav+20] Aviv Navon, Aviv Shamsian, Ethan Fetaya, and Gal Chechik. “Learning the Pareto Front with Hypernetworks”. In: *International Conference on Learning Representations*. 2020.

[NBR04] Ilya Nemenman, William Bialek, and Rob de Ruyter van Steveninck. “Entropy and information in neural spike trains: progress on the sampling problem”. In: *Physical Review E* 69 (5 May 2004), p. 056111. DOI: 10.1103/PhysRevE.69.056111. URL: https://link.aps.org/doi/10.1103/PhysRevE.69.056111.

[NSB02] Ilya Nemenman, F. Shafee, and William Bialek. “Entropy and Inference, Revisited”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press, 2002, pp. 471–478. URL: https://proceedings.neurips.cc/paper/2001/file/d46e1fcf4c07ce4a69ee07e4134bcef1-Paper.pdf.

[NSY92] C. Neti, M.H. Schneider, and E.D. Young. “Maximally fault tolerant neural networks”. In: *IEEE Transactions on Neural Networks* 3.1 (1992), pp. 14–23. DOI: 10.1109/72.105414.

[Neu56] John von Neumann. “Probabilistic logics and the synthesis of reliable organisms from unreliable components”. In: *Automata Studies* 34 (1956), pp. 43–98.

[Neu16] John von Neumann. “Probabilistic Logics and the Synthesis of Reliable Organisms From Unreliable Components”. In: *Automata Studies. (AM-34), Volume 34*. Ed. by C. E. Shannon and J. McCarthy. Princeton University Press, 2016, pp. 43–98. DOI: doi:10.1515/9781400882618-003.

[NS21] Vudtiwat Ngampruetikorn and David J Schwab. “Perturbation theory for the information bottleneck”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[NC10] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[Oat15] AS Oates. “Reliability of silicon integrated circuits”. In: *Reliability Characterisation of Electrical and Electronic Systems*. Elsevier, 2015, pp. 115–141.

[Pan03] Liam Paninski. “Estimation of entropy and mutual information”. In: *Neural Computation* 15.6 (2003), pp. 1191–1253.

[PTL93] Fernando Pereira, Naftali Tishby, and Lillian Lee. “Distributional clustering of English words”. In: *Proceedings of the 31st annual meeting on Association for Computational Linguistics*. 1993, pp. 183–190.

[Pip85] Nicholas Pippenger. “On networks of noisy gates”. In: *26th Annual Symposium on Foundations of Computer Science (SFCS 1985)* (1985), pp. 30–38. DOI: 10.1109/SFCS.1985.41.

[Pip88] Nicholas Pippenger. “Reliable computation by formulas in the presence of noise”. In: *IEEE Transactions on Information Theory* 34.2 (1988), pp. 194–197. ISSN: 0018-9448. DOI: 10.1109/18.2628.

[PW17] Yury Polyanskiy and Yihong Wu. “Strong data-processing inequalities for channels and Bayesian networks”. In: *Convexity and Concentration*. Vol. 161. Springer, 2017, pp. 211–249.

[Poo+19] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. “On variational bounds of mutual information”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5171–5180.

[Pos41] Emil L. Post. “The two-valued iterative systems of mathematical logic”. In: *Annals of Math, Studies* 5 (1941).

[Pre98] John Preskill. “Reliable quantum computers”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (1998), pp. 385–410.

[RBB03] Robert Raussendorf, Daniel E Browne, and Hans J Briegel. “Measurement-based quantum computation on cluster states”. In: *Physical Review A* 68.2 (2003), p. 022312.

[RV18] Danilo Jimenez Rezende and Fabio Viola. “Taming VAEs”. In: *arXiv preprint arXiv:1810.00597* (2018).

[Rin14] David C Rine. *Computer Science and Multiple-Valued Logic: Theory and Applications*. Elsevier, 2014.

[RC23] Zane M Rossi and Isaac L Chuang. “Semantic embedding for quantum algorithms”. In: *arXiv preprint arXiv:2304.14392* (2023).

[Ruc+89] U. Ruckert, I. Kreuzer, V. Tryba, and K. Goser. “Fault-tolerance of associative memories based on neural networks”. In: *Proceedings. VLSI and Computer Peripherals. COMPEURO 89*. 1989, pp. 1/52–1/55. DOI: 10.1109/CMPEUR.1989.93343.

[SKR95] Alessandro Saffiotti, Kurt Konolige, and Enrique H Ruspini. “A multivalued logic approach to integrating planning and control”. In: *Artificial intelligence* 76.1-2 (1995), pp. 481–526.

[Sax+19] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. “On the information bottleneck theory of deep learning”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2019.12 (2019), p. 124020.

[SPW09] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. “DRAM errors in the wild: a large-scale field study”. In: *ACM SIGMETRICS Performance Evaluation Review* 37.1 (2009), pp. 193–204.

[SC90] C.H. Sequin and R.D. Clay. “Fault tolerance in artificial neural networks”. In: *1990 IJCNN International Joint Conference on Neural Networks*. 1990, 703–708 vol.1. DOI: 10.1109/IJCNN.1990.137651.

[Sha+08] Naresh R Shanbhag, Subhasish Mitra, Gustavode de Veciana, Michael Orshansky, Radu Marculescu, Jaijeet Roychowdhury, Douglas Jones, and Jan M Rabaey. “The search for alternative computational paradigms”. In: *IEEE Design & Test of Computers* 25.4 (2008), pp. 334–343.

[Sha+18] Naresh R Shanbhag, Naveen Verma, Yongjune Kim, Ameya D Patil, and Lav R Varshney. “Shannon-inspired statistical computing for the nanoscale era”. In: *Proceedings of the IEEE* 107.1 (2018), pp. 90–107.

[Sha48] Claude Elwood Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[Sho96] P.W. Shor. "Fault-tolerant quantum computation". In: *Proceedings of 37th Conference on Foundations of Computer Science*. 1996, pp. 56–65. DOI: 10.1109/SFCS.1996.548464.

[Sho94] Peter W Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. IEEE. 1994, pp. 124–134.

[SWH20] Noah Shutty, Mary Wootters, and Patrick Hayden. "Tight Limits on Nonlocality from Nontrivial Communication Complexity; a.k.a. Reliable Computation with Asymmetric Gate Noise". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. 2020, pp. 206–217. DOI: 10.1109/FOCS46700.2020.00028.

[ST99] Noam Slonim and Naftali Tishby. "Agglomerative information bottleneck". In: *Advances in Neural Information Processing Systems 12*. The MIT Press, 1999, pp. 617–623.

[Smi81] Kenneth C. Smith. "The prospects for multivalued logic: A technology and applications view". In: *IEEE Transactions on Computers* 30.09 (1981), pp. 619–634.

[Smi88] Kenneth C. Smith. "A multiple valued logic: a tutorial and appreciation". In: *Computer* 21.4 (1988), pp. 17–27. DOI: 10.1109/2.48.

[SK93] WR Softky and C Koch. "The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs". In: *Journal of Neuroscience* 13.1 (1993), pp. 334–350. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.13-01-00334.1993.

[SF11] Sameet Sreenivasan and Ila Fiete. "Grid cells generate an analog error-correcting code for singularly precise neural computation". In: *Nature Neuroscience* 14.10 (Oct. 2011), pp. 1330–1337. ISSN: 1546-1726. DOI: 10.1038/nn.2901.

[SW94] Charles F. Stevens and Yanyan Wang. "Changes in reliability of synaptic function as a mechanism for plasticity". In: *Nature* 371.6499 (Oct. 1994), pp. 704–707. ISSN: 1476-4687. DOI: 10.1038/371704a0.

[Sti20] Susanne Still. "Thermodynamic Cost and Benefit of Memory". In: *Physical Review Letters* 124 (5 Feb. 2020), p. 050601. DOI: 10.1103/PhysRevLett.124.050601.

[SB04] Susanne Still and William Bialek. "How many clusters? An information-theoretic perspective". In: *Neural Computation* 16.12 (2004), pp. 2483–2506. ISSN: 0899-7667. DOI: 10.1162/0899766042321751.

[SS17] DJ Strouse and David J Schwab. "The deterministic information bottleneck". In: *Neural Computation* 29.6 (2017), pp. 1611–1630. ISSN: 0899-7667. DOI: 10.1162/neco_a_00961.

[SS19] DJ Strouse and David J Schwab. "The information bottleneck and geometric clustering". In: *Neural Computation* 31.3 (2019), pp. 596–612.

[Svo+05] K.M. Svore, A.W. Cross, I.L. Chuang, and A.V. Aho. "A flow-map model for analyzing pseudothresholds in fault-tolerant quantum computing". In: *arXiv e-prints*, quant-ph/0508176 (Aug. 2005), quant–ph/0508176. arXiv: quant-ph/0508176 [quant-ph].

[TC23] Andrew K Tan and Isaac L Chuang. "Resource savings from fault-tolerant circuit design". In: *arXiv preprint arXiv:2311.02132* (2023).

[THC23] Andrew K Tan, Matthew Ho, and Isaac L Chuang. "On reliable computation over larger alphabets". In: *arXiv preprint arXiv:2306.13262* (2023).

[Tan+23a] Andrew K Tan, Yuan Liu, Minh C Tran, and Isaac L Chuang. "Error Correction of Quantum Algorithms: Arbitrarily Accurate Recovery Of Noisy Quantum Signal Processing". In: *arXiv preprint arXiv:2301.08542* (2023).

[TTC22] Andrew K Tan, Max Tegmark, and Isaac L Chuang. "Pareto-optimal clustering with the primal deterministic information bottleneck". In: *Entropy* 24.6 (2022), p. 771.

[Tan+23b] Andrew K. Tan, Yuan Liu, Minh C. Tran, and Isaac L. Chuang. "Perturbative model of noisy quantum signal processing". In: *Physical Review A* 107 (4 Apr. 2023), p. 042429. DOI: 10.1103/PhysRevA.107.042429.

[TW19] Max Tegmark and Tailin Wu. "Pareto-optimal data compression for binary classification tasks". In: *Entropy* 22.1 (2019). ISSN: 1099-4300. DOI: 10.3390/e22010007.

[Ter15] Barbara M Terhal. "Quantum error correction for quantum memories". In: *Reviews of Modern Physics* 87.2 (2015), p. 307.

[Tha+05] Darshan D Thaker, Rajeevan Amirtharajah, Francois Impens, Isaac L Chuang, and Frederic T Chong. "Recursive TMR: Scaling fault tolerance in the nanoscale era". In: *IEEE Design & Test of Computers* 22.4 (2005), pp. 298–305.

[Tha+08] Darshan D Thaker, Francois Impens, Isaac L Chuang, Rajeevan Amirtharajah, and Frederic T Chong. "On Using Recursive TMR as a Soft Error Mitigation Technique". In: *Proceedings of International Conference on Parallel Processing (ICPP)*. 2008, pp. 10–15.

[TPB00] Naftali Tishby, Fernando C Pereira, and William Bialek. "The information bottleneck method". In: *arXiv* (2000). eprint: physics/0004057.

[TZ15] Naftali Tishby and Noga Zaslavsky. "Deep learning and the information bottleneck principle". In: *2015 IEEE Information Theory Workshop (ITW)*. IEEE. 2015, pp. 1–5.

[TG17] Cesar Torres-Huitzil and Bernard Girau. "Fault and Error Tolerance in Neural Networks: A Review". In: *IEEE Access* 5 (2017), pp. 17322–17341. DOI: 10.1109/ACCESS.2017.2742698.

[Tur36] Alan M. Turing. “On computable numbers, with an application to the Entscheidungsproblem”. In: *J. of Math* 58.345-363 (1936), p. 5.

[Two+21] Colin R. Twomey, Gareth Roberts, David H. Brainard, and Joshua B. Plotkin. “What we talk about when we talk about colors”. In: *Proceedings of the National Academy of Sciences* 118.39 (2021). ISSN: 0027-8424. DOI: 10.1073/pnas.2109237118.

[Ung07] Falk Unger. “Noise threshold for universality of 2-input gates”. In: *2007 IEEE International Symposium on Information Theory*. 2007, pp. 1901–1905. DOI: 10.1109/ISIT.2007.4557152.

[Ung10] Falk Unger. “Better gates can make fault-tolerant computation impossible.” In: *Electron. Colloquium Comput. Complex.* Vol. 17. 2010, p. 164.

[WA08] Fan Wang and Vishwani D Agrawal. “Soft error rate determination for nanometer CMOS VLSI logic”. In: *2008 40th Southeastern Symposium on System Theory (SSST)*. IEEE. 2008, pp. 324–328.

[Wan+20] Yuchen Wang, Zixuan Hu, Barry C Sanders, and Sabre Kais. “Qudits and high-dimensional quantum computing”. In: *Frontiers in Physics* 8 (2020), p. 589504.

[Wan+18] Zhongrui Wang et al. “Fully memristive neural networks for pattern classification with unsupervised learning”. In: *Nature Electronics* 1.2 (Feb. 2018), pp. 137–145. ISSN: 2520-1131. DOI: 10.1038/s41928-018-0023-2.

[Wei61] Martin H. Weik. “The ENIAC Story”. In: *Ordnance* 45.244 (1961), pp. 571–575. (Visited on 12/18/2023).

[WC63] Shmuel Winograd and Jack D Cowan. *Reliable Computation in the Presence of Noise*. Cambridge, Mass: MIT Press Cambridge, Mass., 1963.

[Wu+20] Tailin Wu, Ian Fischer, Isaac L Chuang, and Max Tegmark. “Learnability for the information bottleneck”. In: *Uncertainty in Artificial Intelligence*. PMLR. 2020, pp. 1050–1060.

[Yek+12] Sergey Yekhanin et al. “Locally decodable codes”. In: *Foundations and Trends in Theoretical Computer Science* 6.3 (2012), pp. 139–255.

[YLC14] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. “Fixed-point quantum search with an optimal number of queries”. In: *Physical Review Letters* 113 (21 Nov. 2014), p. 210501. DOI: 10.1103/PhysRevLett.113.210501.

[ZK16] J. A. Zhang and B. M. Kurkoski. “Low-complexity quantization of discrete memoryless channels”. In: *2016 International Symposium on Information Theory and Its Applications (ISITA)*. Oct. 2016, pp. 448–452.

[Zlo+22] Alexander Zlokapa, Andrew K Tan, John M Martyn, Ila R Fiete, Max Tegmark, and Isaac L Chuang. “Biological error correction codes generate fault-tolerant neural networks”. In: *arXiv preprint arXiv:2202.12887* (2022).