PHOTONIC CIRCUITS AND PROBABILISTIC COMPUTING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF PHYSICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Dmitri Serguei Pavlichin
March 2014

This dissertation is online at: http://purl.stanford.edu/nh655cz4317

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Hideo Mabuchi, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Sebastian Doniach

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Itschak Weissman

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

This work explores computing in the noisy, stochastic, low-power setting, particularly with photonic systems. Stochasticity and noisy operation has been with computing since its earliest days, before the development of extremely reliable components (e.g. CMOS) enabled reliable digital and analog operation and relegated noise to a low level of abstraction. As the limits of current technologies are reached, as power consumption becomes a bottleneck for computing, and as novel stochastic algorithms are developed, a probabilistic approach extends usefully beyond the level of single components: from stochastic encoding of information to entire circuit architectures. Photonic systems - which can involve the interaction of a handful of photons with a handful of internal degrees of freedom like atomic states - provide a natural platform for consideration of noisy, low-power computing.

This is a work in two parts. In Part I I investigate applications of photonic circuits for power-efficient computation. I propose a circuit architecture for the decoding of low density parity-check codes that uses only optical waveguides, noisy optical switches, and is inherently tolerant of faults affecting its components; one of its most appealing features is the graceful degradation of performance and computation time as we lower the power in the laser that powers the device. I also introduce various design motifs that I hope will be useful to future optical engineers. Part II has a more information theoretic flavor. I consider several optics-inspired models for encoding information stochastically — e.g., in the distribution of optical power over multiple waveguides, or multiple frequency bands for a Gaussian channel, prove some optimality results about some of the schemes I propose, and discuss energy-efficient communication for these setups.

Acknowledgement

Thank you!

I am deeply thankful to Hideo Mabuchi, under whose tutelage I've come into my own. Thanks for your advice, encouragement, and patience for all of these years.

To Tsachy Weissman, who's given an ear to my wild-eyed ramblings as I've been learning information theory. Thank you for all of the chats and for your support.

To Persi Diaconis for the encouraging and fun conversations. To Thomas Cover, whom I had known only briefly after taking his class, but whose suggestions and ways were an inspiration to me.

To my bosom friends: Mikey — thanks for all the coffee. You can have my Coupa card and its massive endowment if I quit drinking for good. Also thanks for all the help on so many things for so long. Dan and Dan: I'm really glad we could spend these years together in California. Nancy: Thanks for moving all the way out here just to make sure I graduate! Thanks Yeong Dae for being a friend and the best roommate, better than any of the others by far! Maybe someday again! When I was still an undergrad, I talked to Gopal on the phone and he suggested I look into Hideo's group. It sounded unlikely to me, but what did I know. Charles: for all the late night chats, rather dangerous frisbee, and whiteboard parties. Nikolas, a great collaborator, who's also taught me a ton of computer stuff. Max Greenfeld, who taught me to milk goats. Kevin: I'm glad to have gotten a slice of a Mabuchilab classic in my time here. Joe: I got a lot out of our conversations. Hardeep: you were a great roommate too, on par with Yeong Dae even. Tony: you're pretty cool. To all of the Mabuchilab members and others over these years: ZK, Aaswath, Jeremy, Rolf, Ryan, Nina, Jie, Dodd, Nate, Orion, Chris. Thanks to all of you.

Mama and Papa: thanks for everything, and thanks for bringing me to this country. To all of my relatives in Kiev who helped raise me — I couldn't have done this without you.

Contents

Outline

This thesis explores computing in the noisy, low-power setting. Our discussion is grounded in and draws inspiration from photonic systems and circuits, but leads to some questions that are reasonably natural without reference to a specific physical setup.

This work is thus in two parts, each with a separate introduction. Part I investigates applications of photonic circuits for information processing tasks in the noisy, low-photon number regime. Part II presents an information theoretic perspective on communication setups that are inspired by photonic circuits, but (I hope) are interesting in their own right and arise in other practical settings, like flash memory cells. The parts can be read independently, but both spring from the desire to understand noisy, power-efficient computing.

Part I

Quantum optical devices

Introduction

Part I explores using optical circuits for information processing tasks. These systems, which can involve the interaction of a handful of photons with a handful of internal degrees of freedom like atomic states, provide a natural platform for [consideration] implementation of noisy, low-power computing. "Computing" here broadly means any kind of transformation of information represented in some way that is useful to someone (e.g. computation of Boolean functions, decoding the outputs of noisy communication channels). We will be more precise about the meaning of "noisy" and "low-power" later, but observe that noise has been with computing since its early days, before reliable components existed, requiring one to consider construction of reliable devices from unreliable parts (von Neumann, 1956)[1]. As computers components improved in reliability to the point of practical perfection, noise was banished to live in a low level of computing abstraction for decades, but is in some ways coming back as the limits of CMOS technology are reached, but also with the development of algorithms that live in the statistical setting (e.g. message-passing algorithms like belief propagation), are robust to noise, and can even benefit from it (e.g. algorithms requiring a source of randomness). These schemes find a ready home in stochastic computing architectures (e.g., see the thesis (Vigoda, 2003) for an extended discussion of this mapping for continuous-time analog circuits). We propose one such mapping between optical circuits and these graph-based algorithms in this work. Finally, "low-power" computing goes hand in hand with the noisy setting, but is another motivation for this work.

[1] These notes based on 1952 lectures by von Neumann established much of the theory of stochastic and fault-tolerant computing.

Chapter 1 reviews the physical models for photonic circuits in sufficient detail for someone to reproduce the simulations used in subsequent Chapters. I also present a scheme for the efficient simulation of certain kinds of optical circuits; this technique enabled me to do the simulations of Chapter 2 in time to defend.

Chapter 2 describes a photonic circuit for a decoder for low density parity-check (LDPC) codes and characterizes its performance through simulations, following closely and elaborating upon our work (Pavlichin and Mabuchi, 2013). This circuit uses all-optical components (waveguides, mirrors, noisy optical switches), is powered by a coherent input field (a laser), and operates autonomously (independently of any external (large and expensive) controller). The circuit is also a demonstration of an optical device robust to noise affecting its components. Its performance in terms of decoding probability and speed degrades smoothly as the input power is lowered.

Chapter 3 is a device bestiary of optical circuits and design motifs that I've come up with over the years. Some of these have already found use in our work (Pavlichin and Mabuchi, 2013) and I hope others will be useful to future optical engineers.

Chapter 1

Photonic systems and circuits for programmers

In this Chapter I briefly review photonic systems and circuits. This thesis does not contribute to these ideas, but applies them in Chapters 2 and 3 to design photonic circuits to efficiently do certain kinds of computations. The goal here is to describe the physical models for these systems in sufficient detail to enable someone other than me to code up the same simulations and to extend them to other kinds of circuits. We refer the reader to, e.g., (Wiseman, 2010) for a proper introduction.

This Chapter is organized as follows: I first state the "standard" physical model of quantum systems coupled to light in terms of a master equation. I then describe how to model circuits of such systems and how to simulate such circuits. I then propose an idea for doing these simulations efficiently for a certain kind of system[1].

1.1 Open quantum systems

We work in the framework of (Gough, 2008) and (Gough, 2009) for modeling the interactions of quantum systems with light. Their work was preceded by and built upon the work of (Hudson and Parthasarathy, 1984), (Carmichael, 1993), (Gardiner,

[1] This idea was applied in (Pavlichin and Mabuchi, 2013), but not described therein.

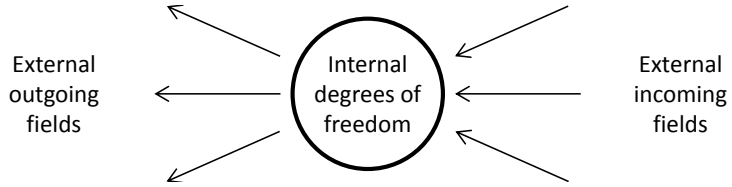

Figure 1.1: An open quantum system with internal degrees of freedom coupled to external fields.

1993), and (Barchielli, 2006). The mathematical foundations of this work lie in quantum stochastic differential equations developed by (Hudson and Parthasarathy, 1984; Gardiner, 1992; Parthasarathy, 1992).

1.1.1 SLH models

The basic model is shown in Figure 1.1. The system has some internal degrees of freedom coupled to n incoming and outgoing field modes. There are three parts to a physical description of the dynamics of this system:

- A system Hamiltonian H describing the time evolution of the internal degrees of freedom.
- An $n \times 1$ *coupling vector* $\mathbf{L}$ that specifies how external field modes interact with the internal degrees of freedom.
- An $n \times n$ unitary[2] *scattering matrix* $\mathbf{S}$ that specifies how incoming field modes "scatter" into outgoing field modes.

We refer to these three objects as an "SLH triplet."

We shall state in a second how these objects are used in describing system dynamics, but we first make some observations. There are always exactly as many outgoing

[2] $\mathbf{S}$ is unitary because of its role in the quantum stochastic differential equation describing the time propagator of a field annihilation operator; we defer the curious reader to (Kerckhoff, 2011) for an expository discussion. Roughly, time propagation operators must be unitary to preserve the L_2 norm of a wavefunction: $|\psi_t\rangle = U_t|\psi_0\rangle$, so $1 = \langle\psi_0|\psi_0\rangle = \langle\psi_t|\psi_t\rangle = \langle\psi_0|U_t^\dagger U_t|\psi_0\rangle \ \forall|\psi_0\rangle$, so $U_t^\dagger = U_t^{-1}$.

as incoming field modes. This may seem counterintuitive - for the example of atomic spontaneous emission there are photons coming out, but not in; the incoming mode supports the time reverse process of an atom absorbing a photon to go to an excited state. It's also possible that some outgoing mode carries only vacuum, but it is still an available mode for photons emitted by whatever it is that makes up the internal degrees of freedom.

There may be no optical modes at all ($n = 0$). In this case we call the system "closed" and the time evolution of its wavefunction is governed by the Hamiltonian according to the Schrödinger equation (and of its density matrix according to the von Neumann equation). There may be $n > 0$ modes, but no internal degrees of freedom. In this case we think of the system as "purely scattering," with a trivial Hamiltonian ($H = 0$), trivial coupling vector ($\mathbf{L} = \mathbf{0}_{n\times 1}$) and some scattering matrix. For example, we treat the beamsplitter as a purely scattering device in Chapters 2 and 3 [3]. The simplest purely scattering "device" is a single optical mode that undergoes a phase shift of ϕ, in which case $H = 0$, $\mathbf{L} = 0$, $\mathbf{S} = e^{i\phi}$.

The time evolution of the internal degrees of freedom of our open quantum system is governed by the master equation:

$$\dot{\rho}_t = -i[H, \rho_t] + \sum_{i=1}^{n} \left(L_i \rho_t L_i^\dagger - \frac{1}{2}\{L_i^\dagger L_i, \rho_t\} \right) \tag{1.1}$$

where ρ is the density matrix for the internal degrees of freedom, L_i is the i-th component of the coupling vector $\mathbf{L}$, and $[A, B] \equiv AB - BA$, $\{A, B\} \equiv AB + BA$ denote the commutator and anticommutator, respectively. Note that for $n = 0$ (i.e. no external fields), the master equation becomes the von Neumann equation. The scattering matrix $\mathbf{S}$ is nowhere to be found in (1.1), but will appear once we describe in Section 1.2 making circuits out of multiple open quantum systems and deriving an overall master equation for the circuit.

[3] The beamsplitter is a complicated physical object, but it has no degrees of freedom in our model, as we assume all of those internal dynamics are very fast and lump their action into the scattering matrix. For an example of a "slow" beamsplitter (made of cavities) where we can not ignore the internal dynamics, see Section 3.2.2.

1.1.2 Observable quantities

Recall that the density matrix ρ for a quantum system acts like a probability density function (pdf) p. p satisfies $\int p(x)dx = 1$ and we compute expectation values of random values by integrating against p, $\mathbb{E}[X] = \int X(x)p(x)dx$; The density matrix ρ satisfies $\operatorname{Tr}\rho = 1$ and we compute the expectation values of operators by tracing against ρ:

$$\langle X \rangle = \operatorname{Tr}[X\rho] \tag{1.2}$$

The density matrix thus tells us all there is to know about a quantum state in the same way as a probability density function determines all expectation values. For example, for a two-state system, the density matrix is a 2×2 matrix and we can compute the probability of finding the system in state $|1\rangle$ by computing $\operatorname{Tr}\Pi_1\rho$, where $\Pi_1 = |1\rangle\langle 1|$ is a projector[4]. We should remember that the X in (1.2) is an operator on a Hilbert space, so if we find some other operator Y that does not commute with X, in general $\langle XY \rangle \neq \langle YX \rangle$, unlike the case of expectation values over pdf's[5]

Note that our density matrix ρ is only keeping track of the internal degrees of freedom and not the external field modes. In a proper treatment of the subject, we would start with a closed system consisting of the "internal" and the "bath" (of field modes) degrees of freedom evolving according to the von Neumann equation, and then "trace over the bath" to derive the master equation (1.1), where the tracing corresponds to something like measuring the outgoing light at all times, but not the internal degrees of freedom. We once again point the reader to the references at the beginning of this section for a proper introduction.

In an actual experiment we are often interested in learning things about the internal degrees of freedom, but are not typically able to observe them directly - that is, we can not actually compute $\operatorname{Tr}[X\rho]$ for an observable X. Instead we interrogate them with an external field (e.g. a laser beam shining on an atom) and then let the outgoing light hit a photodetector, which collects "clicks" at a rate proportional to the light intensity and produces an electrical signal. We capture this by defining an

[4] This is the projection postulate of quantum mechanics.

[5] The generalization of probability theory to handle noncommutative random variables is given by the theory of free probability.

observable[6] $L_i^\dagger L_i$ for each of the n optical channels and measuring its expectation value:

$$\langle L_i^\dagger L_i \rangle = \mathrm{Tr}[L_i^\dagger L_i \rho_t] \sim \text{photodetection rate in } i\text{-th channel} \tag{1.3}$$

where $\sim$ means we are ignoring possible inefficiency of the photodetector and stochastic vacuum fluctuations (see (Gardiner and Collett, 1985) for a proper derivation of this, or the book (Wiseman, 2010)). The expression (1.3) gives the instantaneous photodetection rate at time t. Not all of the n field modes coupled to the system are typically available for observation. For example, spontaneously emitted photons are difficult to collect, as they are not emitted into a particular direction. One must thus engineer field-system interactions so that the photodetection rate in the channels one does have access to are informative about the internal degrees of freedom of interest. At the same time, coupling to external modes can be difficult to avoid and can make life more difficult. The external modes could also be phonons (in an optomechanical system these could correspond to vibrations of a mirror, say); the master equation formalism handles this case as well (see (Botter et al., 2012; Hamerly and Mabuchi, 2012))

1.2 Photonic circuits

A powerful feature of the SLH formalism described in Section 1.1.1 is its ability to describe open quantum systems interconnected to form circuits - letting the outgoing fields of one system form the incoming fields for another. Given the SLH triplets for a collection of quantum systems, one can use the Gough-James circuit algebra to compute a single SLH triplet for the entire circuit viewed as a single open quantum system. This is analogous to taking an arbitrary circuit of resistors, capacitors, and inductors to compute an overall impedance.

In this Section we review the circuit composition rules for systems arranged in series, in parallel, and with feedback. The circuit composition rules are roughly intuitive-looking, and having them enables us to think about designing useful optical

[6] This is manifestly Hermitian, so it is an observable.

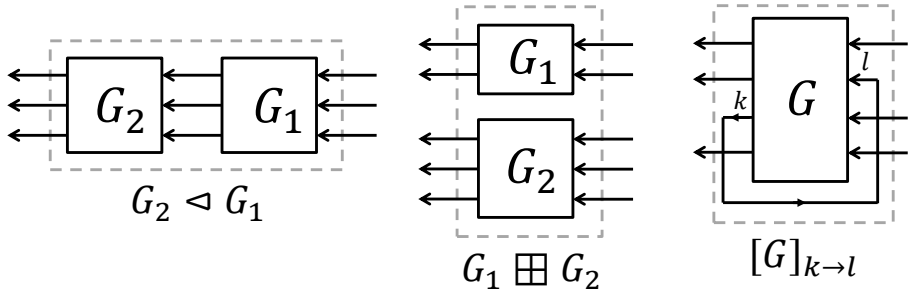

Figure 1.2: The three kinds of composition operations and their notation. Left to right: series product, concatenation product, feedback of output port k to input port l.

circuits, rather than studying individual systems. We apply the circuit composition rules in Chapters 2 and 3 to describe photonic circuits that might be useful for something.

The three kinds of compositions are shown in Figure 1.2 along with their corresponding notation. Let $G_1 = (\mathbf{S}_1, \mathbf{L}_1, H_1)$ and $G_2 = (\mathbf{S}_2, \mathbf{L}_2, H_2)$ be two open quantum systems, both coupled to the same number n of external field modes. The *series product* $G_2 \lhd G_1$ corresponds to a series connection of G_1 and G_2, where the n output modes of G_1 are the n input modes of G_2:

$$G_2 \lhd G_1 \equiv \left(\mathbf{S}_2\mathbf{S}_1,\ \mathbf{S}_2\mathbf{L}_1 + \mathbf{L}_2,\ H_1 + H_2 + \Im\left(\mathbf{L}_2^\dagger \mathbf{S}_2 \mathbf{L}_1\right)\right) \tag{1.4}$$

where $\Im(X) = \frac{1}{2i}(X - \bar{X})$ denotes the imaginary part. Note that the scattering matrix of G_2 now enters the Hamiltonian of the overall system (while the scattering matrix appears nowhere in the master equation (1.1) describing the dynamics of the internal degrees of freedom for a single open quantum system). Some observations: if $\mathbf{L}_1 = 0$ or $\mathbf{L}_2 = 0$, then the two systems are uncoupled, and the overall Hamiltonian is the sum $H_1 + H_2$, which makes sense. Second, the overall scattering matrix is the product $\mathbf{S}_2\mathbf{S}_1$, which is intuitive as well.

The *concatenation product* $G_2 \boxplus G_1$ corresponds to a parallel "connection" of G_1 and G_2, coupled to n_1 and n_2 external modes, respectively, so that the overall system

is coupled to $n_1 + n_2$ external modes:

$$G_1 \boxplus G_2 \equiv \left(\begin{pmatrix} \mathbf{S}_2 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_1 \end{pmatrix}, \begin{pmatrix} \mathbf{L}_1 \\ \mathbf{L}_2 \end{pmatrix}, H_1 + H_2 \right) \tag{1.5}$$

The concatenation product treats two non-interacting systems as a single one.

The *feedback* operation $[G]_{k\to l}$, where $G = (\mathbf{S}, \mathbf{L}, H)$ is coupled to n total modes, is a unary operation that corresponds to feeding back the k-th output mode to the l-th input mode, so that the overall system is coupled to $n-1$ modes:

$$[G]_{k\to l} \equiv (\mathbf{S}_{\rm fb}, \mathbf{L}_{\rm fb}, H_{\rm fb}) \tag{1.6}$$

where

$$\mathbf{S}_{\rm fb} = \mathbf{S}^{[k,l]} + \mathbf{S}^{[k]}_{:,l} \left(1 - S_{k,l}\right)^{-1} \mathbf{S}^{[l]}_{k,:} \tag{1.7}$$

$$\mathbf{L}_{\rm fb} = \mathbf{L}^{[k]} + \mathbf{S}^{[k]}_{:,l} \left(1 - S_{k,l}\right)^{-1} L_k \tag{1.8}$$

$$H_{\rm fb} = H + \Im \left[\left(\sum_{j=1}^{n} L_j^\dagger S_{j,l} \right) \left(1 - S_{k,l}\right)^{-1} L_k \right] \tag{1.9}$$

where $\mathbf{S}^{[k,l]}$ ($\mathbf{L}^{[k]}$) denotes the matrix (vector) formed by removing the k-th row and l-th column (k-th element) of $\mathbf{S}$ ($\mathbf{L}$), $\mathbf{S}_{:,l}$ ($\mathbf{S}_{k,:}$) denotes taking the l-th column (k-th row) of $\mathbf{S}$, and $S_{k,l}$ (L_k) denotes the (k,l)-th element of $\mathbf{S}$ (k-th element of $\mathbf{L}$).

We provide examples of these operations as they arise in Chapters 2 and 3.

1.2.1 Simulations

The master equation (1.1) is an ordinary differential equation (ODE), and so can be integrated numerically using standard techniques. Widely-used specialized software for doing this is available for Matlab (Tan, 1999) and for Python (Johansson et al., 2012, 2013).

This works OK for systems involving a few atomic state degrees of freedom and a small typical photon number[7], but can be impractical for even modest-sized quantum systems because the dimension of the density matrix ρ grows exponentially with the number of interacting components. This is because the Hilbert space $\mathcal{H}$ for two interacting systems with Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ of dimensions n_1 and n_2 is obtained via the tensor product $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ and the dimension is $n_1 n_2$. Things are actually a bit worse: if the density matrix is $n \times n$, then it has n^2 components to keep track of, so numerically integrating the master equation (1.1) requires us to compute a $n^2 \times n^2$ operator (the Liouvillian) corresponding to the right side of (1.1).

A common way of dealing with this issue is the same as the method for computing expectation values for complicated probabilistic models: somehow obtain many samples from the probability distribution of interest and compute an average over the samples. The samples are called quantum trajectories in this setting. Instead of propagating the density matrix ρ_t, we can instead propagate a collection of k wavefunctions $\{|\psi_{i,t}\rangle\}_{i=1,\ldots,k}$[8] and compute expectations values with respect to the trajectories. That is, we approximate $\mathrm{Tr}[X\rho]$ by the sample average

$$\langle X_t \rangle = \mathrm{Tr}[X_t \rho_t] \approx \sum_{i=1}^{k} \langle \psi_{i,t} | X_t | \psi_{i,t} \rangle \tag{1.10}$$

How do we obtain these trajectory samples? This is reviewed in Section 1.3 on efficient simulation techniques. How much easier is it to propagate trajectories than integrate the density matrix? Each wavefunction is an $n \times 1$ vector, rather than an $n \times n$ matrix, so it's that much easier, but is still exponentially hard in the number of interacting components.

Sometimes one has good reason to believe that interactions between some components in a circuit are negligible (in the sense that these systems never become much entangled), in which case one can get away with propagating a smaller subset of the

[7] In simulations, instead of using the infinite-dimensional Fock space spanned by the photon number states, we truncate the Fock space at some cutoff, chosen to be large enough to accommodate typical system behavior. See Chapter 3 for examples of SLH models involving Fock spaces.

[8] We can simulate the propagation of the wavefunctions one by one or in parallel, as the trajectories are independent of each other.

entire density matrix. This is the basic idea of Section 1.3. Another approach to efficient simulations is model reduction. Although the density matrix for a circuit may be large, it is often the case that only a few degrees of freedom really determine the dynamics. The goal of a model reduction approach is to identify those degrees of freedom and to construct an effective model that omits all of the non-important ones. An example of this approach to produce an effective master equation for a quantum optical system is found in (Tezak et al., 2012).

1.3 Data structures for efficient trajectory simulation

Suppose we would like to simulate a large photonic circuit with many interacting components. A full master equation integration is out of the question, so we use trajectory simulations. We may be independently interested in the trajectories, too, to understand the typical behavior of our device. Trajectory simulation can be slow for large optical systems, but we can sometimes make simplifying assumptions to make things easier. In this Section I propose a data structure for the efficient simulation of quantum trajectories in a particular setting - that of little entanglement between different components and sparse circuit connectivity. The method enabled us to run simulations involving tens of thousands of components in (Pavlichin and Mabuchi, 2013), which would have been a hopeless proposition for a more brute force approach.

A comprehensive review of quantum trajectory simulations is found in (Wiseman, 2010); here we provide a brief summary. Simulating a trajectory involves a continuous time evolution of the wavefunction and a point process of "jumps" - collapse operators acting on the wavefunction. Letting $\{L_a\}$ be the set of collapse operators, the instantaneous *rate* r_a of jump a is given by

$$r_a(t) = \|L_a\psi(t)\|^2 \tag{1.11}$$

(and the instantaneous probability of jump a in time dt is $r_a(t)dt$.) If jump a occurs at time t, the wavefunction goes to

$$\psi_{\text{post-jump}}(t) \sim L_a \psi_{\text{pre-jump}}(t) \tag{1.12}$$

where $\sim$ denotes equality up to normalization, and the normalization constant is $\sqrt{r_a(t)}$

The continuous time evolution is determined by the Schrödinger equation

$$i\frac{d\psi}{dt} = \left(H - \frac{i}{2}\sum_a L_a^\dagger L_a\right)\psi \tag{1.13}$$

where the quantity in parentheses is the "effective" non-Hermitian Hamiltonian (in the terminology of (Tan, 1999)).

This note describes a method for efficient simulation in the following setting:

- The system Hilbert space decomposes into many subsystems (groups of components) that tend not to get entangled in the course of a simulation. In the absence of entanglement between subsystems, we can propagate a collection of wavefunctions, one for each subsystem, rather than a single wavefunction over the (much larger) system Hilbert space.

- The circuit wavefunction tends to remain an eigenstate of the overall Hamiltonian in the course of a simulation. In the absence of Hamiltonian dynamics, the system time evolution is dominated by "jumps" — the L terms in the master equation for the circuit — and the jump rates (eq. (1.11)) do not vary in time between jumps (they must be updated only after each jump).

This nice setting seems hard to come by, but arises naturally when we consider large circuits composed of many simple components[9] coupled together via jump operators corresponding to beams propagating around the circuit. This is the case for the error-correcting circuits of our expander code decoding draft that are composed wholly of latch components (aside from state-less beam splitters and coherent inputs)

[9] e.g., components whose internal dynamics have been adiabatically eliminated into a nontrivial scattering matrix $\mathbf{S}$ and trivial $\mathbf{L}$ and H, as for the "probe" cavities in (Kerckhoff et al., 2010) and the LDPC circuit of Chapter 2.

that either switch or don't switch two beam paths via a scattering matrix, accept a set and reset input via two coupling/collapse terms, and have trivial Hamiltonian (0).

Simulation in this setting is equivalent to propagation of a continuous time Markov jump process. Where is the difficulty?

1.3.1 Where's the difficulty?

Consider the steps this simulation involves:

1. Evaluate the rate r_a for each jump operator L_a (eq. (1.11)).
2. Randomly draw the next jump from the set $\{L_a\}$ and randomly draw when it occurs.
3. Apply the next jump operator L_a to the wavefunction (eq. (1.12)).
4. repeat

For a naive implementation, steps 1 and 2 above require an amount of work proportional to the total number of collapse operators. This linear scaling can be slow: In the case of the simulations used in our work on expander code decoding draft (Pavlichin and Mabuchi, 2013), the circuits have a 60,000-dimensional Hilbert space and involve 100,000 jump terms (when fan-in/fan-out is included, this is closer to 1M jump terms).

1.3.2 We can do better

We can improve on this in two ways:

- Exploit the connectivity structure of the circuit to reduce the number of jump rates that must be recomputed (via eq. (1.11)) after a jump is applied.
- Use a data structure that enables efficient sampling of the next jump operator and jump time.

1.3.2.1 Updating jump rates on a graph

First, let's consider using a bipartite graph structure to reduce the number of recomputed rates at each jump. Consider the bipartite graph formed by associating the jump operators with one set of vertices, the Hilbert spaces that they act on with the other set of vertices, and where we add an edge between jump L_a and Hilbert space $\mathcal{H}_i$ whenever L_a acts on $\mathcal{H}_i$ (see Figure 1.3). Now let the *neighbors* N_a of the a-th jump be the set of jumps that act on any of the Hilbert spaces that L_a acts on:

$$N_a \equiv \{b : L_b \text{ and } L_a \text{ act on } \mathcal{H}_i\} \tag{1.14}$$

(note that a jump is its own neighbor. The set of neighbors is colored blue in Figure 1.3.)

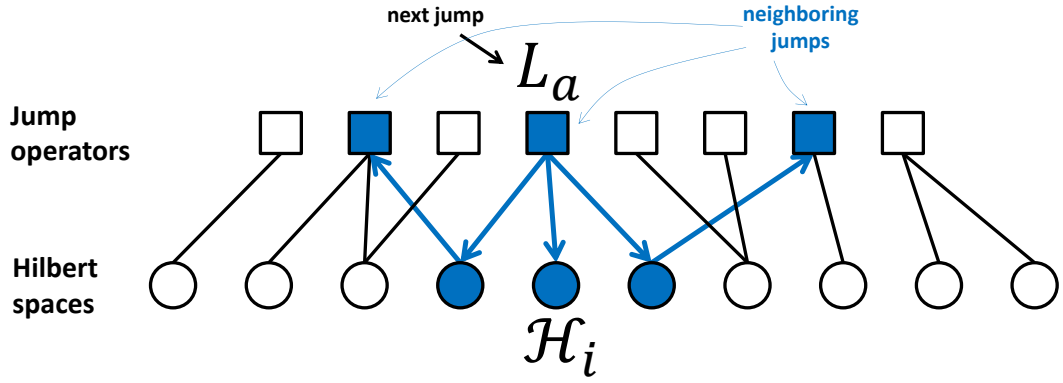

Figure 1.3: Bipartite graph connecting jump operators and the Hilbert spaces they act on. The neighboring jumps of jump L_a are shown in blue. The rates for these jumps are recomputed whenever jump operator L_a is applied.

This bipartite graph structure is reminiscent of message-passing algorithms on factor graphs; here the "messages" are instructions to neighboring jumps to recompute their rate.

Now when jump L_a is applied, the only jumps for which the rate needs to be recomputed are indexed by the neighbor set N_a, since the system wavefunction stays constant (up to normalization) on all Hilbert spaces other than the ones L_a acts on. This spares us from recomputing every jump rate after every jump, with the savings

growing as the bipartite graph becomes more sparse. In the case of the expander code circuit simulations of Chapter 2, there are 100,000 jump operators, but each has only about 6 neighboring jumps on average, so the savings are large.

1.3.2.2 Sampling jumps and jump times

Second, let's consider sampling the jump index and jump time. One way of doing this is to sample the time until the next jump for every jump and take the nearest jump time. This can be inefficient (linear in the total number of jumps) if we recompute all the jump times each time a jump is selected.

One way of doing this efficiently is to form a tree whose leaves are the jump rates $r_a(t)$ (shown in Figure 1.4). The internal (non-leaf) nodes store the sum of the rates of their child nodes. The root node thus stores the sum of all jump rates $r_{\text{Total}}(t)$ - this is the rate for any jump to occur at time t. This tree of sums enables us to both sample a jump index proportional to its jump rate and to update a particular jump rate in time logarithmic in the total number of jump operators.

To sample a jump index, sample $u \sim \text{Unif}(0,1)$, multiply u by the current total jump rate $r_{\text{Total}}(t)$ and wander down the tree from the root to the leaf that contains $u \cdot r_{\text{Total}}(t)$ - this leaf gives the index of the selected jump. We can sample the time until the next jump by inverse transform sampling to obtain a quantity with an exponential distribution with mean $1/r_{\text{Total}}(t)$:

$$t_{\text{jump}} = -\frac{\log u}{r_{\text{Total}}(t)} \tag{1.15}$$

Since the jump rates are occasionally recomputed in the course of a simulation, our tree of sums of rates must change too. This can also be done in time logarithmic in the total number of jumps: we start at the leaf corresponding to an updated jump rate, and update the rate sums of the parent nodes until we reach the root and update the total jump rate.

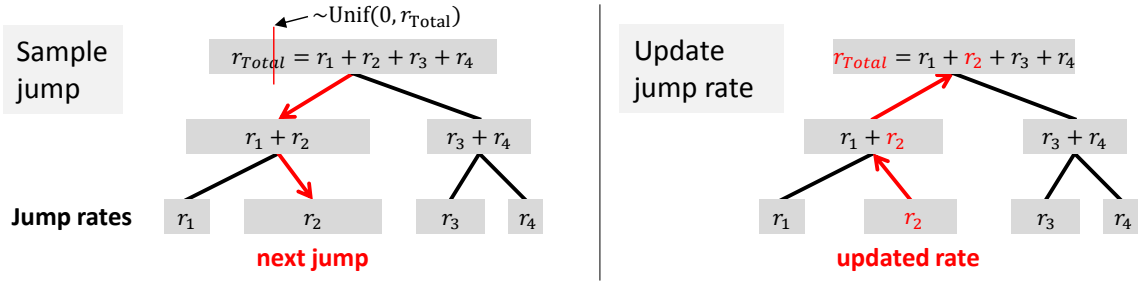

Figure 1.4: Jump rates stored in tree of sums. (left) sampling the next jump by sampling $u \in (0, r_{\text{Total}})$ uniformly, walking down tree. (right) updating a jump rate and parent nodes up to the root.

1.3.2.3 The full algorithm

Figure 1.5 shows a complete iteration of our algorithm: a jump operator and jump time is sampled, the jump operator is applied to the wavefunction, and the jump rates for the neighboring jump operators are recomputed using the new, post-jump, wavefunction (as is the total jump rate to maintain a valid sums of rates tree). We make the jump rates participate simultaneously in two data structures to reap the benefits of both: the bipartite jumps/Hilbert spaces graph lets us recompute only those rates that change after a jump, and the tree of rate sums lets us sample jumps and jump times efficiently.

1.3.2.4 An improvement

We can do less work by classifying the jump terms into those that change the state and those that only probe it[10]. For example, if a jump term involves an atomic lowering operator $L \sim \sigma_{+-} = |+\rangle\langle -|$, then acting with L on the wavefunction changes the state and can result in recomputation of jump rates that touch the Hilbert space for this atom. On the other hand, if $L \sim \Pi_+ = |+\rangle\langle +|$, so that Π_+ is a projection operator, then, given our assumption in Section 1.3 that the wavefunction remains an eigenstate of the effective Hamiltonian H_{eff} during the simulation, application of

[10] This distinction makes sense for cases when the wavefunction is an eigenstate of the system Hamiltonian at all times, so that the "probe" jump terms are proportional to projection operators.

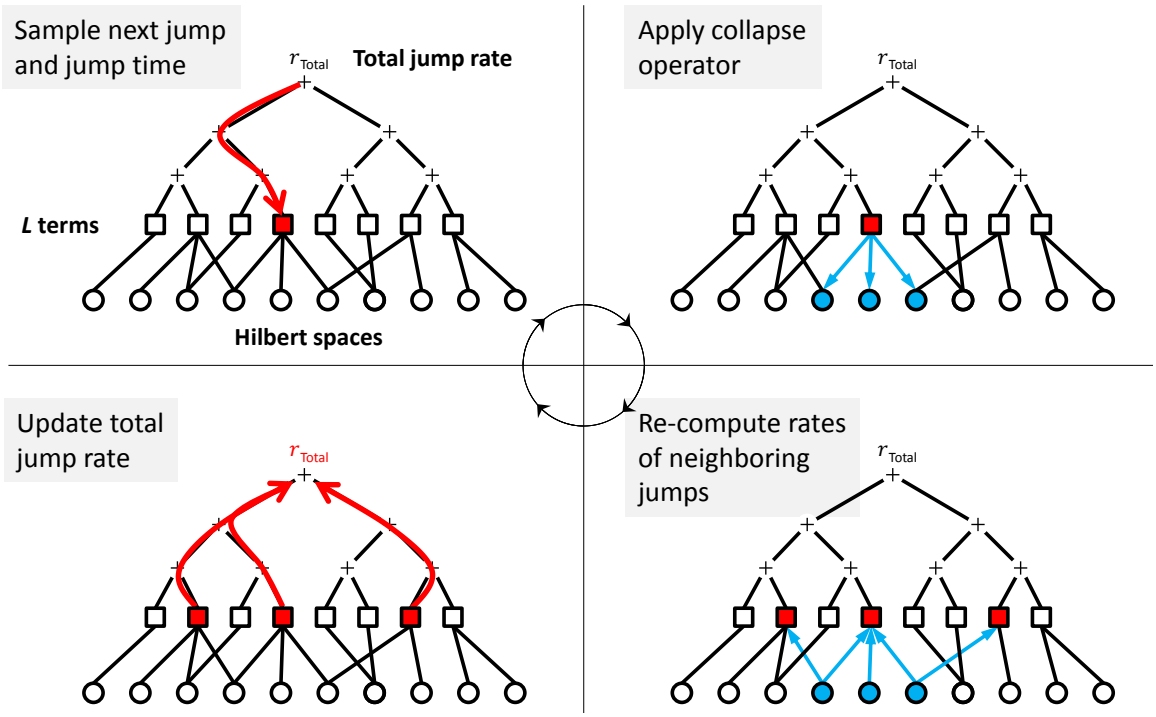

Figure 1.5: A complete iteration of our algorithm (clockwise from top-left).

L does not change the wavefunction (either this component of the wavefunction is in the $|+\rangle$ state, or not, in which case L is applied with rate 0). Therefore, we can exclude this jump term from the simulation entirely (by setting its rate to 0 at all times) and obtain the same dynamics as when we include it. If a jump term involves operators that both probe (project) and change the state, we need only use the probe part to compute the for rate applying this jump term, and only use the non-probe part when applying it.

1.3.3 Possible extensions

It would be nice to extend this to work with non-trivial Hamiltonian. This might be possible if the Hamiltonian evolution is slow compared to the total jump rate, so that the Hamiltonian contribution to the rates can be recomputed rarely. Perhaps we

could propagate approximate time derivatives of the jump rates due to the Hamiltonian evolution and modify our jump sampling procedure for time-varying rates. The possibly-sparse structure of terms in a non-trivial Hamiltonian is already exploited in the time-integration routines we use; perhaps we could exploit it again somehow for recomputing jump rates.

It would be nice, too, to relax our assumption that different components do not become entangled in the course of a simulation. Perhaps we can have a hybrid scheme where some blocks become entangled, requiring more computation, while on some larger scale in the circuit we ignore entanglement.

Chapter 2

A photonic LDPC decoder

This Chapter presents a photonic architecture for decoding of a class of low-density parity-check (LDPC) codes and follows closely our work (Pavlichin and Mabuchi, 2013). This proposal was inspired by the following image:

Photonic circuits involve stateful components coupled via guided electromagnetic fields and their time evolution can be viewed as a stochastic process[1] on a graph, the nodes being the components and the edges corresponding to light-mediated interactions between them. It feels like such circuits might be natural candidates for native implementation of iterative stochastic algorithms based on propagation of information around a graph. Such algorithms, including variants of message-passing schemes like belief propagation, have the flavor of nodes repeatedly exchanging information locally with their neighbors until global convergence. This picture invites an analogy to the dynamics of a network of photonic components, each of which has some internal degree of freedom (e.g., an "atomic" state), coupled via continuous interaction with propagating coherent fields. Thus photonic information processing systems could provide a native hardware platform for the implementation of iterative graph-based algorithms that are currently executed using electronic computers with incommensurate (though universal) circuit architectures that simulate message

[1] See Chapter 1 for a review of the dynamic models for optical circuits.

passing inefficiently. Conversely, this class of algorithms suggests novel circuit architectures for signal processing and computation that are well matched to nanophotonic device physics.

Here we develop an instance of this direct mapping of a graph-based algorithm to a photonic circuit design for a simple and practically useful task: iterative decoding of *expander codes*, a class of low-density parity-check (LDPC) error-correcting codes for communication over a noisy channel. We work in the setting of linear coding theory in which every codeword is required to satisfy a set of parity check constraints, i.e., sums modulo 2 of subsets of its bits. The assignments (0 or 1) of the codeword bits and the values of their parity check sums correspond to the states ($|0\rangle$ or $|1\rangle$) of a collection of two-state systems. Here we have in mind that $|0\rangle$ and $|1\rangle$ ideally should correspond to orthogonal quantum states of an atom-like elementary physical degree of freedom, to facilitate ultra-low energy scales for switching, but our circuit does not require coherent superpositions or entanglement. For decoding a possibly corrupted channel output, we consider a simple iterative decoding procedure for the expander LDPC codes (Sipser and Spielman, 1994, 1996): flip any bit (i.e., $0 \leftrightarrow 1$) that appears in more unsatisfied than satisfied parity check constraints; repeat until no more flips occur. We map this decoding procedure onto a closed-loop feedback circuit: a simple sub-circuit is engineered to encode parity check sum values in the state of an optical field, and another sub-circuit is designed to route feedback optical fields such that the states of certain components are flipped (i.e., $|0\rangle \leftrightarrow |1\rangle$) at a rate that grows with the number of unsatisfied parity check constraints.

The proposed circuit is autonomous, continuous-time and asynchronous. No external controller, measurement system or clock signal is required, so the circuit can be realized as a single photonic device whose only required inputs are stationary coherent optical fields that drive the computational dynamics (i.e., supply power) [2]. This follows the spirit of the systems we have designed in previous work on autonomous quantum memories (Kerckhoff et al., 2010, 2011). In contrast to our earlier work, the decoding circuit in the present proposal is straightforwardly extensible to the

[2] Note that signal processing devices that require only optical forms of power may be of practical interest for large-area fiber optic networks.

long block lengths (thousands of bits) used in practical LDPC implementations, as it involves a simpler feedback circuit architecture [3].

Our circuit requires a collection of two-state latch systems coupled to input and output field modes. Here we consider designs based on the attojoule nanophotonic relay proposed in (Mabuchi, 2009), which is based on ideas of cavity quantum electrodynamics (cavity QED), but any photonic system that functions as a latch potentially could be used in our circuit, e.g., (Mabuchi, 2011). Moreover, our scheme tolerates noisy components (e.g., spontaneous switching of a latch between the 0 and 1 states), can compensate for this noise with increased input optical power, and actually performs optimally (in terms of bits decoded per second) when the components "misbehave" at some nonzero rate. The graceful change in performance with increasing component imperfection and with varying optical input power is important for the practical usefulness of such a circuit. In our circuit design there is no real distinction between power and signal, as the power carried by the optical signal fields drives all the computational dynamics of the components, and it is shown (see Figure 2.9) that simply increasing the optical input power reduces the error correction latency with fixed hardware. Our circuit tolerates a wide range of input powers with a constant performance as measured by bits corrected per joule.

This Chapter is organized as follows (this mirrors the organization of (Pavlichin and Mabuchi, 2013)): We first briefly review linear error-correcting codes and an iterative decoding scheme for expander LDPC codes. We then describe in an intuitive way the operating principles of our photonic circuit implementation of an iterative LDPC decoder. The subsequent Section gives a more detailed picture of our circuit in terms of open quantum systems theory. We then present some numerical tests of our system and conclude with a discussion.

[3] The circuits in our earlier work implement maximum likelihood (ML) decoders for small codes like the three-bit bit-flip code. ML decoding is, however, impractical for larger block lengths, as it requires either a circuit size or decoding time exponentially large in the block length. Iterative decoding algorithms such as the one discussed in this work have a larger error rate than ML decoders, but require only polynomial or even linear resources in the block length

2.1 Linear Codes and Iterative Decoding

We briefly review and set up notation for block binary linear error-correcting codes and an iterative decoding procedure for expander LDPC codes.

2.1.1 Linear Codes

We work with binary bits transmitted in blocks of length n through the binary symmetric channel (BSC) that with some fixed probability independently flips (i.e. $0 \to 1$, $1 \to 0$) the transmitted bits. To protect from errors, the sender restricts the possible channel inputs to the set of codewords—a subset of all 2^n possible inputs. The decoder attempts to find the nearest codeword to the possibly corrupted output of the channel. Equivalently, the bits are stored in memory that accumulates errors with time; the sender/decoder attempt to minimize losses through redundancy in the encoded memory bits.

Linear codes require each codeword $x^n = (x_1, \ldots, x_n)$ to satisfy m parity check constraints. A parity check constraint $\mathbf{c}$ is a subset of the n message bits whose sum is constrained to equal 0 modulo 2:

$$\sum_{j \in \mathbf{c}} x_j = 0 \pmod 2 \tag{2.1}$$

A vector x^n is a codeword if and only if it satisfies every constraint. The rate R of the code is the ratio of the number of non-redundant bits to the total number of bits per transmission, $R = (n - m)/n$.

It is useful to think of a code as an undirected bipartite graph, the Tanner graph (Tanner, 1981), whose n "variable" nodes correspond to the message bits and whose m "check" nodes correspond to the constraints. Edges connect variable nodes and the constraints that include them.

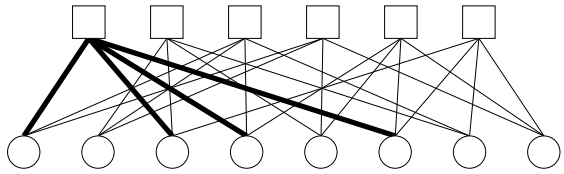

Figure 2.1: Tanner graph for a $(n = 8, l = 3, k = 4)$ LDPC code. Circles (squares) indicate variable (check) nodes. The thick edges indicate that parity check constraint 1 is $x_1 + x_3 + x_4 + x_6 = 0 \pmod 2$ (numbering the check and variable nodes from left to right).

2.1.2 Linear $\supset$ LDPC $\supset$ expander codes

Low-density parity-check (LDPC) codes are linear codes introduced by Gallager in 1962 (Gallager, 1962, 1963) and are among the first known near capacity-achieving efficiently decodable codes. The parity checks of a (n, l, k) LDPC code all include k bits, and each bit is included in l parity checks (in the Tanner graph, each variable node has degree l and each check node has degree k). The codes are "low-density" because the total number of variable-check pairs is ln, linear in the block length n (rather than quadratic in n for a dense graph); the Tanner graph is sparse. The rate of the code is $R = (n - m)/n = (k - l)/k$.

Figure 2.1 shows the Tanner graph for a particular $(n = 8, l = 3, k = 4)$ LDPC code, where variable (check) nodes are drawn as circles (squares), and we have highlighted a particular parity check constraint. This graph would look sparse for larger n.

LDPC codes shine because they can be decoded efficiently by iterative algorithms that have good performance in practice and in theory. These schemes include those in Gallager's original work (Gallager, 1963), as well as message-passing algorithms and belief propagation; for a theoretical analysis of their performance see (Luby et al., 2001; MacKay and Neal, 1997; MacKay, 1999; Richardson and Urbanke, 2001). These schemes all have the flavor of variable and check nodes repeatedly exchanging information about the most likely codeword given the observed channel output and differ from each other in how that information is represented (e.g. binary or real-valued messages) and how new messages are computed from old.

Expander codes are a class of LDPC codes, introduced by Sipser and Spielman (Sipser and Spielman, 1994, 1996), for which a particularly simple iterative decoding procedure exists and which are easy to make by using a random construction. Expander codes require the Tanner graph to be a good expander graph, meaning that the number of check nodes neighboring any small enough subset V of the variable nodes grows fast enough linearly with $|V|$. For our purposes it suffices to note that a randomly sampled bipartite graph with fixed variable and check node degree (a regular LDPC code) probably makes a good expander code (Sipser and Spielman, 1996).

2.1.3 Iterative decoding of expander codes

The iterative decoding procedure that is our focus in this work is the sequential decoder of Sipser and Spielman (Sipser and Spielman, 1996). The variable bits are initially assigned to 0 or 1, equal to the observed output of the channel (we work with a binary symmetric channel that flips incoming bits with probability less than 1/2). The initial assignment of the variables may fail to satisfy all parity check constraints due to errors. The decoding procedure is as follows:

- Flip (i.e. $0 \leftrightarrow 1$) any variable that is included in more unsatisfied than satisfied constraints.
- Repeat until no more variables are flipped.

Each iteration reduces the total number of unsatisfied constraints, so the procedure terminates when either there are 0 unsatisfied constraints (successfully outputting a codeword) or it gets stuck and declares failure to decode. While this procedure could be applied to any binary linear code, (Sipser and Spielman, 1996) prove that for expander codes this procedure removes a constant fraction of errors and, if the initial fraction of errors is low enough, is guaranteed to succeed. For the expander LDPC codes, each variable participates in k constraints, so we flip the variable's assignment if the number of unsatisfied constraints is greater than $k/2$.

Importantly for our work, in (Sipser and Spielman, 1996)'s numerical experiments, it was found that permitting the algorithm to make some amount of backwards progress (sometimes increasing the total number of unsatisfied constraints) increased the probability of success. This suggests the procedure is robust to noise affecting the computation. In our approximate implementation of this iterative algorithm, described below, backwards progress is unavoidable and the hardware itself is noisy, so this robustness of the decoding procedure to noise is desirable.

This procedure is not technically a message-passing algorithm in the sense of (Burshtein and Miller, 2001), in that information flow from a check to a variable node (a possible "flip" instruction) does not exclude information received by the check node from that variable node (the bit state). Nonetheless it is convenient to discuss the error-correcting dynamics, as (Sipser and Spielman, 1996) do, in terms of variable nodes receiving "flip messages" from check nodes.

2.2 A photonic decoding circuit: Overview

We give an intuitive description of the operation of our expander code decoder circuit before giving a more precise description in terms of open quantum systems in the Section that follows.

2.2.1 The idea

Our circuit consists of a collection of two-state ($|0\rangle$ or $|1\rangle$) systems, one for each of n variable and m check nodes in the Tanner graph for an error-correcting code. Information exchange between the variable and check systems is mediated by coherent fields interacting with these systems (e.g. a beam scattering from one atom-cavity system into another). There are two crucial interactions:

- Fields outgoing from a system can encode that system's state (perform a measurement)
- Fields incoming to a system can drive that system into a desired state (apply a control)

These two interactions allow us to construct a closed-loop, autonomous measurement and feedback circuit that achieves:

- **Parity checks/Measurements**: A field scattered (e.g. a beam reflected) from the set of all variable bit systems included in some parity check constraint encodes their sum modulo 2. This field then drives the check system into the $|\text{satisfied}\rangle$ or $|\text{unsatisfied}\rangle$ states ($|0\rangle$ or $|1\rangle$, respectively).

- **Error correction/Feedback**: A field scattered from the set of all check systems that include a particular variable has an amplitude that increases with the number of unsatisfied checks involving that variable. This field then drives the variable system to flip between the $|0\rangle$ and $|1\rangle$ state at a rate proportional to the magnitude of the field amplitude. The more unsatisfied parity checks, the faster the flipping occurs.

The time evolution of this circuit is modeled as a continuous time Markov jump process [4]. The jumps are changes in state ($|0\rangle \leftrightarrow |1\rangle$) and the jump rates depend on amplitudes of fields interacting with the two-state systems. The circuit is autonomous and asynchronous in that there is no external clock signal or external controller to process the parity measurement outcomes and to create an appropriate feedback field.

We note that the iterative decoding algorithm of (Sipser and Spielman, 1996) that our circuit emulates, summarized in Section 2.1.3, can be cast in terms of a continuous time Markov jump process as well: if a variable is included in more unsatisfied than satisfied constraints, set the rate for "flipping" it to $R_{\text{flip}} > 0$, otherwise set $R_{\text{flip}} = 0$. In our implementation, the value of R_{flip} scales with the number of unsatisfied constraints in a different way (and is never 0; see Section 2.3.2.2), but we attain comparable empirical performance in simulation.

Finally, we note that our circuit is essentially classical in its operation, even though we utilize quantum stochastic differential equations (QSDEs) to describe the dynamics of the components and their interactions in order to obtain a circuit model that is valid in the ultra-low power regime of significant quantum fluctuations (photon shot noise).

[4] This description follows from the open quantum systems dynamics treatment of our circuit. See Section 1.3 for details

Entanglement between different subsystems is insignificant and is not exploited, and thus does not need to be protected from interactions with the outside environment.

2.3 A photonic decoding circuit - construction

In this Chapter we describe the photonic component subsystems that make up our circuit, specify their interconnections, and give an intuitive description of our circuit's dynamics. We work in the framework developed by Gough and James (Gough, 2008, 2009) for modeling open quantum systems interacting via coherent fields (Hudson and Parthasarathy, 1984; Carmichael, 1993; Gardiner, 1993; Barchielli, 2006). Chapter 1 briefly reviews this framework in enough detail for the reader to reproduce our simulations.

2.3.1 Photonic circuit components

The basic component of our circuit - used to represent both variable and check node assignments ($|0\rangle$ and $|1\rangle$) - is a photonic latch, shown in Figure 2.2, that behaves like the set-reset latch in electronics. There are several proposals for implementing latching behavior in nanophotonic circuits (Mabuchi, 2009, 2011; Majumdar et al., 2012; Faraon et al., 2008; Nielsen and Kerckhoff, 2011). One such system, a coupled atom-cavity system (Mabuchi, 2009), is shown in Figure 2.2 (panel (b)). Our circuit construction is defined without reference to a particular physical system and assumes that the latch system that is used implements the following protocol.

The latch has a discrete internal degree of freedom (e.g. an atomic state) coupled to two external field modes, labeled "set" and "reset." A signal incoming to the "set" ("reset") input drives the latch into the $|1\rangle$ ($|0\rangle$) state. When neither the set nor reset input is powered, the latch maintains its current state. Usefully for us, driving both the set and reset inputs simultaneously - an undefined condition for the electronic set-reset latch - results in astable behavior, with the latch state repeatedly jumping between the $|0\rangle$ and the $|1\rangle$ state with exponentially-distributed jump times.

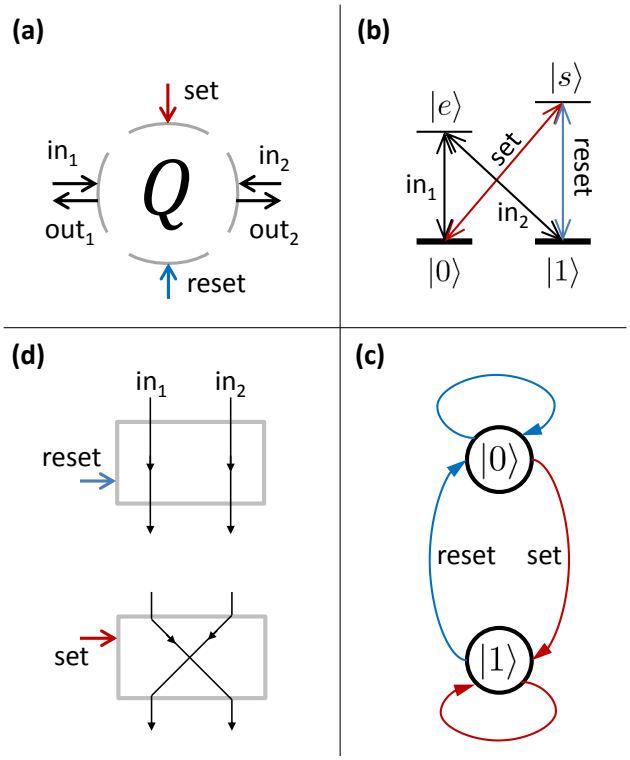

Figure 2.2: Latch component from (Mabuchi, 2009). **(a)** Input-output connections (reflected set and reset outputs not shown). **(b)** Input field couplings to internal states. **(c)** The latch approximated as a two-state continuous time Markov jump process after adiabatically eliminating the excited states $|e\rangle$ and $|s\rangle$ (see (Mabuchi, 2009) for this derivation). **(d)** The latch routes the input fields into output fields, switching them if its internal state is driven to $|1\rangle$.

The latch routes two input channels (in_1 and in_2) into two output channels (out_1 and out_2). When the latch is in the $|0\rangle$ state, the outputs match the inputs ($\text{out}_{1,2} = \text{in}_{1,2}$); when the latch is in the $|1\rangle$ state, the outputs are switched ($\text{out}_{1,2} = \text{in}_{2,1}$).

In addition to the latch, our circuit uses beamsplitters with some fixed transmission and reflection coefficient. Proposals for integrated nanophotonic beamsplitting devices include (Bayindir et al., 2000; Liu et al., 2005). The Gough-James $(\mathbf{S}, \mathbf{L}, H)$ description of these components connected to each other and driven by coherent fields is provided in Appendix 2.A.

2.3.2 Circuit construction

We describe how the latches, beamsplitters, and coherent inputs are used to form our expander code decoding circuit. There are two kinds of interactions to implement between the variable and check systems: parity check sums and feedback to "flip" the variable nodes.

2.3.2.1 Parity checks

Figure 2.3 shows our parity check sum construction. For each parity check $\mathbf{c}$ corresponding to the k-variable constraint $\bigoplus_{i=1}^{k} x_{\mathbf{c}(i)} = 0$, there are k variable latch systems, $Q^{\text{var}}_{\mathbf{c}(1)}, \ldots, Q^{\text{var}}_{\mathbf{c}(k)}$, and one check latch system $Q^{\text{check}}_{\mathbf{c}}$ (here $\oplus$ denotes addition modulo 2). The current assignment (0 or 1) of the variables included in $\mathbf{c}$ is represented by the states ($|0\rangle$ or $|1\rangle$) of the variable latches; the check latch's state is meant to represent the sum of these assignments modulo 2. As shown in Figure 2.3(b), the variable latches share two common optical paths for their in_1 and in_2 inputs and outputs. An input field with amplitude α is incident to input port in_1 of check latch $Q^{\text{var}}_{\mathbf{c}(1)}$. Subsequently, the two output ports of $Q^{\text{var}}_{\mathbf{c}(i)}$ connect to the two input ports of $Q^{\text{var}}_{\mathbf{c}(i+1)}$ for $i < k$. The outputs of the final variable latch $Q^{\text{var}}_{\mathbf{c}(k)}$ connect to the set and reset ports of check latch $Q^{\text{check}}_{\mathbf{c}}$.

Each time a $|1\rangle$ state is encountered at a variable latch along the beam path, the latch switches the beam path between the upper and lower branches. If the output power of the final latch is in the upper (lower) branch, then the parity of the variable

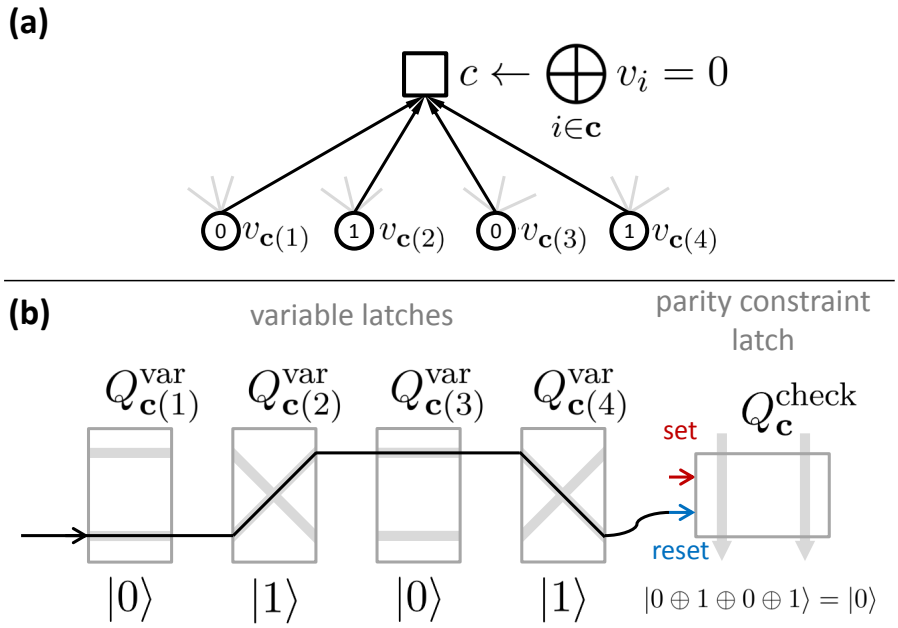

Figure 2.3: Photonic circuit implementing a parity check computation. **(a)** The parity check sum operation for a fragment of the Tanner graph for a linear code (rest of graph in gray). The check node $\mathbf{c}$ is assigned the sum modulo 2 of the variable nodes. **(b)** The photonic circuit implementation of the check sum using variable latches $Q^{\text{var}}_{\mathbf{c}(1)}, \ldots, Q^{\text{var}}_{\mathbf{c}(k)}$ and check latch $Q^{\text{check}}_{\mathbf{c}}$. Power is routed into either the SET or RESET ports of the check latch conditional on the parity of the variable latches' states.

assignment is odd (even), and the SET (RESET) port of the check latch receives power, driving the check latch into the $|\text{unsatisfied}\rangle = |1\rangle$ ($|\text{satisfied}\rangle = |0\rangle$) state. The rate at which the check latch is driven to the appropriate state is proportional to the input field power $|\alpha|^2$ in units of photons per second.

The check latch $Q^{\text{check}}_{\mathbf{c}}$ in turn routes fields that participate in the feedback circuit described in the next Section.

2.3.2.2 Feedback to variables

Figure 2.4 shows our feedback to variables construction. For a variable v, let $\mathbf{v}$ denote the l parity check constraints that include v: $\mathbf{v} = \{\mathbf{c} : v \in \mathbf{c}\}$. The current

value (parity - 0 or 1) of the each check in $\mathbf{v}$ is represented by the state ($|0\rangle$ or $|1\rangle$) of latches $Q^{\text{check}}_{\mathbf{v}(1)}, \ldots, Q^{\text{check}}_{\mathbf{v}(l)}$. As shown in Figure 2.4(c), the check latches share a common optical path. An input field with amplitude β is incident to input port in_1 of latch $Q^{\text{check}}_{\mathbf{v}(1)}$. Subsequently, for each check latch $Q^{\text{check}}_{\mathbf{v}(i)}$, $1 \leq i \leq l$, the second output is fed back into the second input of the same latch after passing through an attenuator (e.g. a beamsplitter) that dumps (e.g. reflects out of the beam path) a fraction $\gamma < 1$ of incident power and transmits a fraction $1 - \gamma$ of the power back into the beam path.

Each time an unsatisfied parity check constraint state ($|1\rangle$ state) is encountered at a check latch along the beam path, the power reaching the next check latch in the path is attenuated by a factor of γ. The output of the final check latch in the path $Q^{\text{check}}_{\mathbf{v}(l)}$ is routed to drive both the SET and RESET inputs of the variable latch Q^{var}_{v}, causing it to "flip" between the $|0\rangle$ and $|1\rangle$ states.

Once a flip of variable v occurs, the parity check system discussed in the previous Section updates the states of the check systems that include this variable, resulting in an updated value of the flipping rate for variable v. If the power in the measurement circuit used to perform the parity check computation is low enough, the feedback circuit may induce multiple flips of the same variable before the measurement system reacts. We consider this situation in the numerical results Section below.

The rate at which the variable latch Q^{var}_{v} flips is proportional to the attenuated power outgoing from the final latch in the beam path:

$$R_{\text{flip}} \sim \gamma^{(l-\#\text{unsat. checks})} \, |\beta|^2 = \gamma^{\#\text{sat. checks}} \, |\beta|^2 \tag{2.2}$$

If all l parity constraints that include a variable v are unsatisfied, the state of variable latch Q^{var} flips with the maximum rate proportional to $|\beta|^2$. If all l constraints are satisfied, the variable is flipped with non-zero rate proportional to $\gamma^l |\beta|^2$. Thus our circuit can induce errors. For $\gamma \ll 1$, a single induced error should be quickly corrected since the rate for correcting it is a factor of $1/\gamma^l \gg 1$ larger than the rate for inducing it.

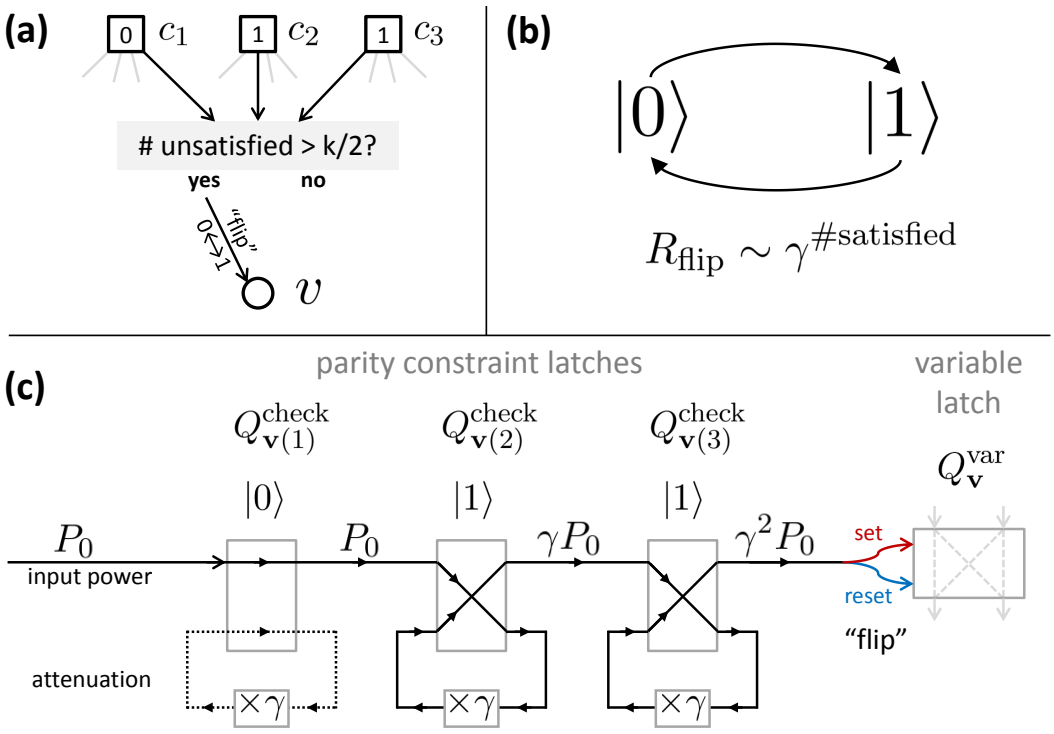

Figure 2.4: Photonic circuit implementing feedback to variables. **(a)** The "flip" operation for a fragment of the Tanner graph of a linear error-correcting code (rest of graph in gray). If a majority of the parity check constraints that include variable v are unsatisfied, the assignment of variable v is flipped (Sipser and Spielman, 1996). **(b)** For our photonic circuit implementation, the rate of "flipping" the variable system state scales exponentially with the number of satisfied parity check constraints. **(c)** The photonic circuit implementation of the error-correcting feedback using check latch systems $Q^{\text{check}}_{\mathbf{v}(1)}, \ldots, Q^{\text{check}}_{\mathbf{v}(l)}$ and variable latch system $Q^{\text{var}}_{\mathbf{v}}$. Power driving the variable system $Q^{\text{var}}_{\mathbf{v}}$ to flip is attenuated by a factor of γ for every satisfied parity check constraint.

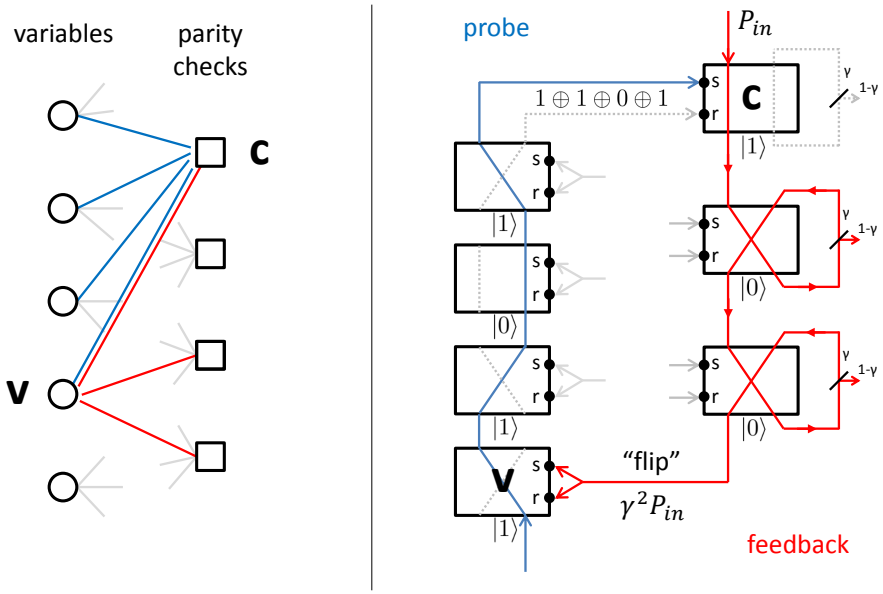

Figure 2.5: Measurement and feedback circuit fragment (right) corresponding to fragment of Tanner graph (left).

Our circuit corrects errors that are involved in i parity check violations on a timescale proportional to $1/\gamma^i$. The smaller we make the attenuation factor γ, the fewer induced errors there are, but the longer the decoding takes to complete. We derive some bounds on the maximum value of γ in terms of the code parameters such that our procedure is likely to succeed in Appendix 2.B. We guess that the attenuation factor γ should not be too small, since the decoding probability may increase when some induced errors are permitted, as observed in (Sipser and Spielman, 1996). This intuition is consistent with our observations in the numerical results Section (2.4).

2.3.3 Complete circuit summary plots

Figure 2.5 shows both the measurement and feedback subcircuits for a fragment of our decoder circuit corresponding to a fragment of the Tanner graph of an error-correcting code. There is one such fragment for each of nl edges in the Tanner graph of the code.

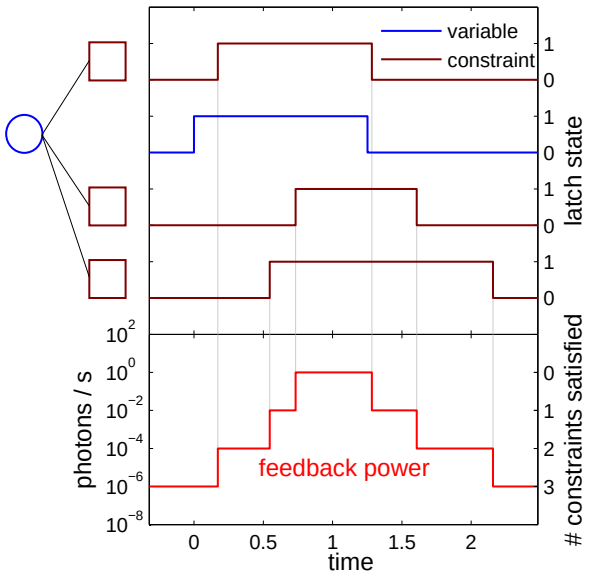

Figure 2.6: Part of a trajectory of the decoding circuit for a fragment of an error-correcting code. (top panel, blue line) state of a latch corresponding to a variable bit. (top panel, dark red lines) states of latches corresponding to parity check constraints that include the blue variable bit. (bottom panel) the feedback power applied to the variable bit, inducing it to "flip" state. On a log scale, this feedback power is proportional to the number of satisfied parity check constraints that include this variable bit. See text for trajectory narration.

Figure 2.6 shows a portion of a simulated trajectory for a fragment of the code. The top panel shows the state ($|0\rangle$ or $|1\rangle$) of a latch corresponding to a variable bit (blue) and the three latches corresponding to the three parity checks that include this bit (dark red). At time 0, an error causes the variable bit latch (blue) to flip state (perhaps the component malfunctioned or the feedback system induced the error). The three check latches corresponding to this bit then turn on (enter the unsatisfied, $|1\rangle$ state) after some exponentially-distributed waiting time (the mean of the waiting time is set by the input probe power used to perform the parity check sum computation). For each check latch that enters the unsatisfied $|1\rangle$ state, the feedback power reaching the variable bit grows by a factor of $1/\gamma$, where γ is the attenuation constant. Around time 1.25, the feedback induces the bit to flip back to the $|1\rangle$ state. After an additional random waiting time, the three latch systems return to the satisfied $|0\rangle$ state. Note that the feedback power reaching the bit is never 0, but reaches a minimum when all parity check constraints are satisfied.

2.3.4 Fan-out

Our decoder circuit requires each variable latch component to participate in multiple (l) parity check constraints, and requires each parity constraint latch component to feed back to multiple (k) variables. Since the latch described in Section 2.3.1 (and in greater detail in Appendix 2.A.3) can switch only a single pair of signal inputs, it is not on its own sufficient for our needs. We can augment our latch to achieve the desired fan-out (and avoid the difficulty of having multiple beam paths access a single structure in a planar circuit) by breaking up each latch into a set of subsystems, each responsible for routing a single in/out signal pair. The subsystems are yet more latches, but each routes only a single pair of in/out signals, corresponding to the latch description of Section 2.3.1.

Figure 2.11 (in Appendix 2.C) shows the circuit for the augmented latch that routes multiple in/out signal pairs. This augmented latch is used implicitly in our circuit description above and is described in detail in Appendix 2.C. The idea is that a single "master" latch receives the two set/reset inputs and then routes power to

drive the "slave" latches into a matching state. Distributing the master latch state to the slave latches requires another optical input with amplitude $\alpha_{\text{f-o}}$ (for fan-out).

The numerical results presented below in Section 2.4 are done at infinite fan-out power (power in the yellow optical line in Figure 2.11) so that the effects of the extra fan-out components on decoding performance can be ignored. We consider the effect of finite fan-out power in Section 2.4.5.

2.4 Numerical Experiments

Table 2.1 lists the parameters used in our simulations. The spontaneous flip rate η models component noise during the computation—all latches in our circuit independently flip ($|0\rangle \leftrightarrow |1\rangle$) with rate η.

parameter	symbol	notes
block length checks per variable variables per check	n l k	$m = nl/k$ parity checks
probe amplitude feedback amplitude fan-out amplitude	α_{pr} α_{fb} $\alpha_{\text{f-o}}$	power $\sim |\alpha|^2$
feedback power attenuation	$0 < \gamma < 1$	rate to flip variable $= |\alpha_{\text{fb}}|^2 \gamma^{\#\text{satisfied checks}}$
spontaneous flip rate	η	all latches independently flip state with rate η

Table 2.1: Simulation parameters

2.4.1 Simulating quantum trajectories

Our circuit evolves according to the master equation (1.1). Rather than solve this equation for the density matrix ρ for our system, we sample multiple trajectories of the system wavefunction $|\psi\rangle$ and average observed quantities over these trajectories, as described in Section 1.2.1. Simulation of quantum trajectories given a master equation in the form of (1.1) is computationally easier than integrating the master

equation and is discussed in detail in (Wiseman, 1996, 2010). One way to perform such simulations is to sample exponentially-distributed jump times for each component of the system $\mathbf{L}$ vector (rate for i-th component is $\sim |\langle\psi|L_i^\dagger L_i|\psi\rangle|^2$), apply the nearest-in-time jump to the system wavefunction, and resample all of the jump times given the new wavefunction. In general, there is a smooth Hamiltonian evolution occurring between jumps as well, but our decoder circuit's Hamiltonian is diagonal in the $\{|0\rangle, |1\rangle\}$ state basis, and this basis is fixed by the components of $\mathbf{L}$ (the jump terms) so we can ignore the smooth evolution and treat the system as a continuous time Markov jump process.

We prefer the trajectory approach in part because we want to average over different random instances of the expander code (with different network connectivities each time) and because it is useful to examine the time evolution of individual trajectories for an intuitive view of the circuit.

2.4.2 Trajectories

We uniformly randomly sample 30 bits to corrupt from the initial all-0 codeword of length $n = 1000$ for a randomly sampled LDPC code with $l = 5$, $k = 10$, and track the remaining number of errors in time. The code is generated by randomly sampling a bipartite graph with 1000 variable nodes each with degree 5, 500 check nodes each with degree 10. We take the feedback attenuation parameter $\gamma = 0.01$, set the feedback power to 1 (arbitrary units), the probe power to something much larger (10^5), set the rate for spontaneous component flips $\eta = 0$, and the fan-out power $|\alpha_{\text{f-o}}|^2$ to ∞ (thus ignoring the effects of fan-out; see Section 2.3.4 for explanation). Figure 2.7 shows the number of errors remaining as a function of time averaged over 999 trajectories, and for three individual trajectories. 999 of 1000 trajectories decoded successfully (converged the all-0 codeword). The one that did not is not included in the average.

We point out two features of the trajectory simulations. One is that (e.g. the red trajectory in Figure 2.7) the number of errors remaining sometimes increases in the course of a simulation. As discussed in our circuit description in Section

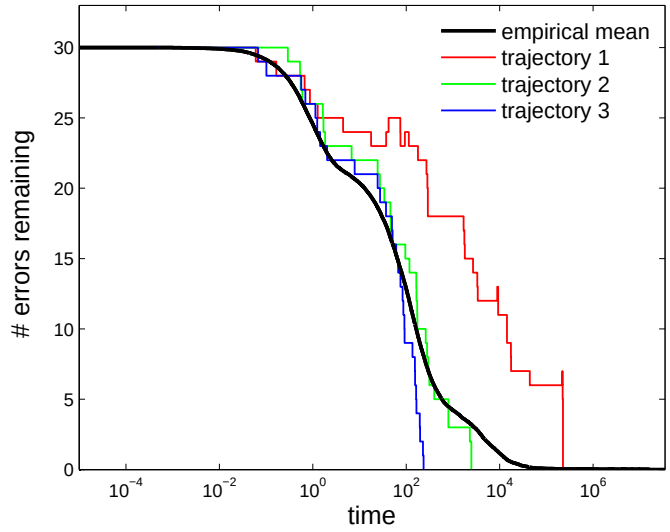

Figure 2.7: Trajectory simulation of the iterative photonic decoder for a $(n = 1000, l = 5, k = 10)$ expander code, 30 initial errors, circuit parameters $\gamma = 0.01$, $|\alpha_{\text{fb}}|^2 = 1$, $|\alpha_{\text{probe}}|^2 = 10^5$, $\eta = 0$, $|\alpha_{\text{f-o}}|^2 = 0$. (black) the mean number of errors remaining vs. time averaged over 999 trajectories. (red, green, blue) the number of errors remaining vs. time for three individual trajectories.

2.3.2.2, the circuit induces errors at some non-zero rate and then corrects the induced errors. Errors are most likely to be induced for variables that are involved in some, but not a majority of parity check violations. When the attenuation constant γ is too high (too little attenuation), the circuit may induce errors faster than they are corrected, resulting in a failure to decode. On the other hand, as γ is decreased, the circuit corrects errors at a lower rate, suggesting an optimal value of γ in terms of a performance vs. decoding time tradeoff. This tradeoff is considered in the next Section.

Second, the empirical mean of 999 trajectories (black trace in Figure 2.7 exhibits three shoulders (alternates between being locally convex and concave) in its decay toward 0. The shoulders are spaced approximately $1/\gamma = 100$ logarithmic time units apart, corresponding to the correction of errors that are involved in 5, 4, and 3 parity check violations, respectively. The mean number of errors remaining first declines significantly at time $t \sim 10^0$, consistent with feedback at maximal rate (no attenuation) $|\alpha_{\text{fb}}|^2 = 1$ flipping variables all $l = 5$ of whose corresponding parity check constraints are initially unsatisfied.

2.4.3 Performance vs. initial number of errors

We simulate our decoding circuit using the same code parameters as (Sipser and Spielman, 1996): a $(n = 40000, l = 5, k = 10)$ expander code, generated by randomly sampling a bipartite graph with 40000 variable nodes, 20000 check nodes, and degree 5 and 10 at the variable and check nodes, respectively. The performance of our decoder in simulation for these parameters is shown in Figure 2.8. This performance (top panel) is somewhat better than that of (Sipser and Spielman, 1996)'s scheme and somewhat worse than their version of the scheme permitting some backwards progress - occasionally allowing the total number of parity constraint violations to increase.

We see in Figure 2.8 (top) that the decoder's performance in terms of block error rate appears to saturate as the attenuation parameter γ decreases. At the same time,

the median time [5] to successfully decode grows as γ decreases (bottom), since the rate to flip bits scales exponentially in γ (eq. (2.2)). Thus we could set γ to the highest achievable value for a given channel error probability, desired mean decoding time, and probability to decode successfully.

2.4.4 Performance vs. input power with noisy circuit components

We consider the decoder's performance as a function of applied input power in terms of probability to decode, decoding rate (bits/s), and decoding energy (bits/J). Additionally, we set some non-zero rate η at which the circuit components undergo spontaneous flips ($|0\rangle \leftrightarrow |1\rangle$). This noise affects both the variable and the check latches and in turn both the measurement and feedback parts of the circuit. Figure 2.9 shows our numerical results for fixed component noise rate η, LDPC code parameters, initial number of errors, and attenuation parameter γ (see caption for parameter values).

We see (top panel of Figure 2.9) that to decode successfully most of the time, the feedback power needs to be large enough to overcome the errors induced by noise in the circuit components, but not much larger than the probe power. When the feedback power is much larger than the probe power, the probe circuit is too slow to turn off the feedback once an error is corrected and too slow to turn on the feedback for new errors (induced by either the feedback or spontaneous flips), so the feedback system may induce more errors than it corrects.

For the bottom panel of Figure 2.9 we fixed the probe to feedback power ratio at 1 and plotted the mean decoding rate and energy versus input power in bit/s, bit/J, respectively [6]. We defined the decoding rate as the reciprocal of the mean decoding

[5] We use the median, rather than the mean, time because as the probability to successfully decode drops sharply around 1800 initial errors, the distribution of decoding times spreads out over orders of magnitude (see Figure 2.8, bottom), with the mean dominated by the few longest trajectories. We imagine in actual use, we would operate some distance (in number of initial errors) below the point at which the decoder breaks down.

[6] These are in arbitrary time and energy units proportional to s and J, since we did not give physical values for our simulation parameters. To compute a fiducial time and energy to decode, we reference the latch switching time estimated in (Mabuchi, 2009) using the parameters of (Barclay

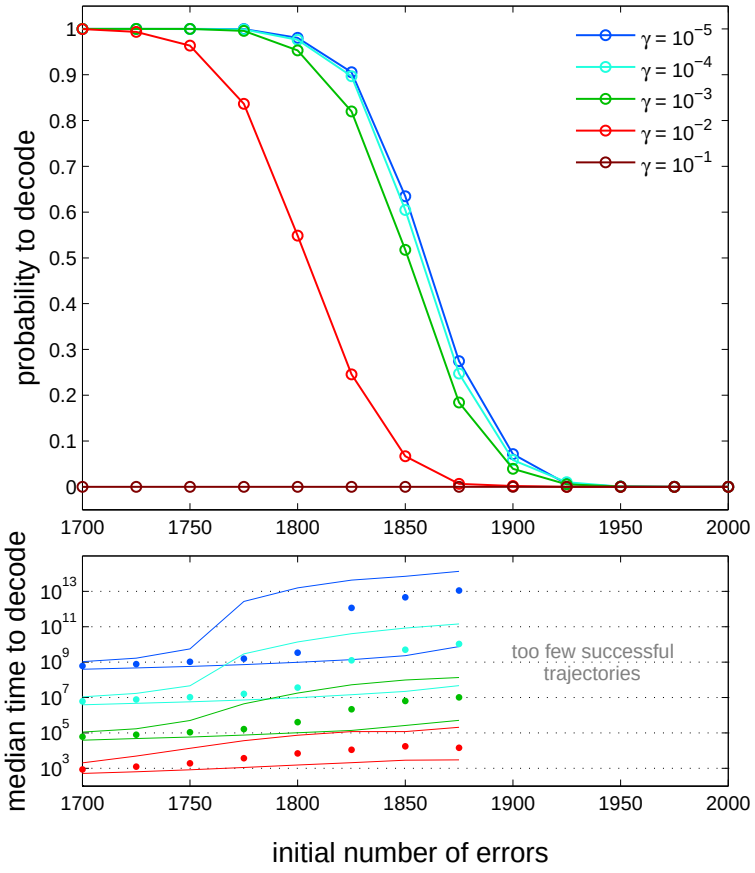

Figure 2.8: Decoder performance with varying feedback power attenuation parameter γ. (top) Probability to successfully decode vs. initial number of errors for several values of attenuation parameter γ (see legend box). (bottom, points) median time to decode conditioned on successfully decoding all errors. (bottom, solid lines) 90% interval for time to decode successfully. We did not track these quantities past 1875 initial errors due to low succesful decoding probability. The code parameters are the same as in (Sipser and Spielman, 1996): $(n = 40000, l = 5, k = 10)$. We set $\alpha_{\text{probe}} = 10^3$, $\alpha_{\text{fb}} = 10$, $\eta = 10^{-80}$, $\alpha_{\text{f-o}} = \infty$. We sampled 3000 trajectories for each data point.

time, conditioned on successfully decoding, and the decoding power as the decoding rate divided by the input power.

We see that for large enough input power, the decoding rate is proportional to the input power, while the energy cost per decoded bit is constant.

2.4.5 Performance vs. fan-out power

The previous numerical results were obtained for infinite fan-out amplitude $\alpha_{\text{f-o}}$, allowing us to ignore the impact on decoding performance of the extra fan-out latches described in Section 2.3.4. Here we consider varying the fan-out amplitude $\alpha_{\text{f-o}}$. Figure 2.10 shows our numerical results for fixed LDPC code parameters (same as for Figure 2.8 and Figure 2.9), fixed attenuation parameter γ, varying initial number of errors, and equal probe and feedback amplitudes $\alpha_{\text{pr}} = \alpha_{\text{fb}}$ (see caption for parameter values).

We see that for high fan-out power ($|\alpha_{\text{f-o}}/\alpha_{\text{fb}}|^2 \geq 10$) the decoder's performance in terms of block error rate vs. initial number of errors appears to saturate to the infinite fan-out power curve in Figure 2.8 [7]. For the case $|\alpha_{\text{f-o}}/\alpha_{\text{fb}}|^2 = 1$ ($= |\alpha_{\text{f-o}}/\alpha_{\text{pr}}|^2$) there is a moderate reduction in performance (about 3%) in terms of maximum number of errors decoded with given block error probability. Decoder performance degrades dramatically for $|\alpha_{\text{f-o}}/\alpha_{\text{fb}}|^2 < 10^{-1}$.

---

et al., 2009) for a gallium phosphide photonic resonator and diamond nitrogen-vacancy system: $\tau_{\text{sw}} \approx 7\mu$s per switch at 1pW set/reset input power. For 1700 initial errors in Figure 2.8, we see (bottom panel) the time to decode scales as $1/\gamma^2$, suggesting that the time to correct errors that satisfy 2 out of 5 parity check constraints dominates the total decoding time. The feedback power to correct these errors is attenuated by a factor of γ^2, so the mean time for switching them is $\tau_{\text{sw}}/\gamma^2$. Setting the feedback power to 1pW, $\gamma = 10^{-2}$ (this is the highest value of γ shown in Figure 2.8 such that the decoder succeeds for most trajectories with 1700 initial errors), we estimate the time to decode to be $\sim \tau_{\text{sw}}/\gamma^2 = 7\mu\text{s}/(10^{-2})^2 = 70\text{ms}$, the energy to decode to be (time to decode)$\cdot$(input power) $= 70\text{ms} \cdot 1\text{pW} = 70\text{fJ}$ per latch; The set/reset inputs of each latch in the circuit receive 1pW input power, so this estimate gives the energy to decode per latch. There are 60,000 total latches in the decoder circuit for the parameters in Figure 2.8 (40,000 variables, 20,000 checks), so the total energy to decode is $\sim$ (energy per latch)$\cdot$(number of latches) $\approx 70\text{fJ} \cdot 60,000 \approx 4\text{nJ}$.

[7] We refer to the red curve in Figure 2.8, corresponding to the attenuation parameter $\gamma = 10^{-2}$. The simulations for Figure 2.8 used a higher probe amplitude ($\alpha_{\text{pr}} = 10^3$) than the finite fan-out power simulations for Figure 2.10 ($\alpha_{\text{pr}} = 10$), leading to slightly better performance in this metric.

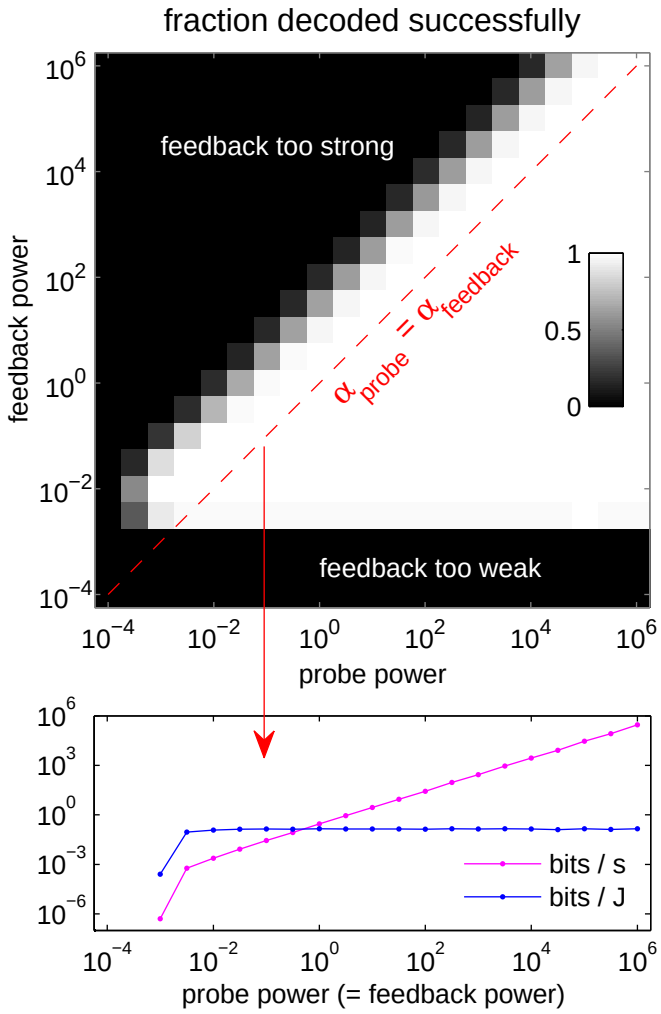

Figure 2.9: Decoder performance with varying probe and feedback power in the presence of component noise. (top, grayscale) Fraction of trials that decoded all of the initial errors succesfully vs. probe and feedback power. (top, red dashed line) fixed probe to feedback power ratio (ratio value 1). (bottom) Section of top plot (marked by red dashed line) corresponding to a fixed probe to feedbak power ratio. (bottom, magenta) mean decoding rate in bits/unit time. (bottom, blue) mean decoding energy (ratio of mean decoding rate and input power). Both performance measures are conditioned on successfully decoding all errors. Component spontaneous flip rate $\eta = 10^{-8}$. $\gamma = 0.01$. Fan-out amplitude $\alpha_{\text{f-o}} = \infty$. Expander LDPC code parameters: block length 40000, $l = 5$, $k = 10$, 1700 initial errors. We sampled 3000 trajectories per grayscale point.

Given the above observations, it is reasonable to set $\alpha_{\text{f-o}} = \alpha_{\text{fb}} = \alpha_{\text{pr}}$ in computing a energy budget for our circuit using actual physical components. This choice of $\alpha_{\text{f-o}}$ doubles the fiducial energy budget computed in Footnote 6 for a gallium phosphide photonic resonator and diamond nitrogen-vacancy system.

2.5 Discussion

We have described a photonic circuit that implements an iterative decoding scheme for expander LDPC codes. This circuit consists of a collection of optical latching relays, whose interactions via coherent fields map naturally onto the subroutines of the iterative decoder.

This circuit is autonomous—it is powered by the same optical signals that it acts upon to implement the decoding procedure, and it requires no external controller, measurement system, or clock signal. It operates robustly in the low-power limit in which quantum fluctuations of the optical fields are significant. The feedback-induced latch state fluctuations provide a natural source of randomness to drive the decoding algorithm. Crucially for the feasibility of such a system, our circuit's performance, as measured by decoding time and error rate, can be tuned smoothly by varying the optical input power. Tuning the input power can be done without loss in efficiency, as our circuit decodes a constant number of bits per Joule at a rate linear in the input power. Thus, noise that acts on the circuit components and potentially disrupts the computation can be overcome by increasing input power until the circuit works.

Our construction highlights the computational utility of cavity QED-based nanophotonic components for ultra-low power classical information processing, and points to the utility of the probabilistic graphical model framework in engineering autonomous optical systems that operate robustly in the quantum noise regime.

2.A Components

We describe the components we need for our decoder circuit in terms of a $(\mathbf{S}, \mathbf{L}, H)$ triplet, focusing on an intuitive input-output picture.

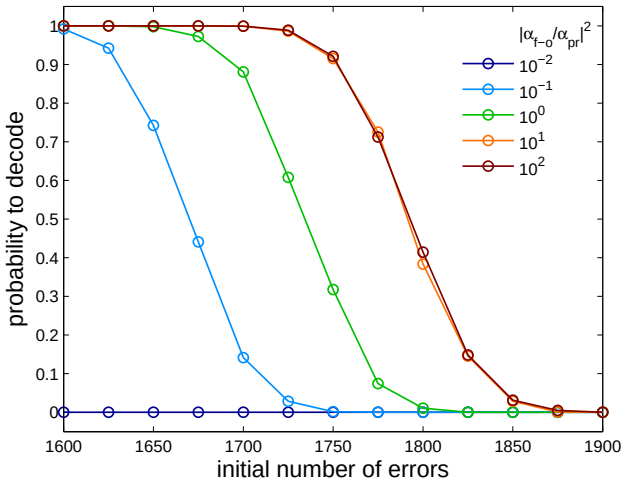

Figure 2.10: Decoder performance with varying fan-out amplitude $\alpha_{\text{f-o}}$. (colors) Probability to successfully decode vs. initial number of errors for several values of fan-out amplitude $\alpha_{\text{f-o}}$ (see legend box). The code parameters are the same as in (Sipser and Spielman, 1996): $(n = 40000, l = 5, k = 10)$. We set $\alpha_{\text{probe}} = \alpha_{\text{fb}} = 10$, $\eta = 10^{-80}$, and varied $\alpha_{\text{f-o}}$. We sampled 3000 trajectories for each data point.

2.A.1 Beamsplitter

To give an intuition for these systems and to specify a components we need, we first describe the beamsplitter as an open Markov quantum system. A 50/50 beamsplitter has two input and two output ports and is parametrized by:

$$B = \left(\mathbf{S} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad H = 0 \right) \tag{2.3}$$

By examining the scattering matrix, we see that for a field incident into input port 1, half the power is transmitted into output port 1 and half is reflected into output port 2 with a π phase shift. The beamsplitter has no internal degrees of freedom that concern us here, so $\mathbf{L} = \mathbf{0}$ and $H = 0$. The scattering matrix for a beamsplitter that transmits a fraction $\gamma < 1$ of incident power - our attenuation component - is a 2 by 2 rotation matrix with angle $\arccos \sqrt{\gamma}$.

2.A.2 Coherent input field

A coherent field input is modeled as a Weyl operator $W_{\vec{\alpha}}$, which displaces n vacuum inputs into coherent states $|\alpha_1\rangle, \ldots, |\alpha_n\rangle$ with amplitudes $\alpha_1, \ldots, \alpha_n$:

$$W_{\vec{\alpha}} = \left(\mathbf{S} = \mathbf{1}_{n \times n}, \quad \mathbf{L} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}, \quad H = 0 \right) \tag{2.4}$$

For example, driving the beam splitter above with $|\alpha\rangle$ in the first input and $|\beta\rangle$ in the second input results in the series connection:

$$\begin{aligned} &B \lhd W_{(\alpha,\beta)} = \\ &\left(\mathbf{S} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad \mathbf{L} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ -\alpha + \beta \end{pmatrix}, \quad H = 0 \right) \end{aligned} \tag{2.5}$$

resulting in the mixing of the two inputs in the two outputs, as we expect.

2.A.3 Latch

In terms of an $(\mathbf{S}, \mathbf{L}, \mathbf{H})$ triplet, the latch is given by the concatenation (parallel product) of two systems: $Q_{\text{set-reset}}$ accepts the set and reset inputs and drives the latch into the $|0\rangle$ or $|1\rangle$ state, and $Q_{\text{in-out}}$ routes the input fields $\text{in}_{1,2}$ into the output fields $\text{out}_{1,2}$. We have $Q = Q_{\text{set-reset}} \boxplus Q_{\text{in-out}}$, where

$$Q_{\text{set-reset}} = \left(\mathbf{S}_{\text{set-reset}} = \begin{pmatrix} \Pi_0 & -\sigma_{10} \\ -\sigma_{01} & \Pi_1 \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, H = 0 \right) \tag{2.6}$$

$$Q_{\text{in-out}} = \left(\mathbf{S}_{\text{in-out}} = \begin{pmatrix} \Pi_0 & -\Pi_1 \\ -\Pi_1 & \Pi_0 \end{pmatrix}, \mathbf{L} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, H = 0 \right) \tag{2.7}$$

where $\Pi_0 = |0\rangle\langle 0|$ and $\Pi_1 = |1\rangle\langle 1|$ are projection operators onto the $|0\rangle$ and $|1\rangle$ states and $\sigma_{01} = |0\rangle\langle 1|$, $\sigma_{10} = |1\rangle\langle 0|$ switch $|0\rangle$ and $|1\rangle$. Conditional on the state of the latch, either $\mathbf{S}_{\text{in-out}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ or $\mathbf{S}_{\text{in-out}} = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$, thus either switching or not switching the input fields. This is the same latch model as that used in our earlier work (Kerckhoff et al., 2010).

A possible physical system that achieves this desired behavior is shown in Figure 2.2 and was first proposed in (Mabuchi, 2009). The $|0\rangle$ and $|1\rangle$ states are degenerate ground states of an atom in a cavity. Set, reset, and input fields are resonant with transitions to one of two excited states ($|e\rangle$ and $|s\rangle$), from which the atom then decays back into one of the ground states. In a regime of strong atom-cavity coupling, the limiting behavior of the switch system is obtained by using the QSDE limit theorem (Bouten et al., 2008) to adiabatically eliminate the excited state dynamics. An alternate proposal for such a switch using a Kerr cavity is found in (Mabuchi, 2011).

2.B Bounds on nonlinearity of feedback

Consider a (n, l, k) LDPC code, and suppose there is only one variable v that needs to be flipped to return to a codeword. This variable participates in l parity check constraints, all of which are violated, so it flips at some maximal rate r. These l parity check constraints together include at most $l(k-1)$ variables other than v (at most because they may have some in common), each of which is involved in least one parity constraint violation, and so flips with rate at least $r\gamma^{l-1}$. The total rate for erroneously flipping any variable other than v is then

$$R_{\text{err}} = l(k-1)\ r\ \gamma^{l-1}$$

In order to flip v before errors accumulate, we set $r > R_{\text{err}}$ and find

$$\gamma < \left(\frac{1}{l(k-1)}\right)^{\frac{1}{l-1}}$$

In our numerical tests we have used $l = 5$, $k = 10$, yielding $\gamma < 0.38$. Numerically we found that our decoder mostly fails to decode already for $\gamma = 0.1$ (see Figure 2.8), but this is an upper bound assuming only one total error.

2.C Fan-in/Fan-out

As discussed in Section 2.3.2, our circuit requires a latch component $Q^{\text{var}}_{\mathbf{v}}$ corresponding to variable bit $\mathbf{v}$ to participate in multiple (l) parity check constraints, and a latch $Q^{\text{check}}_{\mathbf{c}}$ corresponding to parity check $\mathbf{c}$ to feed back to multiple (k) variables. Since the latch described in Section 2.3.1 and Appendix 2.A routes only two input and two output ports (in/out$_{1,2}$), it is insufficient for our needs: we need a latch that routes multiple in/out$_{1,2}$ signal pairs - switching each pair if and only if the latch state is $|1\rangle$ (see upper panel of Figure 2.11). We can augment our latch to achieve the desired fan-in/fan-out in two ways.

2.C.1 Routing multiple signals

One way is to simply add extra input/output ports to the latch system depicted in Figure 2.2: we could have input pairs $\text{in}_{1,2}^{(i)}$ for $i \in \{1, \ldots, N\}$ and the corresponding outputs (in addition to the two set/reset ports) for some integer N, all coupled to the same latch state. This would be difficult to achieve in a nanophotonic system, if only due to constraints of geometry - it would be difficult to have multiple beam paths access a single structure in a planar circuit.

An alternate scheme is depicted in Figure 2.11. The idea is to break up each latch into a set of N subsystems $Q_{\text{route}}^{(1)}, \ldots, Q_{\text{route}}^{(N)}$, each responsible for routing a single in/out signal pair, and a single subsystem Q_{sr} responsible for accepting the set/reset inputs (see Figure 2.11). Each of the $N+1$ subsystems is another latch, but one that routes only a single in/out signal pair and fits the description of Section 2.3.1. The set/reset subsystem Q_{sr} routes power (in orange path in Figure 2.11) to the set/reset ports of the N routing subsystems $Q_{\text{route}}^{(i)}$, driving the state of each routing subsystem to match the state of the set/reset subsystem. Thus the N routing subsystems $Q_{\text{route}}^{(i)}$ all mirror the overall system state, defined as the state of the set/reset subsystem Q_{sr}.

This construction introduces a delay in distributing the state of the set/reset subsystem to the N routing subsystems - due to both the waiting time for a routing subsystem to switch and to the time for a signal to propagate around a circuit (we do not model the latter source of delay for this circuit). The construction also introduces extra circuit components that could be subject to noise (e.g. spontaneously changing their state). We thus need to use high-enough fanout power (proportional to $|\alpha_{\text{f-o}}|^2$, in orange path in Figure 2.11) to make this construction useful.

2.C.2 Accepting multiple set/reset inputs

We note that we can use a similar construction to make a latch system that accepts multiple set/reset inputs in addition to routing multiple in/out signal pairs; though such a system does not appear in our decoder circuit, it may be useful for other purposes. When there are multiple set/reset input pairs for a device, these inputs

lose their interpretation as "set" and "reset" for the elecronic latch. We can instead associate each set/reset pair with an internal state and define an overall state as the sum modulo 2 of these internal states, so that changing any of the internal states changes the overall state. This behavior could be useful if we are interested in having a circuit component with multiple "flip" control inputs.

The idea is to break up the set/reset latch subsystem described above into a set of M subsystems $Q_{\text{sr}}^{(j)}$, $j \in \{1, \ldots, M\}$, each of which accepts only a single set of set/reset inputs. The overall system state is then defined as the sum modulo 2 of the set/reset subsystems, so flipping the state of any of them changes the overall state. The sum modulo 2 is performed as for the parity check circuit described in Section 2.3.2.1 and shown in Figure 2.3 (b). The probe beam path (black path in Figure 2.3 (b)) would now access each of the $Q_{\text{sr}}^{(i)}$ subsystems in sequence before driving the set or reset port of each of the $Q_{\text{route}}^{(j)}$ subsystems, as described in the previous Subsection (orange path in Figure 2.11).

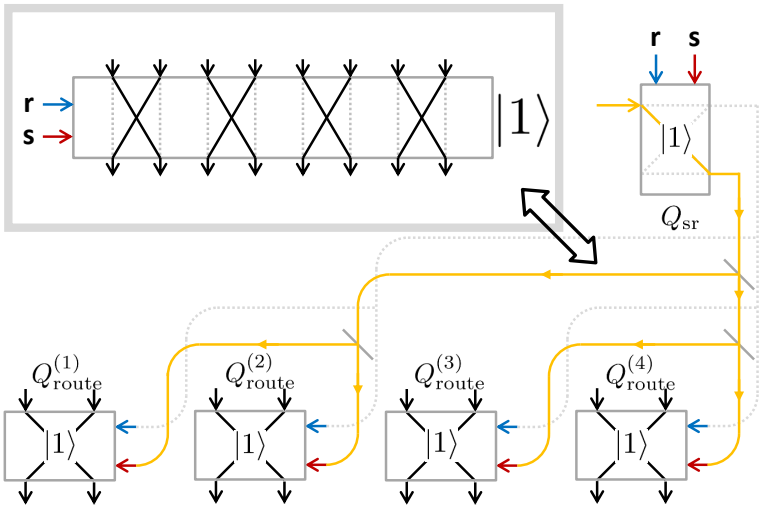

Figure 2.11: (top left inset) A latch that routes multiple in/out signal pairs, as needed by our decoder circuit. (main figure) implementation of latch in inset using only the single in/out signal pair latches described in Section 2.3.1. The routing latches $Q^{(i)}_{\text{route}}$ are each responsible for routing a single in/out signal pair. The set/reset latch Q_{sr} accepts external set/reset inputs and is responsible for distributing its state - the overall latch state - to the routing latches.

Chapter 3

Device Bestiary

In this Chapter I describe some photonic devices that are describable in the photonic circuit framework summarized in Chapter 1 and are not implausible using current technology. I provide examples where some of these may be useful for some computational tasks.

3.1 Stateless components

In this section I consider components that have no internal degrees of freedom — at least none that we dare to simulate. Everything here is built of mirrors, phase shifts, and beamsplitters. When we introduce feedback, the circuits acquire internal state (the circulating fed back mode). The SLH formalism does not easily handle this case; so long as the photon number in these fed back modes is small, our models should be ok.

3.1.1 The Mach-Zehnder interferometer

The humble Mach-Zehnder interferometer is the most ubiquitous and useful component in the fancier circuits that follow. This isn't surprising, since we seek to exploit the resource of coherence for computing and the interferometer routes signals conditional on their well-defined phase.

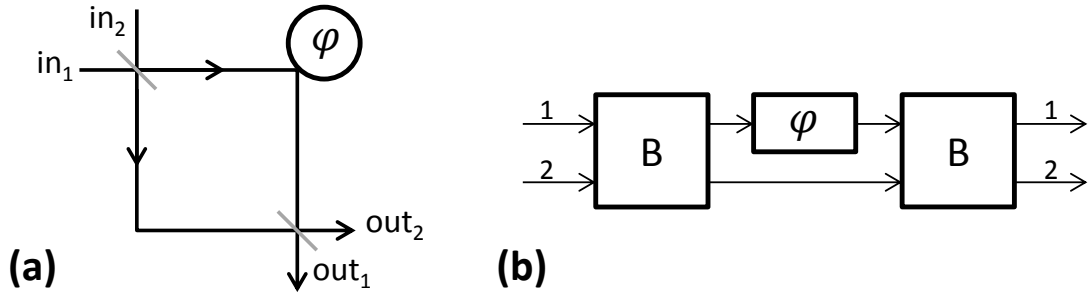

Figure 3.1: Mach-Zehnder interferometer with a phase of ϕ in one leg relative to the other. (a) Circuit schematic showing beam paths. 50/50 beamsplitters in gray. (b) "Music score" notation, where B denotes a 50/50 beamsplitter.

Figure 3.1 shows the Mach-Zehnder interferometer beam paths and the circuit in "music score" notation. The interferometer accepts two coherent input fields in_1 and in_2, mixes them on a beamsplitter, imparts a phase of ϕ in one of its legs relative to the other leg, and mixes the two legs on a second beamsplitter to produce the coherent output fields out_1 and out_2. We shall use a 50/50 beamsplitter in the following sections unless stated otherwise.

The SLH model (1.1.1) for this circuit is:

$$G_{\text{MZ}} = B \lhd (\phi \boxplus I_1) \lhd B = (S_{\text{MZ}}, L_{\text{MZ}}, H_{\text{MZ}}) \tag{3.1}$$

where

$$S_{\text{MZ}} = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} e^{i\phi} & \\ & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} -1+e^{i\phi} & 1+e^{i\phi} \\ -1-e^{i\phi} & 1-e^{i\phi} \end{pmatrix} \tag{3.2}$$

and $L_{\text{MZ}} = \begin{pmatrix} 0 & 0 \end{pmatrix}^\dagger$ and $H_{\text{MZ}} = 0$ are trivial since our Mach-Zehnder has no internal degrees of freedom (the latch of Chapter 2 is a MZ with internal degrees of freedom).

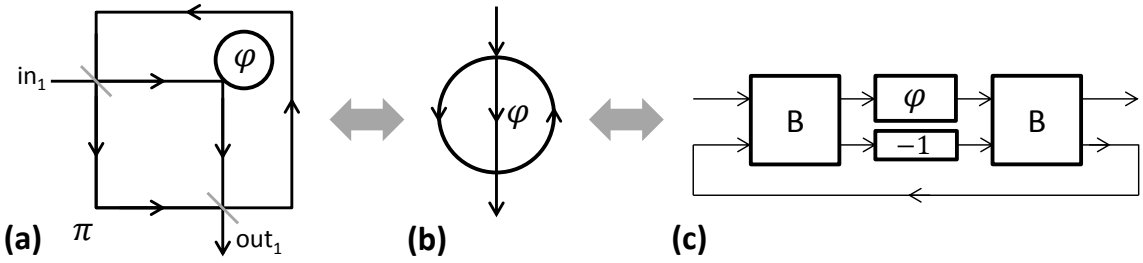

Figure 3.2: Inteferometer with feedback loop. Note the extra phase shift of π in one of the legs. (a) Circuit schematic. Note the π phase shift in the lower leg. (b) Equivalent short-hand circuit (π phase shift implied) (c) Music score notation.

Note that for the special case $\phi \in \{0, \pi\}$ we have

$$S_{\mathrm{MZ}}|_{\phi=0} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad S_{\mathrm{MZ}}|_{\phi=\pi} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.3}$$

so that the interferometer either switches or does not switch the two input signals (and applies a phase of π to the first input). We exploited this observation to construct a programmable latch in Chapter 2.

3.1.1.1 Feedback extensions

Let's consider feeding back to one of the interferometer ports. We do this because interesting things often happen when one considers adding feedback, and we describe this because we end up with a possibly useful cavity-like device.

Let's consider feeding back as shown in Figure 3.2. Here the second output out_2 is fed back into the second input in_1. Additionally, in contrast to the interferometer of Section 3.1.1, we add a phase shift of π in one of the legs. This change is not crucial to the results, and we could obtain the same result by omitting the π phase shift, but instead feeding back the out_2 to in_1 instead. This way we avoid wire crossings and the pictures look nicer. The π phase shift can be added by inserting an extra mirror in one leg of the interferometer.

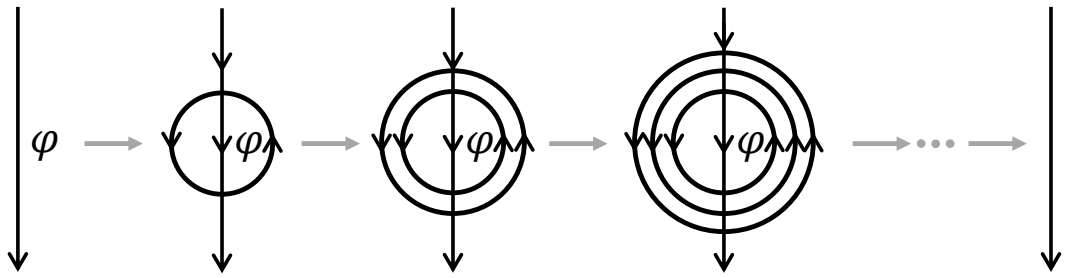

Figure 3.3: Iterated feedback operation starting from a single line imparting a phase shift of ϕ, to the single feedback line circuit of Figure 3.2, to the object with the trivial (identity) ϕ-independent scattering matrix (far right).

We compute the scattering matrix[1] for this device by taking the scattering matrix for the Mach-Zehnder we computed in (3.2) (taking care to add the additional π phase shift in one leg) and applying the feedback from out_2 to in_2. The circuit is given by:

$$G_{\text{FB}} = [B \lhd (\phi \boxplus (-1)) \lhd B]_{2\to 2} \tag{3.4}$$

And the scattering matrix is:

$$S_{\text{FB}} = \left[\frac{1}{2}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} e^{i\phi} & \\ & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}\right]_{2\to 2} \tag{3.5}$$

$$= \left[\frac{1}{2}\begin{pmatrix} 1+e^{i\phi} & -1+e^{i\phi} \\ 1-e^{i\phi} & -1-e^{i\phi} \end{pmatrix}\right]_{2\to 2} \tag{3.6}$$

$$= \frac{1}{2}\left(1+e^{i\phi}\right) + \frac{1}{2}\left(-1+e^{i\phi}\right)\left(\frac{1}{1-\frac{1}{2}\left(-1-e^{i\phi}\right)}\right)\frac{1}{2}\left(1-e^{i\phi}\right) \tag{3.7}$$

$$= \frac{1+3e^{i\phi}}{3+e^{i\phi}} \tag{3.8}$$

where the feedback operation in (3.7) is computed as given in Section 1.2.

Iterated feedback

What happens if we iterate this feedback operation? Why should we care? We shall see an application to construct a kind of cavity below.

The iterated feedback operation is shown in Figure 3.3. Denote by $G_{\mathrm{FB}}^{(n)}$ the circuit obtained by iterating the feedback operation n times:

$$G_{\mathrm{FB}}^{(0)} = \phi \tag{3.9}$$

$$G_{\mathrm{FB}}^{(n)} = \left[B \lhd \left(G_{\mathrm{FB}}^{(n-1)} \boxplus (-1) \right) \lhd B \right]_{2\to 2} \tag{3.10}$$

then with some work we can show that the scattering matrix obtained by applying the feedback operation n times (see Section 1.2) is the 1 by 1 matrix:

$$S_{\mathrm{FB}}^{(n)} = \frac{(2^n - 1) + (2^n + 1)e^{i\phi}}{(2^n + 1) + (2^n - 1)e^{i\phi}} \qquad n \geq 0 \tag{3.11}$$

The limiting scattering matrix as the number of feedback iterations increases $n \to \infty$ is

$$S_{\mathrm{FB}}^{(\infty)} = \lim_{n\to\infty} S_{\mathrm{FB}}^{(n)} = 1 \tag{3.12}$$

Some kind of cavity

This limit conceals, however, the dependence of $\arg S_{\mathrm{FB}}^{(n)}$ on the phase ϕ. This dependence is what makes this device potentially useful. Let's plot the argument of the (1 by 1) scattering matrix versus the phase ϕ. Figure 3.4 shows this. We see that the "output" phase ($\arg S$) depends increasingly strongly on the "input" phase (ϕ) for ϕ near a certain value (π) as the number of feedback iterations n increases. This is similar to the behavior of a cavity and makes sense for the present device: the feedback loop creates an optical mode that can circulate in the device, so a photon in the circulating mode can "see" the phase of ϕ multiple times, so the device thus imparts an overall phase different from ϕ.

[1] The scattering matrix is again the only nontrivial part of this SLH model since there are no internal degrees of freedom, so that L and H are trivial (0).

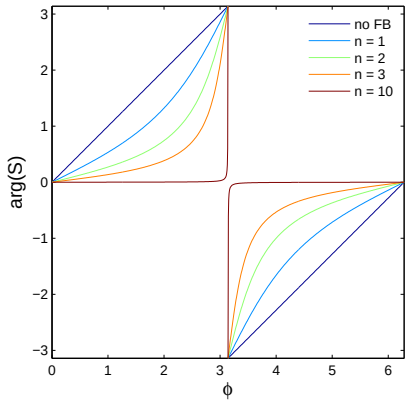

Figure 3.4: Phase of scattering matrix $S_{\text{FB}}^{(n)}$ obtained by applying the feedback operation n times (3.11) versus the no-feedback phase $\arg S_{\text{FB}}^{(0)} = \arg e^{i\phi} = \phi$.

Since $S_{\text{FB}}^{(n)}$ has period of 2π in ϕ, $\arg S_{\text{FB}}^{(n)}$ must change by 2π as ϕ varies from 0 to 2π. On the other hand, since $S_{\text{FB}}^{(\infty)} = 1$, we must have $\arg S_{\text{FB}}^{(\infty)} \to 0$ wherever the arg function is continuous in its argument. Thus we expect to find an increasingly sharp kink near $\phi = \pi$ in $\arg S_{\text{FB}}^{(n)}$ as n grows, and a discontinuity for $n \to \infty$. This is consistent with what we see in Figure 3.4. We lose sensitivity to the innermost phase ϕ for most values of ϕ, but gain sensitivity for ϕ near π.

We measure the sharpness of the kink in $\arg S_{\text{FB}}^{(n)}$ by its derivative. With some work we can show this to be

$$\arg S_{\text{FB}}^{(n)} = \arctan\left(\frac{2^{n+1}\sin\phi}{2^{2n}(\cos\phi+1)+\cos\phi-1}\right) \tag{3.13}$$

$$\Downarrow$$

$$\frac{d}{d\phi}\arg S_{\text{FB}}^{(n)} = \frac{2^{n+1}}{2^{2n}(\cos\phi+1)-\cos\phi+1} \tag{3.14}$$

With some more work we can show that the quantity in (3.14) is maximized for $\phi = \pi$:

$$\left.\frac{d}{d\phi}\arg S_{\text{FB}}^{(n)}\right|_{\phi=\pi} = 2^n \tag{3.15}$$

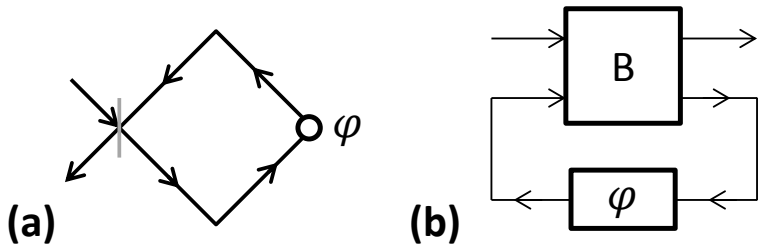

Figure 3.5: Feedback loop to a 50/50 beamsplitter through a ϕ phase shift. (a) Circuit schematic. (b) Music-score notation, rearranged slightly to emphasize the controller-plant analogy.

so the sharpness of the kink grows exponentially in the number of feedback iterations n and the device acts like a cavity.

3.1.2 Poor man's cavity

Let's consider the feedback with phase shift setup shown in Figure 3.5. This device is similar to the feedback with phase shift scheme in Section 3.1.1.1, but has one fewer beamsplitter and thus has fewer optical paths. It is also a canonical device in the sense that it is the simplest example of a feedback device; alternately, we can think of the beamsplitter as a controller for the plant, which in this case has no degrees of freedom and applies a phase shift of ϕ to an incoming signal. The control theory perspective on this device is presented in (Gough and James, 2009) (see Figure 5 in that paper).

Let's compute the SLH model for this device. Denoting the circuit by G we have

$$G = [(I_1 \boxplus \phi) \lhd B]_{2\to 2} \tag{3.16}$$

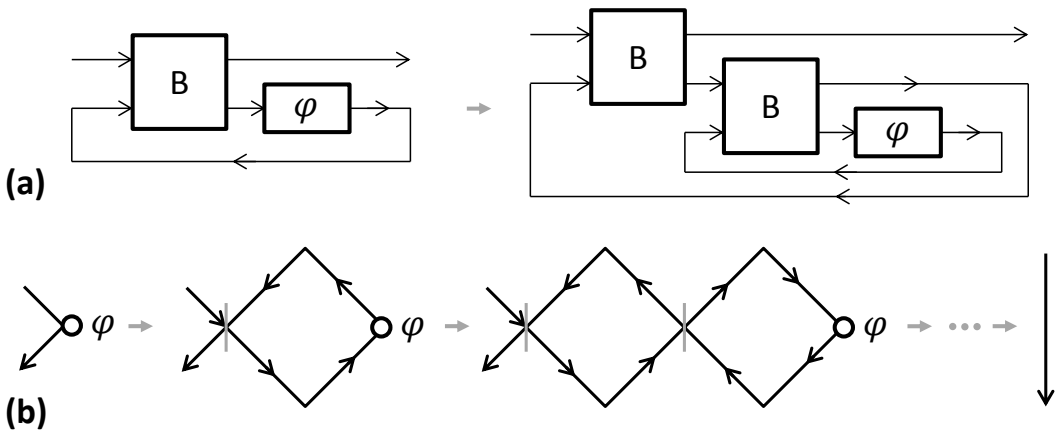

Figure 3.6: Iterated feedback operation for the poor man's cavity of Figure 3.5. (a) Iterations $n = 1$ and $n = 2$ in music score notation. (b) Iterations $n = 0, 1, 2, \ldots, \infty$. Gray lines denote 50/50 beampslitters. The scattering matrix for the $n \to \infty$ case is trivial (identity).

so the scattering matrix is (since there are no internal degrees of freedom, the coupling vector L and the Hamiltonian H are trivial for this device)

$$S = \left[\begin{pmatrix} 1 & \\ & e^{i\phi} \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \right]_{2\to 2} = \left[\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -e^{i\phi} & e^{i\phi} \end{pmatrix} \right]_{2\to 2} \tag{3.17}$$

$$= \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}} \left(\frac{1}{1 - e^{i\phi}/\sqrt{2}} \right) \frac{-e^{i\phi}}{\sqrt{2}} \tag{3.18}$$

$$= \frac{\sqrt{2} - 2e^{i\phi}}{2 - \sqrt{2}e^{i\phi}} \tag{3.19}$$

where the feedback operation in (3.18) is computed as given in Section 1.2.

Iterated feedback

As in the previous Section, let's consider what happens when we iterate the feedback operation, using the above device to stand in for the inner phase shift. The iterated feedback operation is shown in Figure 3.6. Reasoning as in the previous Section, we

might expect the iterated feedback device to boost sensitivity to the inner phase ϕ for some values of ϕ and depress sensitivity elsewhere, thus acting as a cavity.

Denote by $G_{\mathrm{FB}}^{(n)}$ the circuit obtained by iterating the feedback operation n times:

$$G_{\mathrm{FB}}^{(0)} = \phi \tag{3.20}$$

$$G_{\mathrm{FB}}^{(n)} = \left[\left(I_1 \boxplus G_{\mathrm{FB}}^{(n-1)}\right) \lhd B\right]_{2\to 2} \tag{3.21}$$

then with some work we can show that the scattering matrix obtained by applying the feedback operation n times is the 1 by 1 matrix:

$$S_{\mathrm{FB}}^{(n)} = \frac{b_n\sqrt{2} + a_n e^{i\phi}}{a_n\sqrt{2} + b_n e^{i\phi}} \qquad n \geq 0 \tag{3.22}$$

where a_n and b_n for $n \geq 1$ are the sequences of the numerators and denominators to the upper principal and intermediate convergents to $\sqrt{2}$, OEIS A143609 and A084068, respectively (Kimberling, 1997). The first few values are (extending the formulas for a_n and b_n to the $n = 0$ case) $(a_n, b_n) = (1,0), (2,1), (3,2), (10,7), (17,12)$ for $n = 0, 1, 2, 3, 4$.

The limiting scattering matrix as the number of feedback iterations increases $n \to \infty$ is

$$S_{\mathrm{FB}}^{(\infty)} = \lim_{n\to\infty} S_{\mathrm{FB}}^{(n)} = 1 \tag{3.23}$$

As for the Mach-Zehnder with feedback Section 3.1.1.1, let's consider the dependence of $\arg S_{\mathrm{FB}}^{(n)}$ on the phase ϕ, since this yields the overall phase imparted by the device. We plot in Figure 3.7 the argument of the scattering matrix versus the phase ϕ. As in Section 3.1.1.1, we see that the "output" phase ($\arg S$) depends increasingly strongly on the "input" phase (ϕ) for $\phi \approx \pi$ as the number of feedback iterations n increases.

We might expect that the extra feedback loops somehow hide the innermost phase ϕ from an interrogating input field; on the other hand, any light that does leak all the way to the innermost phase bounces around many times before it leaks back out, and so "sees" the phase ϕ many times. Since the overall scattering matrix is periodic in

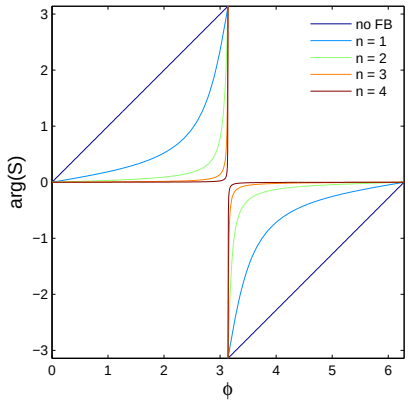

Figure 3.7: Phase of scattering matrix $S_{\mathrm{FB}}^{(n)}$ obtained by applying the feedback operation n times (3.22) versus the no-feedback phase $\arg S_{\mathrm{FB}}^{(0)} = \arg e^{i\phi} = \phi$.

ϕ, sensitivity of $\arg S$ to ϕ (measured by its derivative) that we lose for some values of ϕ we must gain for other values of ϕ.

Let's compute the derivative of $\arg S_{\mathrm{FB}}^{(n)}$ to measure the sharpness of the kink near $\phi \approx \pi$. With some work we can show this to be

$$\arg S_{\mathrm{FB}}^{(n)} = \arctan\left(\frac{2\sin\phi}{\cos\phi\,(p+m)+p-m}\right) \tag{3.24}$$

where $p = (\sqrt{2}+1)^{2n}$ and $m = (\sqrt{2}-1)^{2n}$, and

$$\frac{d}{d\phi}\arg S_{\mathrm{FB}}^{(n)} = \frac{2(\cos\phi(p-m)+p+m)}{(\cos\phi(p+m)+p-m)^2} \tag{3.25}$$

The quantity in (3.25) is maximized for $\phi = \pi$:

$$\left.\frac{d}{d\phi}\arg S_{\mathrm{FB}}^{(n)}\right|_{\phi=\pi} = \left(3+2\sqrt{2}\right)^n \tag{3.26}$$

so the sharpness of the kink grows exponentially in the number of feedback iterations n, and somewhat faster than for the Mach-Zehnder with feedback (3.15).

How can we use such a device? We can imagine using it to enhance the sensitivity of a measurement of a small phase, just like a cavity.

3.2 Components with state

We now turn to optical circuits components that have an internal state that interacts with incoming and outgoing optical fields[2]. We would like to have such devices at our disposal in order to make circuits whose time evolution depends on past circuit states, rather than only upon the instantaneous inputs to the circuit, thus enabling, e.g., a device that acts as a memory. In SLH terms (see Section 1.1.1 for definitions), we would like components with a nontrivial coupling vector L and Hamiltonian H.

Stateful optical circuits also enable two useful things that are not immediately obvious: nonlinearity and the interaction of optical fields with each other. First, by nonlinearity I mean the nonlinear dependence of power in outgoing versus incoming optical fields. The dynamics of systems with no internal state (trivial L and H) are determined completely by their scattering matrix S (see Section 1.1.1). Since S is unitary, the norm (proportional to power) of a vector of optical field inputs is preserved at all times. For a nonlinear circuit, that power power has to go somewhere, and that could be the internal degrees of freedom of a stateful object (e.g., in an optomechanical system, light excites phonon modes, which are in turn dissipatively coupled to an environment). Even an empty cavity can temporarily violate input-output power conservation while the field inside the cavity builds to a steady-state value.

Second, since optical fields do not directly interact with each other, an intermediary system is necessary for the state of one mode to impact another. The evolution of a stateless system is determined completely by the action of the unitary scattering matrix $\mathbf{S}$ on a vector of optical inputs (see Section 1.1.1); thus the only "interaction" permitted in a stateless system is the addition of optical fields. More complicated interactions require an internal state to store information from one field mode and pass it on to another, thus requiring our system to have a nontrivial (non-zero) coupling $\mathbf{L}$

[2] BS already has internal state, but we do not model it

vector. A nontrivial system Hamiltonian could enable more useful dynamics for the stateful component (e.g., in an atom-in-a-cavity system, external fields are coupled to the cavity mode (via the **L** vector), which is coupled to the atomic degrees of freedom (via the Hamiltonian)).

Further motivation: Nonlinearity and the interaction of different field modes are necessarry features for the implementation of universal logic. Suppose we think of the phase of our coherent fields as encoding a binary value (e.g., $0 \leftrightarrow 0$, $\pi \leftrightarrow 1$). Then we can implement a NOT gate by bouncing a beam off a mirror ($\phi \rightarrow \phi + \pi$, so $x \rightarrow x \oplus 1$). We could implement an OR gate by mixing two coherent fields of equal power on a 50/50 beamsplitter (all of the light will leave from one port if the two phases add to 0 modulo π, and from the other port otherwise). But NOT and OR are insufficient for universal logic. We could ask for an AND gate to make a universal set of gates, but this is a nonlinear device - the output is only high when both inputs are high, and 0 otherwise. A proposal for a NAND gate sufficient for universal logic) using stateful, nonlinear devices (Kerr cavities) is found in (Mabuchi, 2011).

We further observe that there is a tradeoff between the nonlinearity of a device and the latency associated with its operation. When we wed the function of a device to its internal dynamics and interactions with external fields, we are limited by the timescales of those internal dynamics internal-field interaction dynamics. Purely scattering/linear/passive devices (like beamsplitters or phase shifts) operate instantaneously in our framework. Of course, even a mirror has internal degrees of freedom (conducting electrons that radiate a reflected field), but we do not model these or treat them as very fast compared to other degrees of freedom in our circuits (like atomic states coupled to cavity modes).

In this Section we present the simplest stateful components - empty optical cavities, atoms coupled to optical modes - these we combined to form a latch device in Chapter 2. Along the way we illustrate some uses for these circuits for some computational tasks.

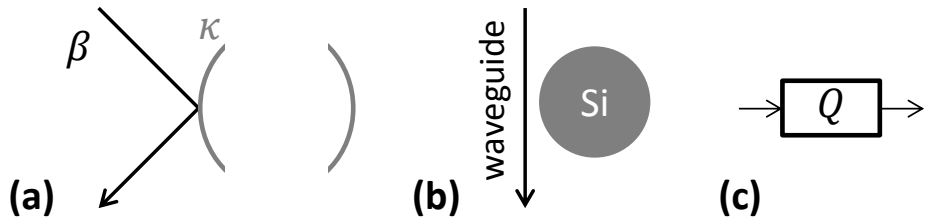

Figure 3.8: (a) Circuit schematic for a single-port cavity with cavity field decay rate κ driven by coherent field with amplitude β. (b) A cartoon showing a silicon microring resonator coupled to a waveguide. (c) Music-score notation for a single-port cavity.

3.2.1 Empty cavities

A basic component that will recur below is the empty cavity coupled to an external coherent driving field. Figure 3.8 shows the common setup of a single coherent optical mode with amplitude β coupled to a single-port cavity. The state space for the cavity is a Fock space, and the states (density matrices over this Fock space) correspond roughly to probability distributions for the number of photons in the cavity[3]. The incoming field mode contributes photons to the cavity mode, and photons from the cavity mode contribute to the outgoing field. A physical realization would be a free space laser beam bouncing off a cavity formed by two mirrors or, as shown in the cartoon in Figure 3.8 (b), a microring resonator coupled to a nearby waveguide.

The geometry of a cavity, say, the distance between the two opposing mirrors in a Fabry-Pérot cavity, establishes the set of resonant frequencies for the cavity, since only wavelengths that divide the cavity length a half integer number of times can form standing waves in the cavity. In what follows we shall mostly be interested in only a single one of these supported frequencies and refer to the corresponding cavity mode as "the cavity mode."

[3] "Roughly" in the sense of the Q-function, $Q(\alpha) = \frac{1}{\pi}\langle\alpha|\rho|\alpha\rangle$, where $|\alpha\rangle$ is a coherent state.

The SLH model for the undriven empty cavity is given by:

$$\mathbf{S} = \mathbf{1}_{1\times 1} \tag{3.27}$$
$$\mathbf{L} = \sqrt{2\kappa}\, a \tag{3.28}$$
$$H = \Delta\, a^\dagger a \tag{3.29}$$

where a is the annihilation operator for the cavity Fock space, $a^\dagger$ is the adjoint of a, κ is the cavity field decay rate, and $\Delta = \omega_\mathrm{c} - \omega_\mathrm{p}$ is the detuning between the resonant cavity frequency and incoming probe field frequency. The L term corresponds to the process of a photon leaking out of the cavity into the outgoing field, and $L^\dagger$ corresponds to an incoming photon leaking into the cavity.

When the empty cavity is driven by a coherent field with amplitude β, the driven circuit expression is given by:

$$Q_\mathrm{driven} = Q \lhd W_\beta \tag{3.30}$$

where W_β is the Weyl operator. We compute the driven SLH model:

$$\mathbf{S} = \mathbf{1}_{1\times 1} \tag{3.31}$$
$$\mathbf{L} = \sqrt{2\kappa}\, a + \beta \tag{3.32}$$
$$H = \Delta\, a^\dagger a + i\sqrt{\kappa/2}\left(a\beta^* - a^\dagger\beta\right) \tag{3.33}$$

where β^* is the complex conjugate of β. Note that the second term in the Hamiltonian corresponds to the exchange of photons between the cavity mode and the external field mode.

In the Sections below I present several devices that use only the empty cavity driven by coherent inputs as their sole building block and are useful for something.

3.2.2 Cavity beampsplitter

A single empty cavity can be useful as a beamsplitter. Consider the arrangement shown in Figure 3.9. Here a single cavity is coupled to two waveguides, cavity decay

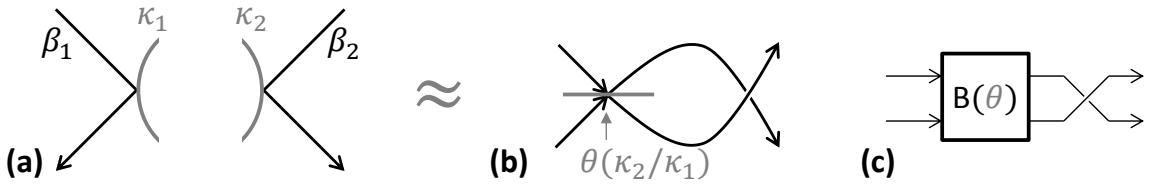

Figure 3.9: (a) Circuit schematic for a two-port cavity with cavity field decay rates κ_1, κ_2 driven by coherent fields with amplitudes β_1, β_2. (b) For time-invariant coherent input fields, the circuit in (a) is equivalent to a beamsplitter with mixing angle $\theta(\kappa_1/\kappa_2)$ given by (3.45) followed by a switch of the outgoing fields. For time-varying fields, there is a transient behavior on a timescale of $\max(1/\kappa_1, 1/\kappa_2)$. (c) Music-score notation.

rates κ_1 and κ_2. We shall see that this arrangement can act as a beamsplitter of arbitrary mixing angle.

One may wonder why one would ever use a cavity instead of a beamsplitter given that: 1) a beamsplitter works for a range of optical frequencies, while our cavity requires that the incoming fields be resonant with the cavity and 2) a beamsplitter works "instantaneously" while with cavities we must wait for a transient behavior to disappear. On the other hand, it may be natural to work only a single frequency in an optical setup, and beamsplitting materials might be hard to come by or manufacture.

Let's work out the SLH model for this device, then the Heisenberg equations of motion for the output fields, and then the steady-state transfer matrix from input to output fields. Denote by G the circuit obtained by driving the two-sided cavity with coherent input fields with amplitudes β_1, β_2:

$$G = Q_{2\text{-port}} \lhd (W_{\beta_1} \boxplus W_{\beta_2}) \tag{3.34}$$

The SLH model is

$$\mathbf{S} = \mathbf{1}_{2\times 2} \tag{3.35}$$

$$\mathbf{L} = \begin{pmatrix} \sqrt{2\kappa_1}\ a + \beta \\ \sqrt{2\kappa_2}\ a^\dagger + \beta^* \end{pmatrix} \tag{3.36}$$

$$H = i\sqrt{\kappa_1/2}\left(a\beta_1^* - a^\dagger\beta_1\right) + i\sqrt{\kappa_2/2}\left(a\beta_2^* - a^\dagger\beta_2\right) \tag{3.37}$$

where we have chosen $\Delta = 0$ (see (3.29)). With some work[4] we can show the Heisenberg equation of motion for the cavity field annihilation operator a is given by:

$$\langle \dot{a} \rangle = \mathrm{Tr}\left[a\dot{\rho}\right] = \mathrm{Tr}\left[a\left(-i[H,\rho] + \sum_{i=1}^{2}\left(L_i\rho L_i^\dagger - \frac{1}{2}\left\{L_i^\dagger L_i, \rho\right\}\right)\right)\right] \tag{3.38}$$

$$= -\left(\left(\kappa_1 + \kappa_2\right)\langle a\rangle + \left(\sqrt{2\kappa_1}\ \beta_1 + \sqrt{2\kappa_2}\ \beta_2\right)\right) \tag{3.39}$$

where we substituted the master equation (1.1) for $\dot{\rho}$ and $\langle\rangle$ denotes expectation with respect to the density matrix ρ.

In steady state, we set $\langle \dot{a} \rangle = 0$ to obtain the steady state cavity field:

$$\langle a \rangle = -\frac{\sqrt{2\kappa_1}\ \beta_1 + \sqrt{2\kappa_2}\ \beta_2}{\kappa_1 + \kappa_2} \tag{3.40}$$

The output from the i-th port is given by $\langle L_i^\dagger L_i \rangle$, where L_i is given in (3.36). Because 1) the two input fields are both coherent states $|\beta_1\rangle$, $|\beta_2\rangle$, 2) the Hamiltonian consists only of terms proportional to the cavity mode annihilation operator a or its adjoint, and 3) coherent states are eigenstates of the annihilation operator, the output fields must be coherent states as well, and therefore satisfy $\langle L_i^\dagger L_i\rangle = \langle L_i^\dagger\rangle\langle L_i\rangle$ [5]. We

[4] This takes some work by hand; I used the QNET package developed by Nikolas Tezak and others (Tezak et al., 2012) to do the computation.

[5] $\langle a^\dagger a\rangle = \mathrm{Tr}\left[a^\dagger a\rho\right] = \mathrm{Tr}\left[a|\alpha\rangle\langle\alpha|a^\dagger\right] = |\alpha|^2\,\mathrm{Tr}\left[\rho\right] = |\alpha|^2 = \langle a^\dagger\rangle\langle a\rangle$.

compute these quantities using (3.36) and (3.40):

$$\langle L_1 \rangle = \frac{\beta_1 (\kappa_2 - \kappa_1) - 2\beta_2 \sqrt{\kappa_1 \kappa_2}}{\kappa_1 + \kappa_2} \tag{3.41}$$

$$\langle L_2 \rangle = \frac{\beta_2 (\kappa_1 - \kappa_2) - 2\beta_1 \sqrt{\kappa_1 \kappa_2}}{\kappa_1 + \kappa_2} \tag{3.42}$$

Using the above equations we find that in steady state:

$$\left\langle \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} \right\rangle = \begin{pmatrix} \frac{\kappa_2 - \kappa_1}{\kappa_1 + \kappa_2} & -2\frac{\sqrt{\kappa_1 \kappa_2}}{\kappa_1 + \kappa_2} \\ -2\frac{\sqrt{\kappa_1 \kappa_2}}{\kappa_1 + \kappa_2} & \frac{\kappa_1 - \kappa_2}{\kappa_1 + \kappa_2} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} \tag{3.43}$$

$$= \begin{pmatrix} -\frac{1-r}{1+r} & -2\frac{\sqrt{r}}{1+r} \\ -2\frac{\sqrt{r}}{1+r} & \frac{1-r}{1+r} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} \tag{3.44}$$

where $r = \kappa_2/\kappa_1$. Now setting

$$\theta \equiv \arcsin\left(\frac{-1+r}{1+r}\right) \quad \Leftrightarrow \quad r = \frac{1+\sin\theta}{1-\sin\theta} \quad \text{for } \theta \in (-\pi/2, \pi/2) \tag{3.45}$$

we can rewrite the relation in (3.44) as

$$\langle \mathbf{L} \rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \vec{\beta} \tag{3.46}$$

This is the product of a reflection and a rotation matrix. Thus the steady state output amplitudes are obtained by rotating the inputs by $\theta(r = \kappa_2/\kappa_1)$ and then swapping the two outputs with each other, thus justifying the picture in Figure 3.9 of a beampsplitter followed by a swap.

Note that while the scattering matrix of a beamsplitter is a rotation matrix, the scattering matrix of this empty cavity device is trivial (identity) (3.35). On the other hand, a beamsplitter has a trivial Hamiltonian, while the cavity device does not (3.37). It turns out that the $\mathbf{L}$ vector (which determines the output fields) of a beamsplitter driven by coherent inputs matches the expected value of the $\mathbf{L}$ vector of our cavity device. This equality only holds in the steady state condition $\langle \dot{a} \rangle = 0$; if the amplitudes of the coherent inputs to the cavity device switch suddenly,

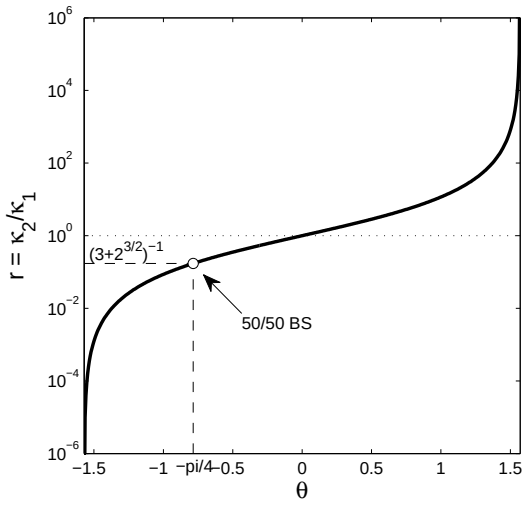

Figure 3.10: (solid line) The ratio $r = \kappa_2/\kappa_1$ vs mixing angle $\theta \in (-\pi/2, \pi/2)$, given in (3.45). The dot shows the value of $r = (3+2\sqrt{2})^{-1}$ at which $\theta = -\pi/4$ and the device acts like a 50/50 beamsplitter followed by a switch of the two outputs. The other (orthogonal) 50/50 beamsplitter corresponds to $\theta = \pi/4$, $r = (3+2\sqrt{2})$ (not shown).

there is a transient behavior violating the beamsplitter similarity on a timescale of $\max(1/\kappa_1, 1/\kappa_2)$. We may thus want to engineer large values of κ_1, κ_2 to reduce the time that the device "misbehaves." Larger values of κ require, in the case of microring resonators, closer placement of the cavity to a waveguide, which may be difficult to accomplish reliably.

Let's plot the dependence of the cavity field decay ratio $r = \kappa_2/\kappa_1$ and the mixing angle $\theta \in (-\pi/2, \pi/2)$ in Figure 3.10. We see that any value of the mixing angle θ is attainable by an appropriate choice of κ_2/κ_1. Two values are of special interest:

- A 50/50 beamsplitter corresponds to $\theta = -\pi/4 \Rightarrow 1/r = \kappa_1/\kappa_2 = 3+2\sqrt{2}$. It's fun to note that this is the same constant as the maximum sensitivity of the beamsplitter with feedback "poor man's cavity" (c.f. Section 3.1.2) to an inner phase. For a nanophotonic circuit, we could control κ_2/κ_1 by controlling the distance from two waveguides to a microring resonator.

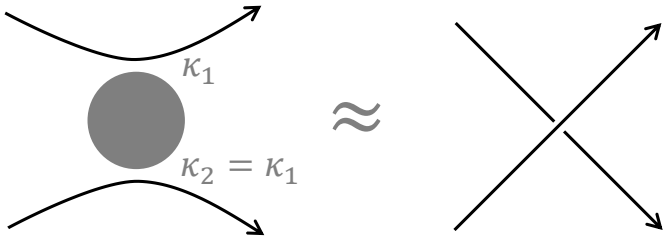

Figure 3.11: Two nanophotonic waveguides (black) approach a microring resonator (gray) and come apart having swapped states via the cavity. The empty cavity device with $\kappa_1 = \kappa_2 \Rightarrow \theta = 0$ corresponds to switching the two input fields for coherent input fields in steady state.

- Setting $\kappa_1 = \kappa_2 \Rightarrow \theta = 0$ corresponds to no rotation followed by a switch of the two outputs. This can be useful to implement a waveguide crossing in a nanophotonic circuit; instead of crossing two waveguides we would instead make the two waveguides come near a microring resonator and then come apart again as shown in Figure 3.11. It could be difficult to engineer $\kappa_1 = \kappa_2$ within a given tolerance, however. Again, higher κ is desirable due to a shorter transient behavior duration.

Part II

Probabilistic, energy-efficient computing

Introduction

In this part I consider encoding information in the realization of a random signal in several photodection-inspired settings. Why? The search was initially motivated by the desire to use optical circuits in ways beyond emulating the digital logic devices pervasive today. Of course, even getting to the point of reliably emulating a single logical gate in an optical device optically is hard, but on the other hand we could try something else. Optical signals present a new resource, giving us an opportunity to think about what kind of computing is natural in this setting.

Another motivation is to design noise-tolerant systems. In the early days of computing, before reliable components could allow us to forget the noisy nature of components at a higher level of abstraction, fault-tolerant computing was a hotter topic[6]. These days noise is reappearing as power consumption is becoming an ever larger constraint on computing. Optical signals at low photon number are unavoidably noisy for reasons of quantum mechanics. It would be nice to represent information in a way that deals gracefully with the kind of noise we encounter in optical systems and has noise-tolerance built in, allowing us to trade the energy, speed, and correctness of computations[7].

Finally, this part is here because some of the questions I present have neat information-theoretic answers, and are fun on their own. Some of these results also have applications for things other than photonic circuits; e.g. the distribution coding schemes of Chapter 4, the scheme of Chapter 5 for the additive white Gaussian noise

[6] In particular, a foundational set of lecture notes by von Neumann established a theoretical framework for robust, noisy computation (von Neumann, 1956).

[7] "Correctness" could stand for, say, the probability to successfully correct the output of a noise channel, as for the LDPC decoder circuit of Chapter 2.

(AWGN) channel in the low signal to noise regime, and the random linear constraints coding scheme of Chapter 6 could be useful in storing memories in flash cells efficiently with respect to energy. I discuss these connections as they arise.

I begin in Chapter 4 with a toy model that captures the counting noise involved in photon detection and not much else. I present this as a problem in channel coding[8] and look for schemes that come close to capacity. Even this setup turns out to be too hard for me, so I consider a quantized version of this model that is more tractable (the quantization amounts to ranking the optical fields by their intensities - permutation quantization). I provide some results of a large deviations flavor for this toy model and suggest some natural, physically realizable extensions.

In Chapter 5 I consider energy-efficient communication. The idea is that some commonly used channel models have a parameter that acts like a power constraint (like a mean power constraint for the AWGN or peak rate constraint for the Poisson channel). How can we use a given power budget most efficiently over multiple transmissions? Suppose we give ourselves the freedom to either "split" the power available to a single channel into multiple noisier channels, or to do the reverse thing, "aggregate" identical noisy channels into a single less noisy channel. In some settings this is a natural operation (e.g., splitting optical power into different waveguides or frequency bands, or increasing the number of flash memory cells while holding the total charge fixed); is it worth doing? In terms of channel capacity the answer can be positive. We find that it is reasonable to say a channel is "power-efficient" at power P^* when its capacity scales linearly for values of the power near P^*, motivating the search for these efficient points and splitting and aggregation as a way of getting to them. We further investigate this question in the finite block length regime. Finally, Chapter 5 presents a capacity-achieving distribution for the AWGN at low power that might seem counterintuitive - the distribution is a "spike train," where the sender either sends 0 power most of the time or a lot of power rarely. These observations lead to a proposal of a communication scheme where the sender communicates by encoding a message in the location of the spikes.

[8] Storing information in a computer memory can be cast as a channel coding problem: the sender writes to memory and the receiver later reads out a possibly corrupted version, possibly with measurement noise.

Finally, in Chapter 6 I consider information representation schemes inspired by random constructions in graphical models and error correcting codes. The idea is to use objects with many degrees of freedom, encode message in binary string of properties of those objects, and then measure those binary properties at the output. Hopefully, an ill-understood channel for these objects turns into a binary symmetric channel on the strings of properties.

Chapter 4

Distribution coding

I consider a toy model that attempts to generalize a common setup in stochastic computing and to capture the counting statistics arising in setups involving photon detection. Let's start with the physical representation of a bit in a computer — a voltage level that takes on one of two values. For the sake of symmetry, niceness of the equations, and energy conservation, let's suppose instead that each bit is represented by two "wires" or "bins" [1] exactly one of which is high at a given time and the other is low.

What are natural perturbations to this picture? One can imagine that the bits are noisy: perhaps the voltage fluctuates in time so that a "1" sometimes looks more like a "0." Alternately, perhaps the one attempts to encode more information in the bit by using more voltage levels than {high,low}. One could also have more wires than two. We are also interested in power-limited information representation. Perhaps the mean energy per wire must decrease as the number of wires grows to conserve total energy. Let's summarize the picture so far:

- We have a collection of wires that each carries some amount of stuff (like electrons or photons).
- The total amount of stuff is constrained somehow (per unit of time, say).

[1] I'm going to call them "wires" or "bins" (to emphasize the counting statistics to come) from now on; this could refer to optical beam paths, frequency bands, or cells in a flash memory, depending on the setting.

- We measure the amount of stuff in each wire in a noisy way.

4.1 The multinomial channel

Let's choose a concrete model with the above features that has something to do with photon detection. When a photodetector is exposed to light, we do not get a direct measurement of the light's power; instead we get a stream of "clicks" - photodetection events - at a rate proportional to the power. The longer one waits for the clicks to accumulate, the better an estimate one has of the power.

Let's suppose that we have some total amount of power split between k wires. We think of the particular way in which the power is split (a probability mass function) as the channel input. We wait a while and tally the clicks received in each wire, so the channel output is a histogram that counts the clicks per wire.

How long do we wait before terminating observation of the wires? We could wait until we have collected a fixed number n of clicks - this is the setting we choose below. If we model the stream of clicks in each wire as a Poisson process with a rate proportional to the power, then the time until we have collected exactly n clicks is a random variable. We could instead wait a fixed amount of time, in which case the total number of clicks would not be constant, and the number of clicks per wire would have a Poisson distribution[2]. We could also do something sequential or have some more complicated termination criterion. We work in the "fixed counts" setting for its tractability. In the "fixed counts" setting, we can also pretend that time is discrete and exactly one click arrives from exactly one wire per unit of time. The the large deviations results of Section 4.3 are relevant when the number of counts is large, in which case the expected time to collect them should concentrate about its expectation. We expect "fixed time" to not be too different from "fixed counts," though we have not done a careful analysis in "fixed time."

[2] The joint distribution for the number of clicks in each wire would factor into a product of Poisson pmf's for each wire; this is nice, but our results in Section 4.3 need the different kind of nice afforded by a fixed sample size.

Note that when we wait until we collect exactly n clicks, there is no need to set a total power parameter in the problem. Increasing the total power means we wait a shorter time on average to collect n photodetection events. We can thus fix n without specifying a relationship between the power in the signal and the distribution of photodetection event times, so the parameter n stands in for the total power.

Let's formalize this and set up some notation. Denote the channel input and output by X, Y, respectively. Let Δ^{k-1} denote the standard $(k-1)$-simplex of k-component probability mass functions:

$$\Delta^{k-1} = \left\{ (p_1, \ldots, p_k) : p_i \geq 0 \ \forall i, \ \sum_{i=1}^{k} p_i = 1 \right\} \tag{4.1}$$

The channel input is a distribution of the available power over the k wires, corresponding to a point on the $k-1$-simplex:

$$x = (p_1, \ldots, p_k) \in \Delta^{k-1} \tag{4.2}$$

The channel outputs are samples of size n from the input distribution. Since we assume the clicks are sampled independently, identically from the same distribution, we can summarize the output by counting the number of clicks in each wire to produce a histogram. It is convenient to rescale this histogram by dividing the click counts by n, the total number of clicks. Let Δ_n^{k-1} denote the set of pmf's with k components such that every component's denominator divides n - this is the set of types as defined in (Cover and Thomas, 2006)[3].

$$\Delta_n^{k-1} = \Delta^{k-1} \cap \left\{ \left(\frac{n_1}{n}, \ldots, \frac{n_k}{n} \right) : n_i \in \{0, 1, \ldots, n\} \forall i \right\} \tag{4.3}$$

Then the channel output corresponds to a point on the set of types:

$$y = (\hat{p}_1, \ldots, \hat{p}_k) \in \Delta_n^{k-1} \tag{4.4}$$

[3] Each element T in the set of types Δ_n^{k-1} corresponds to the set of all n independent draws from a k-component pmf that have the same histogram T; for example, the sequence 11122 and 12121 both have the type $T = (3/5, 2/5) \in \Delta_5^1$.

where $\hat{p}_i = (\#$ clicks in i-th wire$)/n$. The reader can check that the total number of possible outputs (types) is

$$|\Delta_n^{k-1}| = \left(\begin{array}{c} n+k-1 \\ k-1 \end{array} \right) \leq \min(n^k, k^n) \tag{4.5}$$

with limiting expressions

$$|\Delta_n^{k-1}| \to \begin{cases} n^{k-1}/(k-1)! & : n \to \infty \\ k^n/n! & : k \to \infty \end{cases} \tag{4.6}$$

The number of outputs (types) is polynomial in both n and k, holding k or n fixed, respectively.

A note on notation: For the reader unhappy with our choice of notation, see this footnote[4].

The output histogram is a sample of size n from the input distribution. That is, the probability of a particular output y given an input x is then given by:

$$P(y|x) = \text{MultinomialPMF}(n\hat{p}_1, \ldots, n\hat{p}_k; p_1, \ldots, p_k) \tag{4.7}$$

$$= \left(\begin{array}{c} n \\ n\hat{p}_1, \ldots, n\hat{p}_k \end{array} \right) \prod_{i=1}^{k} p_i^{n\hat{p}_i} \tag{4.8}$$

where the prefactor is a multinomial coefficient.

Figure 4.1 shows an example use of this channel for $k = 3$ wires and $n = 15$ samples. Figure 4.2 reminds readers of the geometry of a simplex as the set of probability mass functions. Figure 4.3 plots the input and output pmf's on the 2-simplex. The sample size n is varied by subplot and multiple output samples are drawn from the same input. See the caption for details.

[4] We have used the usual notation of X and Y to refer to the channel inputs and outputs, respectively. Also as is typical, a lowercase x or y refers to a value of the input or output, while the upper case refers to the random variable. Perhaps confusingly, we are using $x = (p_1, \ldots, p_k)$, $y = (\hat{p}_1, \ldots, \hat{p}_k)$. This choice emphasizes that the input and output are both probability mass functions, and that the output is a sample from the input and can be used as an estimator for the components of the input distribution (hence $\hat{p}$).

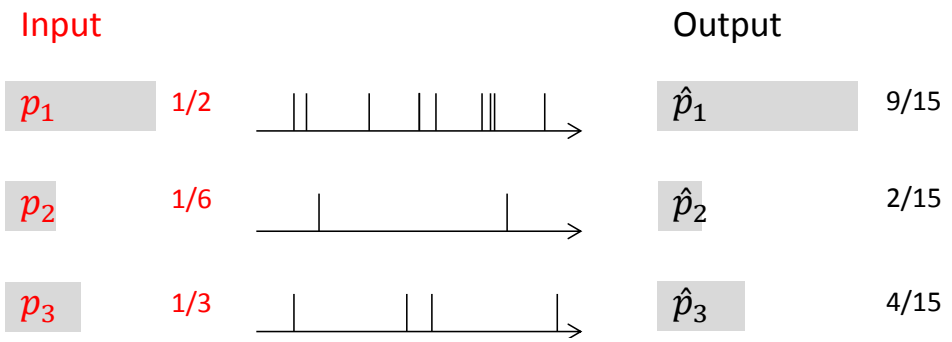

Figure 4.1: Sample use of the multinomial channel for $k = 3$ wires, $n = 15$ samples, input power distribution (red text) $x = (\frac{1}{2}, \frac{1}{6}, \frac{1}{3})$. The output histogram (black text) happens to be $y = (\frac{9}{15}, \frac{2}{15}, \frac{4}{15})$. The vertical black ticks along the wires stand for photodetection events ("clicks").

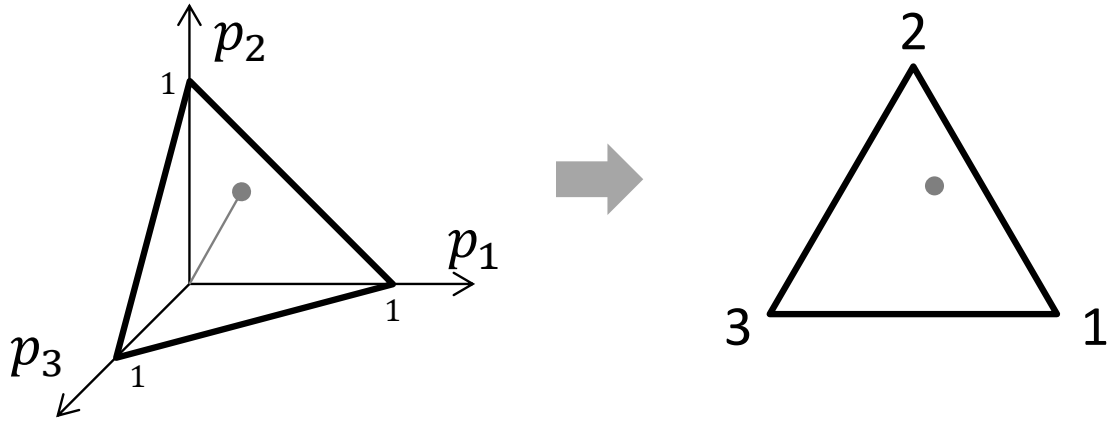

Figure 4.2: (left) The 2-simplex $\Delta^2 = \{(p_1, p_2, p_3) : \sum_i p_i = 1, p_i \geq 0 \ \forall i\})$. A point $p \in \Delta^2$ is shown in gray. (right) The 2-simplex viewed from the $(1, 1, 1)^{\mathrm{T}}$ direction. The corresponding pmf p from the left subplot is plotted in gray. This is the projection we use throughout this thesis.

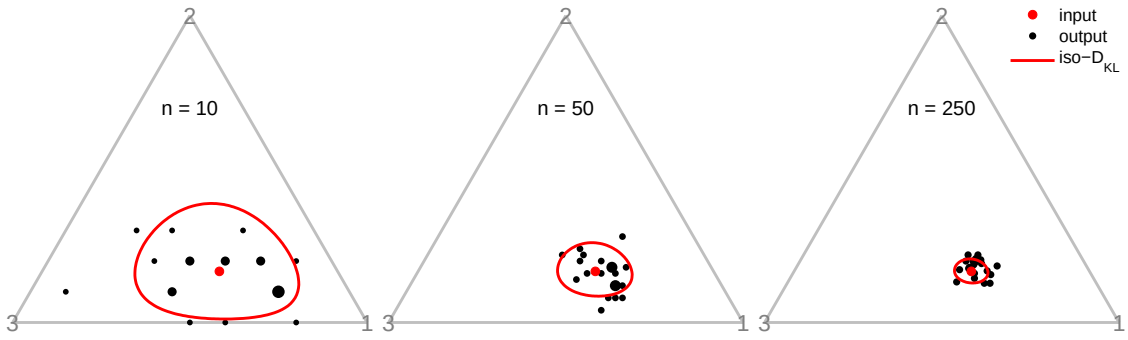

Figure 4.3: Multiple uses of the multinomial channel plotted on the 2-simplex (see Figure 4.2). In each subplot the input distribution is $x = (\frac{1}{2}, \frac{1}{6}, \frac{1}{3})$. The sample size n varies from subplot to subplot (see text in figure). We drew n samples 20 times from the input pmf in each subplot and plotted a black dot for each output pmf. Larger black dots indicate more occurrences of the same output pmf. The red contours are iso-Kullback-Leibler divergence contours: each point z along the contour satisfies $D(z||x) = 2/n$ (see Sections 4.3.2 and 4.3.5 for a discussion of why we consider these contours).

We have thus specified the "multinomial channel" and can ask two canonical questions:

- What is the capacity of this channel?
- Are there practical schemes that come close to achieving capacity, or at least do reasonably well?

The answers to these questions are roughly unknown. "Roughly" means that much work has been done in related settings, in the asymptotic $n \to \infty$ regime in the statistics literature in the work on reference priors by Bernardo and others (Bernardo, 1979; Berger et al., 2009), and in the information theory literature in the work on the minimal description length by Rissanen and others (Rissanen, 1978; Hansen and Yu, 2001). This thesis does not add to this work, but we can make some observations: We can obtain a crude upper bound on the capacity by observing that there are at most $(n+1)^k$ possible channel outputs (types), so that the capacity is at most $k \log(n+1) \sim k \log n$. Waving our hands in the air, we can say something about sample standard deviations scaling like $\sqrt{n}$ to obtain a sharper upper bound of $(k/2) \log n$; we do not provide more than this heuristic in this work.

4.2 Permutation quantization and the single shot setting

The multinomial channel turns out to be too hard for me to work with, but let's consider instead a related setting for which I propose a practical scheme and prove some large deviations-flavored optimality results.

A common way of dealing with noise is through quantization: incoming values are binned to the nearest quantized value (say, by keeping only the first few significant digits) possibly at the cost of reduced information about the inputs. Voltages in electronics are typically quantized into "0" and "1." What is a natural extension of quantization to multiple wires? A poor idea is to let only one wire of k be "high" and all the others low; this is poor because then the maximum number of outputs

grows only as $\log k$, and we can do much better. A better "idea" is to use the k wires independently for a total of 2^k possible outputs (note that if power is measured by the fraction of "1"s, then power is not conserved in this case).

A better idea in terms of output alphabet size is to rank the wires in order of increasing power carried for a total of $k! \gg 2^k$ outputs. This ranking is ill-defined if each wire carries one of only two values for the power, but we will give ourselves the freedom to load an arbitrary fraction of the total power in each wire; For example, if we wait until we receive $n = 10$ clicks, then the outputs $(\frac{3}{10}, \frac{5}{10}, \frac{2}{10})$ and $(\frac{4}{10}, \frac{6}{10}, 0)$ both correspond to the ranking $(2, 3, 1)$. Even so, the number of arriving clicks may happen to match for two wires, in which case we will have to flip a coin or resolve the ambiguity somehow else. Power conservation is potentially enforceable for this scheme, since we could make sure all possible inputs are obtained by permuting some distribution of the power in each wire.

Of course, an even better idea in terms of a larger output alphabet is to use an arbitrary quantization scheme, possibly let the quantization depend on the sample size n, or use no quantization at all. Any of the types in Δ_n^{k-1} is a possible channel output, and since the number of types is polynomially large in the sample size n for a fixed number of wires k (see (4.6)), for n growing fast enough with k (at least fast enough to resolve all $k!$ permutations) this can be many more outputs than the $k!$ permutations (which does not even depend on n). At this point we are back to the original, unquantized multinomial channel.

We could ask for the optimal quantization with respect to some criterion balancing performance in terms of error protection and complexity of the scheme[5], but we proceed with the permutation setting because it seems like an intuitive thing to try and because there is a nice answer. The resulting scheme might also be reasonably practical[6] and its analysis provides a fun example of large deviations and hypothesis testing.

Finally, we will be interested in the single shot setting rather than the channel coding setting. "Single shot" means that we will seek a scheme that performs well

[5] Thanks to Surya Ganguli for suggesting this question.

[6] e.g., there exist proposals for permutation-valued flash memory cells. See references in Section 4.5.1.

over one use of the channel, rather than allowing the sender and receiver to exploit dependence between successive uses of the channel (as in channel coding, where some received bits can protect other bits from noise by redundancy). This is a restriction over all possible communication schemes, but a scheme that works well in the single shot setting is appealing for its possible practicality: decoder and encoder complexity can be lower, since successive channel uses need not be stored. Perhaps coding across multiple shots is difficult for an optical or electronic implementation if, say, we are using a computer whose "bits" are permutations thus represented. Second, this is the setting in which I made some progress, so I present it.

The next Section formally states the problem and our results.

4.3 A permutation-resistant distribution

Let's consider the setting where instead of reliably communicating the entire pmf, we attempt to reliably communicate only the permutation corresponding to that pmf. We shall work below in the single channel use setting ("single shot").

4.3.1 Notation, definitions

Given a k-component pmf $p = (p_1, \ldots, p_k) \in \Delta^{k-1}$ and a permutation on k elements $\sigma \in S_k$, where S_k is the symmetric group on k elements, we say that σ *corresponds* to p if $p_i \leq p_j \Rightarrow \sigma(i) < \sigma(j)\ \forall i, j$, where $\sigma(i) \in \{1, \ldots, k\}$ denotes the rank of element i in permutation σ. If p is totally ordered (that is, $p_i \neq p_j\ \forall i \neq j$), then a unique permutation corresponds to p and we call this permutation σ_p. Note that in our notation more probable elements have higher rank.

We call $R_\sigma \subset \Delta^{k-1}$ the *sector corresponding to* permutation σ:

$$R_\sigma = \{\text{pmf } p \in \Delta^{k-1} : p \text{ is totally ordered and } \sigma_p = \sigma\} \tag{4.9}$$

The sectors are open sets in the subspace topology on $\Delta^{k-1} \subset \mathbb{R}^k$. The closure of R_σ in the standard topology is $\bar{R}_\sigma = \{\text{pmf } p : \sigma \text{ corresponds to } p\} \subset \Delta^{k-1}$. The $k!$

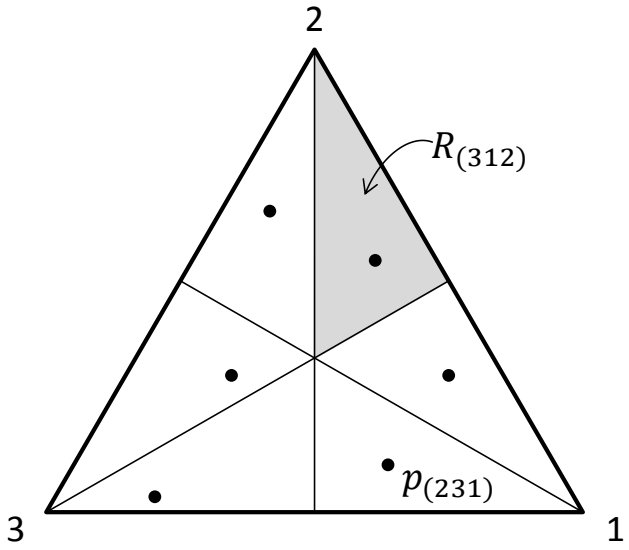

Figure 4.4: The 2-simplex Δ^2, showing the 3! sectors corresponding to each element of the symmetric group on 3 elements S_3 (see text). The sector $R_{(312)}$ is shaded. Every pmf in this sector corresponds to the permutation (312). We attempt to find 3! pmfs (black dots; $p_{(231)}$ is labeled), one per sector, to minimize error in communicating the permutations corresponding to these points (see text).

sectors are disjoint and almost partition Δ^{k-1} (the only missing points are the pmfs that are not totally ordered, which form a set of measure 0).

Figure 4.4 shows the 3! sectors of Δ^2.

4.3.2 PMF channel

The channel is the multinomial channel from Section 4.1: inputs are distributions (pmfs) on k elements, outputs are finite samples (histograms) from those distributions. Recall that n denotes the number of samples, $p \in \Delta^{k-1}$ the true (input) pmf from which the samples are drawn, and $\hat{p} \in \Delta_n^{k-1}$ the type (output; histogram) of the n samples (that is, $\hat{p}_i = (\#$ samples in i-th wire$)/n$). The channel transition probability matrix is given by the multinomial distribution pmf with parameter vector p, which

we approximate to leading exponential order in preparation for the large-deviations discussion below[7] :

$$P(\hat{p}|p) = \binom{n}{n\hat{p}_1, \ldots, n\hat{p}_k} \prod_{i=1}^{k} p_i^{n\hat{p}_i} \tag{4.10}$$

$$\doteq e^{-nD(\hat{p}||p)} \tag{4.11}$$

where $\doteq$ denotes[8] equality to leading exponential order in n and $D(\hat{p}||p)$ denotes[9] the K-L divergence between pmfs $\hat{p}$ and p.

Let $\sigma = \sigma_p$, $\hat{\sigma} = \sigma_{\hat{p}}$ denote the permutations corresponding to the channel input and output, respectively. We assume that p and $\hat{p}$ are both uniquely ordered; since $\hat{p}$ is a histogram of a finite sample, this condition can be violated, in which case our communication scheme declares an error (see Section 4.3.4 for a discussion of this case).

4.3.3 Input restrictions

Our goal is to communicate permutations by sending distributions that correspond to them into the channel. We make several simplifying restrictions. The latter two of these amount to guesses about the form of "the answer" (to the question posed in Section 4.3.5):

- As stated earlier, the channel is single-use: we must guess the permutation on the basis of a single received histogram, so that $\hat{\sigma} = f(\hat{p})$ for some function f.
- There is only one possible channel input pmf per permutation $\sigma \in S_n$. Call this input p_σ.

[7] Note that in this Section we forgo the usual notation of x and y denoting the channel input and output, respectively, in favor of p and $\hat{p}$ to emphasize the input as a pmf and the output as a sample from the input. We will "remember" that the pmf's are used in a communication scheme later.

[8] $a_m \doteq_m b_m \Leftrightarrow \lim_{m\to\infty} \frac{1}{m} \log(a_m/b_m) = 0$ denotes equality to leading exponential order.

[9] $D(q||p) = \sum_i q_i \log(q_i/p_i)$, setting $0 \log 0 = 0 \log(0/0) = 0$.

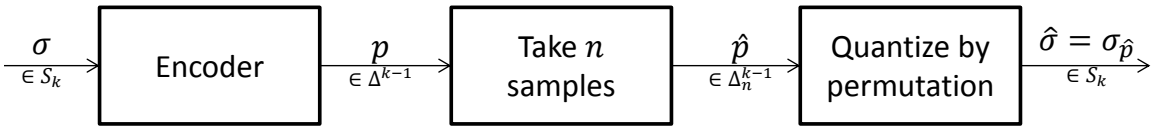

Figure 4.5: A channel between permutations, where an encoder maps permutations to pmfs, samples are drawn from the pmfs, and the output permutation is obtained by quantizing (sorting) the sample histogram.

- The input pmfs p_σ are all permutations of each other (abusing notation):

$$\sigma = \sigma'' \circ \sigma' \Rightarrow p_\sigma = p_{\sigma'' \circ \sigma'} = \sigma''(p_{\sigma'})$$

The third restriction implies that the maximum likelihood decoder sets $\hat{\sigma} = \sigma_{\hat{p}}$. We may want to assume a uniform prior on the permutations, in which case the maximum *a posteriori* (MAP) decision rule for guessing the permutation is $\hat{\sigma} = \sigma_{\hat{p}}$. We will be interested in the large deviation properties of our communication scheme, where assumptions about the prior on permutations will not impact our final results.

4.3.4 Induced permutation channel

The above restrictions thus induce a channel where both inputs and outputs are permutations, as shown in Figure 4.5. The channel consists of an encoder $p(\sigma)$ that maps permutations to pmfs to which they correspond: $p(\sigma)$ satisfies $\sigma_{p(\sigma)} = \sigma$. The encoder is followed by drawing a sample of size n from the pmf, followed by quantization of the sample histogram (i.e., sorting) to produce the output permutation (ties in the sample histogram are resolved arbitrarily). Figure 4.6 plots the sampling and quantization step on a 2-simplex. The sample size n is varied by subplot and multiple output samples are drawn from the same input. See the caption for details.

The channel transition matrix for this channel is (treating σ and σ' as the input and output permutations, respectively):

$$P(\sigma'|\sigma) = \sum_{\hat{p} \in R_{\sigma'} \cap \Delta_n^{k-1}} P(\hat{p}|p(\sigma)) \tag{4.12}$$

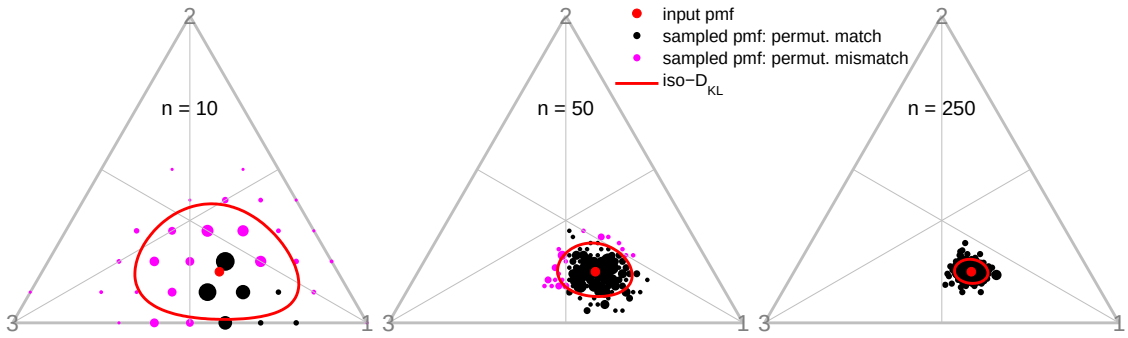

Figure 4.6: Multiple uses of the permutation channel plotted on the 2-simplex (see Figure 4.2). In each subplot the input permutation is $\sigma = (231)$, which the encoder chooses to map to the pmf $p = (\frac{1}{2}, \frac{1}{6}, \frac{1}{3})$. The sample size n varies from subplot to subplot (see text in figure). We drew n samples 200 times from the input pmf in each subplot and plotted a dot for each sampled pmf (histogram). Larger dots indicate more occurrences of the same sampled pmf. Magenta dots indicate mismatching input and output permutations (or if there is an ambiguity for cases when the sampled pmf is on the boundary of two sectors). Black dots indicate matching input and output permutations. The red contours are iso-Kullback-Leibler divergence contours as in Figure 4.3.

where $P(\hat{p}|p(\sigma))$ is the multinomial pmf (4.11), $R_{\sigma'}$ is the sector corresponding to permutation σ' (see Section 4.3.1), and Δ_n^{k-1} is the set of types with k components with denominator n (see Section 4.1). This is an incomplete characterization of the channel matrix pending a resolution of the case that a sampled pmf is on the boundary of two sectors[10]. We can treat this case as a separate "boundary error" output[11], call it E, in which case

$$P(E|\sigma) = 1 - \sum_{\sigma' \in S_k} P(\sigma'|\sigma) \tag{4.13}$$

We shall be interested in the large sample size n limit, in which case the probability of "boundary error" turns out to be (see next Sections) the dominant source of error. (4.12) and (4.13) are complicated expressions to use in computations, but we shall only work out the large sample size n limits and large deviations results below, which don't require us to compute them exactly.

Once we identify an encoder function from permutations to pmfs, we can treat this permutation channel as a symmetric discrete memoryless channel (DMC) and obtain its capacity[12]. The meat of this work is in identifying the "best" encoder function in terms of minimizing the associated DMC crossover probabilities (4.12) for the single-shot communication scheme setting.

4.3.5 A permutation-resistant distribution

The restrictions above lead us to state an optimization problem: find $p^* \in \Delta^{n-1}$ that minimizes the probability of error (guessing an incorrect permutation)

[10] This happens when two or more components of the sampled pmf (histogram) $\hat{p}$ have the same value, so that multiple permutations correspond to $\hat{p}$; e.g. both permutations $(1,2,3)$ and $(2,1,3)$ correspond to $\hat{p} = (\frac{1}{10}, \frac{1}{10}, \frac{8}{10})$.

[11] We could be less conservative and flip a coin to resolve the ambiguity, but this won't make our communication scheme much more reliable.

[12] The capacity achieving distribution is uniform on the $k!$ input permutations by symmery. The capacity is then $C = (1-P(E))(\log k! - H(P(\cdot|\mathbf{1})))$, where $P(E)$ is the "boundary error" case treated as an erasure and $P(\cdot|\mathbf{1})$ is a row of the channel transition matrix (here the row corresponding to $\mathbf{1}$, the identity permutation).

$$P_{\mathrm{e}} \equiv P(\text{input and output permutations don't match}) \tag{4.14}$$
$$= P\left(\hat{\sigma} \neq \sigma_{p^*}\right) \tag{4.15}$$

where we are using the maximum likelihood rule $\hat{\sigma} = \sigma_{\hat{p}}$.

Since the input pmf's are restricted to be permutations of each other, let's assume without loss of generality that p^* is sorted:

$$i < j \Rightarrow p_i^* < p_j^* \tag{4.16}$$

We can compute the error probability exactly, assuming a uniform prior on input permutations and choosing an arbitrary $\sigma \in S_{k-1}$:

$$P_{\mathrm{e}} = \sum_{\sigma' \neq \sigma} P(\sigma'|\sigma) + P(E|\sigma) \tag{4.17}$$

where the terms can be evaluated using (4.12) and (4.13). This is something complicated, and this is the point where results from large deviations theory come to our aid.

In the large sample size n limit, the probability of error is dominated by the case when two adjacent bins in the received histogram $\hat{p}$ are out of order or equal (recall that we assumed that p^* is sorted. The probability of multiple errors is exponentially smaller than a single one). Call such events E_i for $i \in \{1, \ldots, n-1\}$:

$$E_i = \{\hat{p} \in \Delta^{k-1} : \hat{p}_i \geq \hat{p}_{i+1}\} \tag{4.18}$$

Note that we let $\hat{p}$ take on values in the simplex Δ^{k-1}, rather than the set of types Δ_n^{k-1}, since we will apply the Conditional Limit Theorem where the limit is $n \to \infty$.

Now applying Sanov's theorem (Sanov, 1958) in the notation of (Cover and Thomas, 2006) we can compute their probabilities to first order in the exponent:

$$P(E_i) \doteq e^{-nD(p^{E_i}||p^*)} \tag{4.19}$$

where p^{E_i} denotes the I-projection onto the events E_i:

$$p^{E_i} \equiv \arg\min_{\hat{p} \in E_i} D(\hat{p}||p^*) \tag{4.20}$$

p^{E_i} is the conditional limit distribution of i for the conditional limit theorem (see (Cover and Thomas, 2006)):

$$P(i|\hat{p} \in E_i) \to p_i^{E_i} \text{ as } n \to \infty \tag{4.21}$$

so if we happen to observe an error of type E_i, the sampled pmf $\hat{p}$ is close to p^{E_i}. The intuition for this is that $P(E_i)$ is obtained by summing over all types in E_i weighted by their probabilities. There are only polynomially many types (4.6) and the probability of each is exponentially small in n (4.11), so the single largest term dominates the sum, and the sum is equal to it to leading exponential order.

The error probability and error exponent are

$$P_\text{e} \doteq \max_i P(E_i) \doteq \max_i e^{-nD(p^{E_i}||p^*)} \tag{4.22}$$

$$-\frac{1}{n} \log P_\text{e} \to \min_i D(p^{E_i}||p^*) \tag{4.23}$$

In our setup, we can guess the p^{E_i}: our channel communication setup is equivalent to a hypothesis-testing setup, where we are deciding between hypothesis p^* and the hypothesis formed by swapping adjacent entries of p^*, denoted for notational convenience as $\sigma^i(p^*)$ for $1 \leq i \leq k-1$:

$$\sigma^i(p_j^*) = p^*_{\sigma^i(j)} \tag{4.24}$$

where σ^i is the permutation that swaps elements i and $i+1$ and fixes all other elements:

$$\sigma^i(j) = \begin{cases} i+1 & : j = i \\ i & : j = i+1 \\ j & : \text{otherwise} \end{cases} \tag{4.25}$$

Now the conditional limit distribution is given by (see, e.g., (Cover and Thomas, 2006) section on hypothesis testing, (11.7), equation (11.200)):

$$p^{E_i} \sim (\sigma^i(p^*))^\lambda \odot (p^*)^{1-\lambda} \tag{4.26}$$

for some λ, where the exponentiation is done point-wise, $\odot$ denotes the Schur (point-wise) product of two vectors, and $\sim$ denotes equality of vectors up to normalization. To minimize the error exponent over the choice of $k!$ input distributions, we set $\lambda = 1/2$, so that

$$p^{E_i} = \frac{1}{Z_i}\sqrt{\sigma^i(p^*) \odot p^*} \tag{4.27}$$

where Z_i is a normalization constant.

We can now state our optimization problem for p^*:

$$p^* = \arg\max_{p \in R_{\mathbf{1}}} \min_{i \in \{1,\ldots,k-1\}} D(p^{E_i}||p^*) \tag{4.28}$$

where $R_{\mathbf{1}}$ denotes the sector corresponding to the identity permutation — this imposes the constraint that p^* must be sorted. Figure 4.7 shows this setup on the 2-simplex.

4.3.6 Solving for p^*: some observations

For a sector R_σ of Δ^{n-1} (defined in (4.9)), we denote the *exterior face* of R_σ by

$$\mathrm{EF}(R_\sigma) = \bar{R}_\sigma \cap \{p : p(\sigma(1)) = 0\} \tag{4.29}$$

where $\bar{R}_\sigma$ is the topological closure of R_σ (see Section 4.3.1). Since $\sigma(1)$ corresponds to the element with lowest rank, $\mathrm{EF}(R_\sigma)$ is the set of all pmfs to which σ corresponds and whose smallest entry is 0. $\mathrm{EF}(R_\sigma)$ is homeomorphic to Δ^{k-2}. Abusing our notation, $\mathrm{EF}(\mathrm{EF}(R_\sigma))$ is the set of all pmfs to which σ corresponds and whose two smallest entries are 0. $\mathrm{EF}^{(k-1)}(R_\sigma)$ is homeomorphic to a point — in Δ^{k-1}, this point is $p_i = \delta_{i,\sigma(k)}$. $\mathrm{EF}^{(k)}(R_\sigma) = \emptyset$.

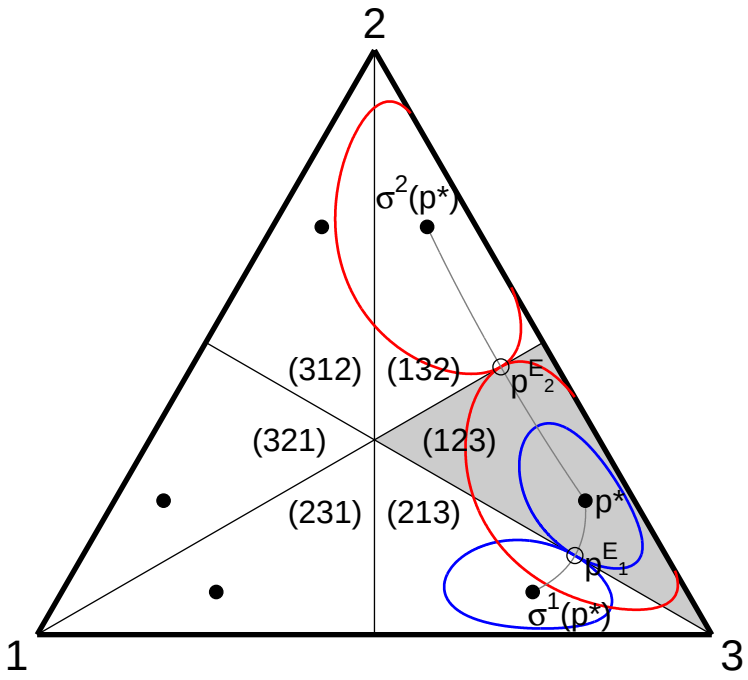

Figure 4.7: The setup for our optimization problem on the 2-simplex. Sectors are labeled by the permutation they correspond to. We seek p^* that maximizes the KL divergence $D(p^{E_i}||p^*)$ to the nearest sector (in this case $R_{(213)}$ or $R_{(132)}$) neighboring the sorted sector $R_{(123)} = R_{\mathbf{1}}$ (gray fill). (gray lines) geodesics: $\sim (\sigma^i(p^*))^{\lambda} \odot (p^*)^{1-\lambda}$, with $\lambda \in [0, 1]$. (open circles) the conditional limit distributions p^{E_i} corresponding to a swap of entries i and $i+1$ in the permutation that corresponds to the histogram channel output, obtained by setting $\lambda = 1/2$ along the geodesics. Note that we relabeled vertices 1 and 3 relative to Figure 4.4.

- We expect $p_1^* = 0$, so that $p^* \in \mathrm{EF}(R_{\mathbf{1}})$. Recall that we require p^* to be sorted, so that $\sigma(i) = i$, so $\sigma(1)$ has lowest rank and p_1^* is the smallest component of p^*. We can always reduce the probability of error by taking any probability mass in the smallest (first) bin and putting it in, say, the biggest (last) bin.

- We expect $p^* \notin \mathrm{EF}(\mathrm{EF}(S_{\mathbf{1}}))$. Otherwise, $p_i^* = 0$ for some $i > 1$. This violates our requirement that p^* is uniquely ordered.

- We expect p^* to satisfy:

$$D_i \equiv D(p^{E_i}||p^*) = c \; \forall i \tag{4.30}$$

  for some constant c. Otherwise, suppose that $D_i = D(p^{E_i}||p^*)$ is the min of $\{D_m\}_{m\in\{1,\ldots,k\}}$, D_j is the next-min, and $D_j > D_i$. Since the divergences are continuous functions of p^*, we can always decrease the probability of error by increasing D_i (and do this without decreasing D_j below the value of D_i) by setting $p_{i+1}^{**} = p_{i+1}^* + \epsilon$, $p_{j+1}^{**} = p_{j+1}^* - \epsilon$ for some small enough $|\epsilon|$, contradicting the optimality of p^*. This argument would fail if $p_{i+1}^* = 0$ for some $i \in \{1, \ldots, k-1\}$, but from the above remark $p^* \notin \mathrm{EF}(\mathrm{EF}(S_{\mathbf{1}}))$.

It might be natural to have a positive floor on the min value of p^*. We shall see that our solution for p^* below extends to this case, and we will use this case to deal with a version of our permutation coding setup that has an extra noise input in Section 4.4.

4.3.7 Solving for p^*: intuitive argument

The number of counts in the i-th bin of the received histogram for n samples drawn from p^* is binomially-distributed:

$$n\hat{p}_i \sim \mathrm{Bino}\,(n, p_i^*) \tag{4.31}$$

By the above observations, we want to have $D_i = D_{i+1} \forall i \leq k-1$, so we want p^* to satisfy approximately:

$$\mathbb{E}[n\hat{p}_i] + \sqrt{\text{Var}[n\hat{p}_i]} = \mathbb{E}[n\hat{p}_{i+1}] - \sqrt{\text{Var}[n\hat{p}_{i+1}]} \; \forall i \leq k-1 \tag{4.32}$$

so that the fraction of probability mass overlapping between adjacent bins is approximately equal for all bins.

For the binomial distribution, $\mathbb{E}[n\hat{p}_i] = np_i^*$ and $\text{Var}[n\hat{p}_i] = np_i^*(1-p_i^*) \approx np_i^*$ for $p_i^* \ll 1$ (we expect most values of p_i^* to be in this regime for large k since otherwise the probabilities couldn't add up to 1). Substituting these values into (4.32) we obtain the recurrence relation:

$$p_i^* + \sqrt{\frac{p_i^*}{n}} = p_{i+1}^* - \sqrt{\frac{p_{i+1}^*}{n}} \tag{4.33}$$

Solving the recurrence relation (4.33) for p_i^* we find:

$$p_i^* \sim (i+\kappa)^2 \tag{4.34}$$

where κ is a free parameter determined by p_1^*. By the first observation in Section 4.3.6, $p_1^* = 0$, so we guess that $\kappa = -1$ and

$$p_i^* = \frac{1}{Z}(i-1)^2 \text{ for } i \in \{1, \ldots, k\} \tag{4.35}$$

where $Z = \sum_{i=1}^{k}(i-1)^2$. It turns out that this guess is exactly right.

4.3.8 Solving for p^*: exact solution

Problem statement

Let's solve the optimization problem (4.28) for p^*. Recall that we have defined

$$p^{E_i} = \frac{1}{Z_i}\sqrt{\sigma^i(p^*) \odot p^*} \text{ with } Z_i = \sum_{i=1}^{k}\sqrt{\sigma^i(p_i^*)\; p_i^*} \tag{4.36}$$

and

$$D_i = D(p^{E_i}||p^*) \tag{4.37}$$

where $\sigma^i(p^*)$ is obtained by swapping p_i^* and p_{i+1}^*. We are seeking p^* that satisfies

$$D_i = D_j = c \; \forall i, j \tag{4.38}$$

and is uniquely ordered:

$$p_i^* < p_{i+1}^* \; \forall i \tag{4.39}$$

and maximizes $c = D_i \; \forall i$.

Computation

See Appendix 4.A.

Results

Let's summarize what we have found. The solution to the optimization problem (4.28) for any number of wires k — the permutation-resistant pmf — is (up to permutation of the entries):

$$p_i^* = \frac{1}{Z_\kappa}(i + \kappa)^2 \tag{4.40}$$

for $i \in \{1, \ldots, k\}$, where $\kappa \geq -1$ is fixed by our choice for $p_1^* \in [0, 1/k)$ and Z_κ is a normalization constant computed in (4.91). The optimal choice of κ in terms of error probability is $\kappa = -1$, in which case

$$p_i^* = \frac{1}{Z}(i - 1)^2 \tag{4.41}$$

for $i \in \{1, \ldots, k\}$, where $Z = Z_{\kappa=-1}$ computed in (4.93).

Thus power (pmf mass) scales as the square of the rank of the pmf's component. Note that for the $k = 2$ case, the solution says that we should put all of the power in one wire and none in the other; this makes sense. For the $k = 3$ case, the solution is $p^* = (0, \frac{1}{5}, \frac{4}{5})$. As $\kappa \to \infty$, p^* approaches the uniform distribution and the probability of error approaches 1 (see below).

Figure 4.8 (top) shows the solution p^* for several values of κ (corresponding to several value of $p_{\min}$). We see that both neighboring sectors do indeed look to be the same distance in KL divergence away from p^*.

To leading exponential order in the sample size n the probability of error is

$$P_{\mathrm{e}} \doteq e^{-nD_i} = e^{-nc} \tag{4.42}$$

where D_i is the error exponent (rate function for incorrectly inferring a permutation that swaps ranks i and $i+1$ of the channel input permutation), independent of i, computed in (4.98). The probability of error is then obtained by substituting (4.98) into (4.42):

$$\begin{aligned} P_{\mathrm{e}} &\doteq e^{-nD_i} = e^{-nc} && (4.43)\\ &= \left(1 - \frac{1}{Z_\kappa}\right)^n && (4.44)\\ &= \left(1 - \frac{6}{2k^3 + 3(2\kappa+1)k^2 + (6\kappa^2+6\kappa+1)k}\right)^n && (4.45)\\ &\to e^{-3\frac{n}{k^3}} \text{ as } k \to \infty, n \to \infty, n/k^3 \to \infty && (4.46) \end{aligned}$$

Thus to maintain a constant probability of error to leading exponential order as the number of bins/wires k grows, the sample size n must grow as the cube of k. Figure 4.8 (bottom) shows the error exponent using both the exact and approximate expressions (4.98), (4.99), respectively.

Since there are multiple possible errors ($k-1$ errors, corresponding to two adjacent ranks flipping in the received distribution), we should multiply P_{e} above by $k-1$, and the sample size necessary to maintain a constant probability of error should become $(k/3)\log(k-1)$, but this polynomial prefactor for P_{e} does not change the result to leading exponential order.

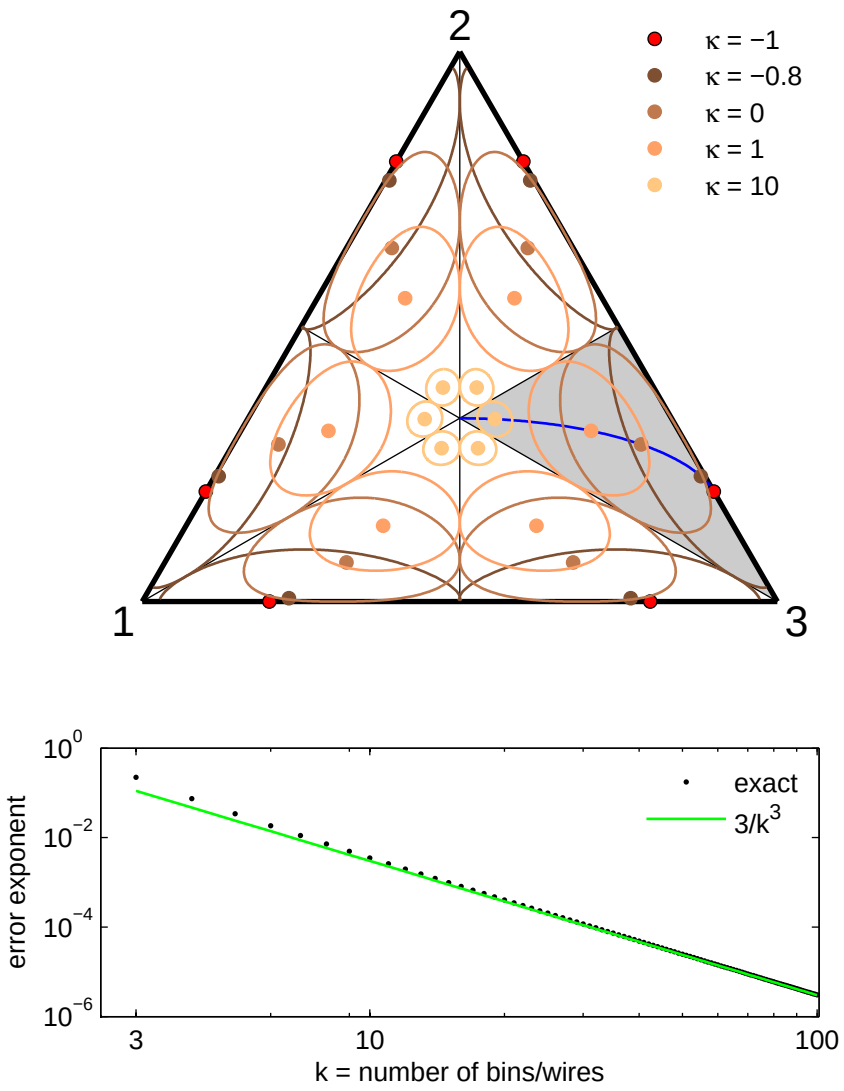

Figure 4.8: (top) The solution to our permutation-resistance optimization problem on the 2-simplex. The 2-simplex is divided into sectors corresponding to the 3! permutations. The sector $R_{(123)}$ is shaded. The solid dots correspond to p^* for varying values of κ (see legend). Iso-Kullback-Leibler divergence contours are plotted in same colors. The blue curve is the set of pmf's p^* for $\kappa \in [-1, \infty)$. (bottom) The error exponent (rate function for incorrectly guessing the input permutation) using both the exact and approximate expressions (4.98), (4.99), respectively.

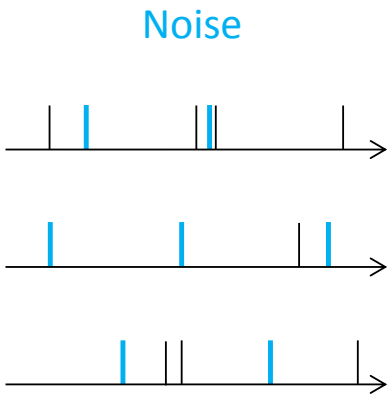

Figure 4.9: Sample use of the extra noisy multinomial channel with $k = 3$ wires and a uniform p_{noise} extra noise pmf. The "dark counts" are colored in blue, but the observer can't distinguish them from the other counts.

The conditional limit distribution (sampled pmf $\hat{p}$) conditioned on error E_i occurring is obtained by taking the geometric mean of entries p_i^* and p_{i+1}^*:

$$p_j^{E_i} = \frac{1}{Z_i} \begin{cases} (i+\kappa)(i+\kappa+1) & : j \in \{i, i+1\} \\ (i+\kappa)^2 & : \text{otherwise} \end{cases} \tag{4.47}$$

where it turns out that $Z_i = e^{1/D_i}$ (computed in (4.88)).

4.4 Permutation coding: the (extra) noisy case

Next we consider a generalization of the setup in the previous Section to include a natural-looking source of noise. Suppose that the output $\hat{p} \in \Delta_n^{k-1}$ of the multinomial channel (Section 4.1) is no longer formed by taking n samples from the true input pmf $p \in \Delta^{k-1}$, but is instead sampled from a noise-corrupted version of p, where the noise adds some probability mass to each wire regardless of the input power to that wire. We think of some of the incoming "clicks" as drawn from the true input pmf p, and others drawn from "noise," and the observer is unable to distinguish between these two kinds of clicks. This happens in photodetection setups, where the extra clicks are called "dark counts." Figure 4.9 shows this setup for $k = 3$ wires.

We model this noise by additively mixing the true input pmf p with a noise pmf p_{noise}:

$$p_\alpha \equiv \alpha p_{\text{noise}} + \bar{\alpha} p \tag{4.48}$$

where $\alpha \in [0, 1]$, $\bar{\alpha} \equiv 1 - \alpha$, so larger α means more noise. I work out results below for the uniform noise case and discuss later how to possibly extend the results to non-uniform noise. In the uniform noise case:

$$p_\alpha = \alpha \frac{1}{k} + \bar{\alpha} p \tag{4.49}$$

There are now two sources of noise in this communication setup (hence the "extra-"noisy): 1) the "counting" noise due to drawing a finite sample from p to form $\hat{p}$; this is the noise we tried to deal with by quantizing $\hat{p}$ in the previous Section. 2) the extra "dark count" noise that we have just introduced. If we wanted to, we could write down a channel transition matrix as in Section 4.1 for this extra noisy channel in terms of the noise strength α, ask for this channel's capacity, and ask for practical schemes that come close to achieving it. We didn't have much to say on these questions previously for the non-extra-noisy case, and we don't have much to add now. We do, however, extend the permutation quantization scheme of the previous Section to the extra noisy setting.

The extra noisy setting is shown on the 2-simplex Δ^2 in Figure 4.10. The extra noise pulls every pmf towards the middle (uniform distribution). In particular, the image of the boundary of the simplex under the noise map (4.49) is a smaller, concentric triangle (blue dashed triangle). Stronger noise maps pmfs closer to the uniform distribution. The image of the boundary of the simplex (blue dashed triangle) under (4.49) shows the feasible set of noise-corrupted input pmfs - points outside this set do not have a preimage (input pmf) under (4.49) that is a valid pmf (some entries are negative).

This picture suggests a way of adapting our permutation quantization scheme to the noise. The set of solutions p^* (4.40) from the previous Section is plotted for

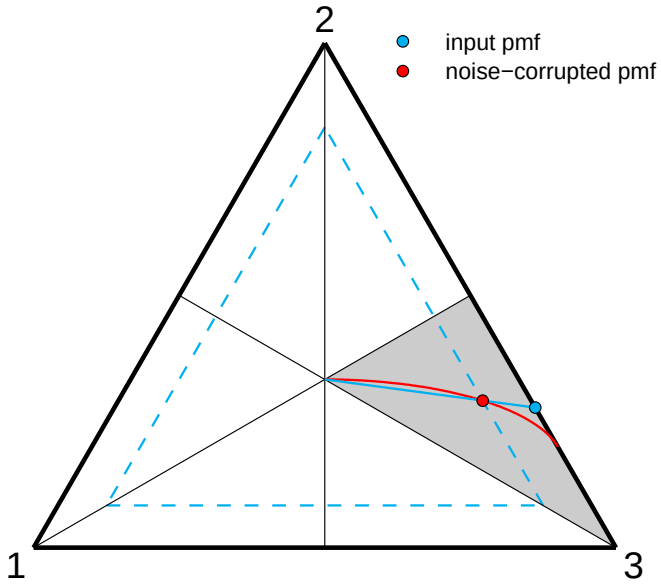

Figure 4.10: The extra noisy setting shown on the 2-simplex ($k = 3$ wires) for noise magnitude $\alpha = 0.25$ (4.49). The 2-simplex is divided into sectors corresponding to the 3! permutations. The sector $R_{(123)}$ is shaded. The dashed blue line is the image of the boundary of the 2-simplex under the noise map (4.49). The red curve is the set $\mathcal{K}$ (4.50): pmfs p^* equidistant from neighboring sectors in KL divergence (and are thus maximally permutation-resistant), indexed by parameter $\kappa \in [-1, \infty)$; see Figure 4.8 for another look. The red point is the intersection of the red curve and dashed blue line - this is the noise-corrupted input pmf. The blue dot is the back-extension of the red dot to the simplex boundary - this is the optimal input pmf for this noise level α with respect to the criterion of minimizing the probability of guessing the wrong output permutation.

parameter $\kappa \in [-1, \infty)$. Let's denote this set by $\mathcal{K}$ (again up to permutation of p^*):

$$\mathcal{K} \equiv \left\{ \text{pmf } p^* : p_i^* = \frac{1}{Z_\kappa}(i+\kappa)^2 \right\}_{\kappa \in [-1,\infty)} \tag{4.50}$$

where Z_κ is a normalization constant (computed in (4.91)). Since $p_i^* \sim (i+\kappa)^2$, the set $\mathcal{K}$ is not convex and does not contain its image under the noise map (4.49). Thus, if a pmf $p \in \mathcal{K}$, then after corruption with uniform noise, $p_\alpha = \alpha\frac{1}{k} + \bar{\alpha}p \notin \mathcal{K}$. We would like $p_\alpha \in \mathcal{K}$, however, so that p_α possesses the permutation-resistance property of being equidistant in KL divergence from neighboring sectors of the simplex.

We can achieve this by choosing a pmf p such that after corruption by noise it lands on the set of permutation-resistant pmfs $\mathcal{K}$. This is shown in Figure 4.10: we find the point $p_\alpha \in \mathcal{K}$ that is farthest from the uniform distribution (to minimize permutation error probability) and is still in the feasible set (noise-corrupted pmfs, bounded by the blue dashed triangle in Figure 4.10. p_α is the red point.). p_α will be the noise-corrupted input pmf (4.49). To find the input pmf p, we then back-extend p_α to the boundary of the simplex (blue line in Figure 4.10; p is the blue point).

Here is the symbolic version of this graphical construction: the input pmf $p^{*(\alpha)}$ is given by inverting the noise map (4.49)

$$p_i^{*(\alpha)} = \frac{1}{\bar{\alpha}}\left(\frac{1}{Z_{\kappa_\alpha}}(i+\kappa_\alpha)^2 - \alpha\frac{1}{k}\right) \tag{4.51}$$

for $i \in \{1, \ldots, k\}$, where with some work we can show that

$$\kappa_\alpha = \frac{1}{6(1-\alpha)}\left(3(k+1)\alpha - 6 + \sqrt{3\alpha(k-1)(k(4-\alpha)-\alpha-2)}\right) \tag{4.52}$$

and Z_{κ_α} is computed in (4.91):

$$Z_{\kappa_\alpha} = \sum_{i=1}^{k}(i+\kappa_\alpha)^2 = \frac{1}{6}\left(2k^3 + 3(2\kappa_\alpha+1)k^2 + (6\kappa_\alpha^2+6\kappa_\alpha+1)k\right) \tag{4.53}$$

As in the previous Section, we compute the error exponent (rate function for incorrectly guessing the input permutation):

$$P_{\mathrm{e}} \doteq e^{-nc} \tag{4.54}$$

$$= \left(1 - \frac{1}{Z_{\kappa_\alpha}}\right)^n \tag{4.55}$$

where P_{e} is the probability of guessing an incorrect input permutation.

These expressions are hard to look at, so let's compute some asymptotic expressions.

Low noise limit

First, for the low noise limit $\alpha \to 0$, we can see from (4.51) that $p^{*(\alpha)} \to p^*$ as $\alpha \to 0$, which makes sense. Let's expand the expression for $p^{*(\alpha)}$ (4.51) in powers of α:

$$p_i^{*(\alpha)} = p_i^{*(\alpha=0)} + \left(\frac{2\sqrt{6}(i-1)(2k-3i+2)}{k(2k-1)^{3/2}(k-1)^{1/2}}\right)\alpha^{1/2} + \frac{i^2}{k^3}\Theta\left(\alpha^{3/2}\right) \tag{4.56}$$

$$= \frac{6(i-1)^2}{k(2k-1)(k-1)} + \frac{i^2}{k^3}\Theta\left(\alpha^{1/2}\right) \tag{4.57}$$

where the second line is a cruder approximation than the first, but makes it easier to see that the components of $p^{*(\alpha)}$ shift linearly in $\sqrt{\alpha}$.

Let's compute the error exponent c, where $P_{\mathrm{e}} \doteq e^{-nc}$, for the weak noise $\alpha \to 0$ regime using (4.55, 4.53, and 4.52):

$$c = -\frac{1}{n}\log\left(1 - \frac{1}{Z_{\kappa_\alpha}}\right) \tag{4.58}$$

$$= -\frac{1}{n}\log\left(1 - \frac{6}{k(2k-1)(k-1)} + \left(\frac{6\sqrt{6}}{k(2k-1)^{3/2}(k-1)^{1/2}}\right)\alpha^{1/2} + \frac{1}{k^3}\Theta\left(\alpha\right)\right) \tag{4.59}$$

$$= -\frac{1}{n}\log\left(1 - \frac{3}{k^3}\left(1 - \sqrt{3\alpha}\right) + \Theta\left(\frac{1}{k^4}\right) + \sqrt{\alpha}\,\Theta\left(\frac{1}{k^4}\right) + \frac{1}{k^3}\Theta\left(\alpha\right)\right) \tag{4.60}$$

so in the appropriate limits, the error exponent degrades by a factor of $(1 - \sqrt{3\alpha})$ relative to the noiseless case (4.46) and the error probability is given by

$$P_{\mathrm{e}} \doteq e^{-3\frac{n}{k^3}\left(1-\sqrt{3\alpha}\right)} \tag{4.61}$$

Thus the sample size to maintain the same error probability as for the noiseless case must grow by a factor of $(1-\sqrt{3\alpha})^{-1}$. For noise strength $\alpha = 0.01$, this is about 21% more samples per channel use.

Large noise limit

What happens in the large noise limit $\alpha \to 1$, $\bar{\alpha} \to 0$? Let's expand the expression for $p^{*(\alpha)}$ (4.51) in powers of $\bar{\alpha}$:

$$p_i^{*(\alpha)} = \frac{2(i-1)}{k(k-1)}\left(1 + \frac{(3i - 2(k+1))}{6(k-1)}\bar{\alpha} + \frac{i}{k}\Theta\left(\bar{\alpha}^2\right)\right) \tag{4.62}$$

$$= \frac{2(i-1)}{k(k-1)} + \frac{i^2}{k^3}\Theta\left(\bar{\alpha}\right) \tag{4.63}$$

where the second line is a cruder approximation than the first, but makes it easier to see that in the high noise regime $\bar{\alpha} \to 1$, the power (pmf mass) scales linearly with the rank of the pmf's component, rather than quadratically as for the no-extra-noise $\alpha = 0$ regime (4.40).

Let's compute the error exponent c, where $P_{\mathrm{e}} \doteq e^{-nc}$, for the large noise $\bar{\alpha} \to 1$ regime using (4.55, 4.53, and 4.52):

$$c = -\frac{1}{n}\log\left(1 - \frac{1}{Z_{\kappa_\alpha}}\right) \tag{4.64}$$

$$= -\frac{1}{n}\log\left(1 - \frac{\bar{\alpha}^2}{(k-1)^2 k} + \frac{1}{k^3}\Theta\left(\bar{\alpha}^3\right)\right) \tag{4.65}$$

so in the appropriate limits, the error exponent grows as $\bar{\alpha}^2$ and the error probability is given by

$$P_{\mathrm{e}} \doteq e^{-\frac{n}{k^3}\bar{\alpha}^2} \tag{4.66}$$

In the large noise regime, doubling the strengh of the input pmf reduces the sample size we need to maintain a given probability of error by a factor of 4.

Non-uniform noise

Suppose that our input pmf is corrupted by non-uniform noise: $p_\alpha = \alpha p_{\text{noise}} + \bar{\alpha} p$, where p_{noise} is non-uniform. Then the image of this noise map applied to the simplex boundary (the blue dashed triangle on Figure 4.10) is no longer a rescaled unit simplex, but is linearly distorted (on Figure 4.10, the blue dashed triangle would no longer be equilateral). We can apply the same recipe as above to find the farthest-from-uniform pmf in $\mathcal{K}$ in the image of the noise map, but we would now have different error exponents for different input permutations (i.e., a transition matrix $P(\sigma'|\sigma)$ for our permutation DMC whose rows are not permutations of each other). We might therefore want to put a non-uniform prior on input distributions for our permutation DMC. We might for the non-uniform noise case abandon permutation quantization in favor of a more general quantization scheme.

4.5 Permutation coding and error correction

So far we have looked at the single shot setting and attempted to minimize the probability of incorrectly inferring the input permutation. We worked in a large sample size limit, so that the probability of an error was small and dominated by the single most likely incorrect permutation (which involved swapping two adjacent ranks).

Let's try to generalize this a bit to the case where the sample size is not so large that errors are improbable. Perhaps the sample size is small enough that at least some ranks are guessed incorrectly most of the time. Perhaps the sample size n is even on the order of k, the number of wires/bins (or at least smaller than quadratic in k^2), so that it is impossible for the output pmf (a histogram) to be uniquely ordered and thus to correspond to a unique permutation (see Section 4.3.1 for definitions of these terms).

4.5.1 Some background

A common approach is to use error correction. The idea of using permutations to communicate over a Gaussian channel is due to (Slepian, 1965), wherein the single symbol channel inputs are restricted to a certain set of values and the set of codewords is obtained by permuting some initial codeword, and a procedure was given for finding numerically optimizing some code parameters to minimize error probabilities. Further work on using subsets of the symmetric group S_k as the codebook is due to (Blake, 1974; Blake et al., 1979) and on using subgroups as codebooks by (Chadwick and Kurz, 1969; Bailey, 2009). Related recent work on "rank modulation" for possible enhancement of flash memory storage capacity is due to (Jiang et al., 2008, 2009; Barg and Mazumdar, 2010; Wang and Bruck, 2010; Wang, 2013).

Our motivation for studying communication with permutations in this Chapter is that this setting arose reasonably naturally as a quantized version of communication by sampling from input distributions (see Section 4.1 and Section 4.2). I do provide below a few sketches of ideas for error-correcting permutation codes that as far as I know are new (Section 4.5.3.1), but this Chapter's contribution is mainly in identifying the encoding and quantization strategy for a permutation channel mediated by a multinomial sampling step (see Figure 4.5). Aside from the early work by (Slepian, 1965) and more recent work on rank modulation for flash memories (see references in previous paragraph), much work on using permutations for error correction does not specify a physical channel where permutations might arise, or uses a metric on permutations that doesn't clearly correspond to some natural-looking process acting on and degrading an input permutation. I clarify this point below in reviewing several popular metrics on permutations. The following discussion (Section 4.5.2) doesn't add new material, but at least frames the question I would like to get at in future work.

4.5.2 In search of a compatible permutation metric

We are sticking to the single-shot communication setting in this Section, so what does error correction mean for us? Let's use an analogy from error correction in the noisy channel coding setting for a binary symmetric channel (BSC), where the channel is

used multiple times: A codebook is a set of codewords - allowable channel inputs (binary strings in this case). The channel output is some other binary string. So long as the noise isn't too large and the codewords are not too many, the channel output is probably closest in Hamming distance to the true channel input. Finding the codeword of minimum Hamming distance from the output is equivalent in this case to maximum likelihood (codeword-wise, not symbol-wise) decoding.

Although our permutation communication scheme is single-shot, a single permutation has multiple (k) components, which we will use in place of long binary strings. Our codebook consists of some subset of the permutation group S_k that forms the set of allowable channel inputs. In place of Hamming distance, we need some metric on permutations $\sigma_1, \sigma_2 \in S_k$; the Kendall τ distance $K(\sigma_1, \sigma_2)$ is a possible choice (Kendall, 1970)[13], where K counts the number of pair inversions. Previous work on using permutations to store data in a similar setting and using the Kendall τ distance includes (Wang, 2013). Another option is Spearman's footrule (based upon Spearman's rank-correlation coefficient (Spearman, 1904))[14] $D(\sigma_1, \sigma_2)$, which counts the total distance each rank has to move for the permutations to match, where "D" is as in (Diaconis and Graham, 1977). D and K are similar in that $K \leq D \leq 2K$ (Diaconis and Graham, 1977).

The analogy to decoding the binary symmetric channel is unsatisfactory, however, in that unlike the Hamming distance for the binary symmetric channel, both metrics K and D are incompatible with maximum likelihood decoding in our setting. Here compatibility means this: given a communication channel whose input x and output x' have the same alphabet $x, x' \in \mathcal{X}$ and channel transition matrix $P(x'|x)$, a metric d is called *compatible* with the channel if

$$P(x'|x) = f(d(x, x')) \tag{4.67}$$

where $f : \mathbb{R}_+ \to \mathbb{R}_+$ is some function. For example, for the binary symmetric channel with crossover probability p and block length m, $P(x'|x) = p^{d(x,x')}(1-p)^{m-d(x,x')}$, where d is the Hamming distance.

[13] $K(\sigma_1, \sigma_2) \equiv |\{(i,j) : i < j, \left(\sigma_1^{-1}(i) - \sigma_1^{-1}(j)\right) \cdot \left(\sigma_2^{-1}(i) - \sigma_2^{-1}(j)\right) < 0\}|$.
[14] $D(\sigma_1, \sigma_2) \equiv \sum_i |\sigma_1(i) - \sigma_2(i)|$.

Consider the following example of the incompatibility we claim: suppose the input permutation is $\mathbf{1}$ (identity) and consider $\sigma_1 \in S_k$ formed by swapping m pairs of adjacent ranks and $\sigma_2 \in S_k$ formed by placing the $m+1$-th rank first[15]. Then $K(\mathbf{1}, \sigma_1) = K(\mathbf{1}, \sigma_2)$ and $D(\mathbf{1}, \sigma_1) = D(\mathbf{1}, \sigma_2)$[16]. Recall that in our permutation communication setting, the channel consists of encoding a permutation in a pmf p (power distribution over k wires), sampling n times from this input pmf, and quantizing (by sorting) the output pmf (see Figure 4.5). Thus, the number of counts in the i-th wire/bin is binomially-distributed with parameters n, p_i. For large enough n and k it is therefore much more likely to have many adjacent bins swap ranks in the output pmf than to have a single bin deviate by a large amount from its expected rank[17].

We would thus like a metric on permutations that gives more weight to larger rank swaps than Kendall's τ and Spearman's footrule. Quadratic weight seems like a reasonable idea given the comments in Footnote 17:

$$S(\sigma_1, \sigma_2) \equiv \sum_{i=1}^{k} (\sigma_1(i) - \sigma_2(i))^2 \tag{4.68}$$

where we have used "S" as in (Diaconis and Graham, 1977). (Diaconis and Graham, 1977) also derive many relationships between and statistical properties of the metrics D, K, S, and others.

Is the metric S sufficient to complete our BSC decoding analogy; that is, is it compatible with our permutation channel? The question is not fully posed until we identify an encoder that maps permutations into pmfs. In the previous Section, in constructing an encoder that minimizes the asymptotic (in sample size n) error probability, we found an encoder such that the probability of bins i and $i+1$ swapping rank in the output is independent of i. This seems like a desirable property, but I don't know if it means that the metric S is compatible with our channel. The author is curious to find a metric on permutations that is compatible with this channel.

[15] e.g., $\sigma_1 = (2143)$, $\sigma_2 = (3124)$ for $k = 4, m = 2$.

[16] $= 2$, $= 4$, respectively, for the example in Footnote 15.

[17] This can be seen by approximating the binomials with normals and waving our hands that the probability of receiving a rank i positions wrong scales as e^{-i^2} (omitting the variance), so this probability is not invariant under splitting the "wrong" distance i into $i_1, i_2 : i_1 + i_2 = i$.

Are there other channels whose inputs and outputs are permutation-valued for which we can name a compatible metric? There is a natural choice, which we construct again by analogy with the BSC. An alternative way to characterize the BSC is as follows: starting somewhere on the $\{0,1\}^n$ hypercube, do a random walk[18] on the hypercube for s steps, where s is drawn from a Poisson distribution of mean m. Then the position of the random walk at time s is the same in distribution as the output of BSC with crossover probability p, where p is a function[19] of n and m. The BSC can also be constructed from the multinomial channel of Section 4.1 by sampling from the uniform pmf on n bins (the samples correspond to coordinate flips for the random walk) and then keeping only the parity of the histogram $\hat{p}$ (that is, $n\hat{p} \to n\hat{p} \pmod 2$).

For one possible permutation channel, our random walk is on the symmetric group S_k and at each time step we swap the rank of two adjacent bins. In the language of Section 4.3.1 and Figure 4.4, the random walk hops between adjacent sectors of the simplex Δ^{k-1}. This defines a channel with transition matrix $P(\sigma_t|\sigma_0)$, where σ_0 is the starting point of the random walk and σ_t is the permutation after t steps. I do not know if the quadratic weight metric (4.68) is compatible with this channel. Heuristically, the behavior is diffusion on a regular graph (each vertex is connected to $k-1$ vertices), so distribution of random walkers after t steps might be something like Gaussian in the distance from the start, justifying the quadratic distance metric. Whether this is true requires us to consider more than the fact that the graph is regular (we must consider the actual connectivity of adjacent sectors).

We do not need a compatible metric on permutations to create error-correcting permutation codes, but it would be nice to have a reasonable-seeming one. The author wonders whether the Kendall τ distance is one that best matches the actual noise channel for flash memory cells, given that the charge levels in a flash memory are not arbitrary labels, but measure something, so that perhaps it is much more likely that many pairs of adjacent bins swap than that a single bin arrives far from its expected rank. On the other hand, the Kendall τ distance might be natural if the channel is a

[18] The random walk is not lazy; exactly one coordinate changes parity at each time step.
[19] Sum over odd values: $p = \sum_{i \in \{1,3,5,\ldots\}} \text{PoissonPMF}\,(i; m/n)$.

sequence of "push-to-top" operations, as discussed in (Jiang et al., 2009), where the rank pushed to the top is perhaps drawn uniformly; we have not included an analysis of this case in this work.

4.5.3 Some ideas for permutation codes

Let's return to the setting of the single shot permutation-quantized multinomial channel of Section 4.2 and 4.3 (see Figure 4.5) and suggest some error correcting codes. As outlined in Section 4.5.1, our codebook consists of a subset or maybe a subgroup of the symmetric group S_k, so while our communication scheme is single-shot, the k ranks act like a block of bits in the usual (repeated use) noisy channel coding setting. We haven't found a metric on permutations compatible with maximum likelihood decoding for this channel in the sense of (4.67), but we have a channel matrix (4.12) so we can do maximum likelihood decoding.

Figure 4.11 illustrates the idea. Here we have selected two permutations, (132) and (231), as our codewords. So long as the output permutation is at most one adjacent transposition away from the input pmf (e.g. outputs (312) or (123) for input (132)), we can error correct the output permutatation to the correct input permutation. See the caption for details. Now how do we build sequences of codes as k grows for various rates?

4.5.3.1 Magic squares

One idea is to use magic squares. A magic square is a doubly stochastic matrix with integer entries; that is, the row and column sums are all the same. We can view the entries of a magic square as corresponding to the ranks of a permutation, so we are interested here in normal magic squares[20]: m by m squares such that each number (rank) $(1, \ldots, m^2)$ appears exactly once. Below is the historic normal magic square

[20] Henceforth by "magic square" we shall mean "normal magic square."

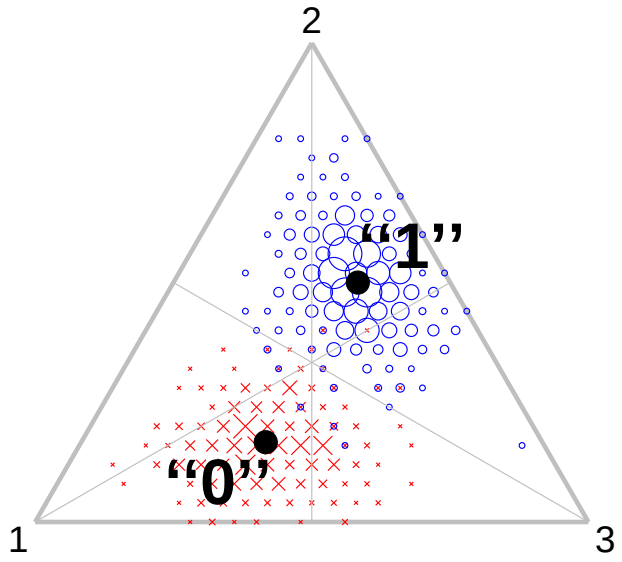

Figure 4.11: A permutation error-correction code, where “0,” “1” correspond to the permutations (231), (132), respectively, and the permutations (321), (213) (resp. (312), (123)) are error-corrected to “0” (resp. “1”). The 3! sectors of the 2-simplex are bounded by the gray lines. The encoder in this case maps (231) $\to p = (\frac{1}{3}, \frac{1}{2}, \frac{1}{6})$, (132) $\to p = (\frac{1}{6}, \frac{1}{2}, \frac{1}{3})$ (plotted as black dots on the 2-simplex). We drew 25 samples from both input pmfs 200 times and plotted a dot for each sampled pmf (histogram) (red crosses for “0”, blue circles for “1”). Larger markers indicate more occurrences of the same sampled pmf. A few uncorrected errors (blue or red markers on the wrong side of the 2-simplex) are seen.

of order 3:

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix} \tag{4.69}$$

The idea is to use only those permutations that form a magic square (we'll have to specify some order of filling the square), so we work now only with the symmetric group $S_k = S_{m^2}$ for $m \geq 3$ [21]. Call this subset[22] the set of magic permutations $M_{m^2} \subset S_{m^2}$:

$$M_{m^2} \equiv \{\sigma \in S_{m^2} : \sigma \text{ corresponds to a magic square}\} \tag{4.70}$$

The decoder then maps the output permutation to the most likely input magic permutation.

What can we say about this scheme? The magicness of a square is not a local property - it takes on the order of m looks to establish that a given square is not magic - so it feels like it should take quite a large perturbation to go from one magic permutation to another.

How far apart are two magic squares? Let's reflect that there are two sources of magic squares: the first source is that given a magic square, we can make more magic squares by permuting its rows and columns and by transposing it. This yields $2(m!)^2$ *equivalent* magic squares (where the 2 is due to the transpositions). Denote the set of corresponding permutations by $M^{\sigma}_{m^2} \subset M_{m^2}$:

$$M^{\sigma}_{m^2} \equiv \{\sigma' \in S_{m^2} : \sigma' \text{ corresponds to an equivalent magic square as } \sigma\} \tag{4.71}$$

[21] Normal magic squares exist for the positive integers except for $m = 2$, and the $m = 1$ case isn't useful here, so we assume $m \geq 3$.

[22] We can check that it is not a subgroup of S_{m^2}.

where σ is some magic permutation. Then the rate of our code is

$$R = \frac{\log \text{ size codebook}}{\log \text{ size input alphabet}} \tag{4.72}$$

$$= \frac{\log |M^{\sigma}_{m^2}|}{\log |S_{m^2}|} \tag{4.73}$$

$$= \frac{\log (2(m!)^2)}{\log ((m^2)!)} \tag{4.74}$$

$$\to \frac{1}{m} \quad m \to \infty \tag{4.75}$$

so when we restrict the codebook to only a subset of magic squares that are equivalent to each other, the rate vanishes as the size $m \to \infty$. Table 4.1 shows the first few approximate values for the rate as m increases.

m	R
3	0.334
4	0.230
5	0.177
6	0.145

Table 4.1: Approximate rate R for permutation code using only magic squares of size m that are all equivalent to each other.

The kinds of perturbation required to produce an error in this equivalent-only coding scheme is large - two columns or rows must switch, so on the order of m ranks must arrive incorrectly[23].

Now let's consider the other source of magic squares: non-equivalent squares. The number of these (counting only the equivalence classes) is only known up to $m = 5$, and an estimate is available for $m = 6$ (see Footnote 24 for references[24]). Table 4.2 below summarizes what is known and states the rate of our code if we use all of the equivalence classes of magic squares, not just a single one as above. We provide two estimates for the rate: R_1 corresponds to a codebook that uses only a

[23] And then probably not only incorrectly, but very far from their expected value, though we have not checked this.

[24] The case $m = 4$ is due to Frénicle de Bessy in 1693. The case $m = 5$ is due to Richard Schroeppel in 1973. The estimate for $m = 6$ is due to (Pinn and Wieczerkowski, 1998).

single representative of each of the classes of non-equivalent magic squares and R_2 corresponds to a codebook that uses all magic squares (the second codebook is a factor of $2(m!)^2$ larger than the first). Thus

$$R_2 = \frac{\log |M_{m^2}|}{\log |S_{m^2}|} \tag{4.76}$$

where M_m is defined in (4.70).

m	#non-equivalent magic squares	R_1	R_2
3	1	0	0.334
4	880	0.221	0.451
5	275305224	0.335	0.512
6	$(0.17745 \pm 0.00016) \cdot 10^{20}$	0.463	0.608

Table 4.2: Approximate rates for permutation codes using only all non-equivalent magic squares (R_1) and all magic squares (R_2).

We don't know $|M_{m^2}|$, but the table looks encouraging in that the rate grows with the size m. Magicness imposes a sublinear number of constraints ($2m$) on a permutation (of length m^2), so perhaps the rate goes to 1 as $m \to \infty$ or at least does not vanish. The author is curious to know the answer.

We also don't know how close two non-equivalent magic squares can be, and so cannot not provide an estimate for the error probability for this communication scheme[25]. Switching any two ranks breaks the magic; larger moves are needed. If a square subset of the entries of a magic square form a smaller magic square (this is possible), these entries can be transposed and row- and column-permuted while preserving the magic. Larger magic squares have more subsets, but on the other hand the mean rank grows with the square size, requiring a more improbable noise to effect an error per subset, so it is not clear to the author if the overall error probability shrinks or grows with m. An understanding of these issues would help us set the sample size n as a function of m to control the probability of error.

[25] at a fixed sample of size n from a pmf to which a magic permutation corresponds

Nor do we have a practical decoding scheme in mind beyond the maximum likelihood decoder[26]. Why discuss the magic squares schemes at all? The author thinks they are neat, and not obviously unpromising in the $m \to \infty$ limit (c.f. Table 4.2). Perhaps for a small value of m the performance could be characterized numerically and a not-too-large codebook stored, perhaps rendering the scheme useful for a flash memory storage application (if the cells are already laid out in a 2-dimensional square, it might be feasible to compute the row and column sums in hardware and then feed back on that information somehow). We could also imagine extending this to the case where some ranks are allowed to repeat ("multipermutations"), but the magicness of the squares remains.

4.5.3.2 Random codes

How about random permutation codes? By analogy with the random codebook used to prove achievability in the channel coding theorem, let's uniformly randomly draw m elements of the permutation group S_k to form our codebook. How do the error probability under maximum likelihood decoding and code distance and scale with k and m?

Relatedly, given a group G, draw some group elements $(g_1, \ldots, g_m)$ uniformly at random and consider either the subgroup induced by these elements. What is the size of this randomly generated subgroup? Given some metric on the group, what is the minimum or typical distance of the subgroup viewed as a codebook? For finite groups G, it might be helpful to have answers for symmetric groups, since every finite group is isomorphic to a subgroup of a symmetric group. I have nothing to say about this, but it would be an interesting scheme to consider.

[26] Checking the row and column sums of a matrix could give us a clue about unusually large or small entries.

4.A Computation of the permutation-resistant distribution

This Appendix computes the solution to the optimization problem stated in Section 4.3.8.

Let's compute the error exponents D_i (4.30). First, let's observe that

$$p_j^{E_i} = \frac{1}{Z_i} \begin{cases} \sqrt{p_i^* p_{i+1}^*} & : j \in \{i, i+1\} \\ p_j^* & : \text{otherwise} \end{cases} \tag{4.77}$$

so

$$\begin{aligned} D_i &= D(p^{E_i} || p^*) && (4.78) \\ &= \sum_{j \notin \{i,i+1\}} \frac{p_j^*}{Z_i} \log\left(\frac{p_j^*/Z_i}{p_j^*}\right) \\ &\quad + \frac{1}{Z_i}\sqrt{p_i^* p_{i+1}^*}\left(\log\left(\frac{\sqrt{p_i^* p_{i+1}^*}}{Z_i\, p_i^*}\right) + \log\left(\frac{\sqrt{p_i^* p_{i+1}^*}}{Z_i\, p_{i+1}^*}\right)\right) && (4.79) \\ &= -\left(1 - p_i^* - p_{i+1}^*\right)\frac{\log Z_i}{Z_i} \\ &\quad - 2\sqrt{p_i^* p_{i+1}^*}\frac{\log Z_i}{Z_i} && (4.80) \\ &= \left(\left(\sqrt{p_{i+1}^*} - \sqrt{p_i^*}\right)^2 - 1\right)\frac{\log Z_i}{Z_i} && (4.81) \\ &= c \ \forall i && (4.82) \end{aligned}$$

where we use the convention $0 \log 0 = 1$, $0 \log 0/0 = 1$ for the case $p_i^* = 0$.
Let's compute the Z_i (using eq. (4.77)):

$$Z_i = \sum_{j=1}^{k} \sqrt{\sigma^i(p_j^*)\ p_j^*} \tag{4.83}$$

$$= \sum_{j \notin \{i,i+1\}} \sqrt{(p_j^*)^2} + \sum_{j \in \{i,i+1\}} \sqrt{\sigma^i(p_j^*) p_j^*} \tag{4.84}$$

$$= 1 - p_i^* - p_{i+1}^* + 2\sqrt{p_{i+1}^*\ p_i^*} \tag{4.85}$$

$$= \left(1 - \left(\sqrt{p_{i+1}^*} - \sqrt{p_i^*}\right)^2\right) \tag{4.86}$$

Substituting this into eq. (4.81) we obtain:

$$-\log Z_i = c \tag{4.87}$$

$$\Downarrow$$

$$\sqrt{p_{i+1}^*} - \sqrt{p_i^*} = \sqrt{1 - e^{-c}} \equiv c' \in [0,1] \tag{4.88}$$

where $c' \in [0,1]$ follows from $c = D_i \geq 0$ since D_i is the K-L divergence.

Solving this recurrence relation for p_i^* we find

$$p_i^* = \left((i-1)c' \pm \sqrt{p_1^*}\right)^2 \tag{4.89}$$

where we are free to choose the value of p_1^*. Let's take the negative solution (the two solutions correspond to the same set of solutions $\{p^*\}$, parametrized differently) and rewrite this in the alternate form (eq. (4.34)):

$$p_i^* = \frac{(i+\kappa)^2}{Z_\kappa} \tag{4.90}$$

where $\kappa = -1 + \sqrt{p_1^*}/c' \geq -1$ and

$$Z_\kappa = \sum_{i=1}^{k} (i+\kappa)^2 = \frac{1}{6}\left(2k^3 + 3(2\kappa+1)k^2 + (6\kappa^2+6\kappa+1)k\right) \tag{4.91}$$

What remains is to choose κ to maximize $D_i = c$. It turns out the maximum is achieved for $\kappa = -1$ (alternately, we can impose the constraint $p_1^* = 0$, since we know from the first observation in Section 4.3.6 that it corresponds to maximizing $D_i = c$, and obtain $\kappa = -1$), yielding exactly the result (4.35) obtained by an intuitive argument:

$$p_i^* = \frac{1}{Z}(i-1)^2 \text{ for } i \in \{1, \ldots, n\} \tag{4.92}$$

$$Z = \sum_{i=1}^{k}(i-1)^2 = \frac{1}{6}\left(2k^3 - 3k^2 + k\right) = \frac{1}{6}k(2k-1)(k-1) \tag{4.93}$$

As noted earlier, it may be natural to have a floor on the minimum value of p^*. Suppose $p_1^* = p_{\min} > 0$, then we choose κ to satisfy (using eq. (4.90) with $i = 1$):

$$\frac{(1+\kappa)^2}{Z_\kappa} = p_{\min} \tag{4.94}$$

A solution always exists since $\frac{(1+\kappa)^2}{Z_\kappa} \in [0, 1/k)$ (map monotonic in $\kappa \in [-1, +\infty)$) and $p_{\min} \in [0, 1/k)$.

Let's compute the error exponent $D_i = c$ (rate function for making an error in inferring the sent permutation) for the case $\kappa = -1$ (optimal error exponent). Substituting eq. (4.90) into eq. (4.86), and this into eq. (4.81), we find:

$$\begin{aligned}
D_i = c &= -\log Z_i && (4.95)\\
&= -\log\left(1 - \left(\sqrt{p_{i+1}^*} - \sqrt{p_i^*}\right)^2\right) && (4.96)\\
&= -\log\left(1 - \frac{1}{Z_\kappa}\right) && (4.97)\\
&= -\log\left(1 - \frac{6}{2k^3 + 3(2\kappa+1)k^2 + (6\kappa^2+6\kappa+1)k}\right) && (4.98)\\
&\to \frac{3}{k^3} \text{ as } k \to \infty && (4.99)
\end{aligned}$$

Independent of κ for $k \to \infty$

Chapter 5

Energy efficiency in noisy channel coding

This Chapter explores schemes for noisy channel coding with an eye to energy efficiency, in particular focusing on the additive white Gaussian noise (AWGN) channel and the Poisson channel. The motivation for this train of thought initially came from considering the possible advantages of photonic computing circuits: these devices would not clearly offer better speed and are not smaller than classical computing components, but they use potentially orders of magnitude less power. How can we optimize them with regard to the advantage they potentially have?

For many popular channel models there is a parameter that acts like a power constraint, e.g. the mean power-constrained AWGN and the peak rate-constrained Poisson channel[1]. It might be a natural thing to "split" the power available to one such channel into multiple, noisier channels (e.g., splitting the AWGN power into multiple frequency bands, or increasing the number of flash memory cells while keeping constant the total charge available to all of them), or to "aggregate" multiple channels into a single, less noisy one. If possible, are these operations worth doing in terms of the total throughput of the split or aggregated channels? In attempting to find a unifying perspective for splitting and aggregation, we find that it's reasonable to call a channel "power-efficient" for some input power P^* if the capacity is linear

[1] Definitions come later.

in the power in the neighborhood of P^*. This offers a possible answer to the question of how to most efficiently use a given power budget over multiple transmissions. Channel splitting and aggregation let us go to a power-efficient point if we start at an inefficient one. Sections 5.2 and 5.4 apply these ideas to the AWGN channel, the Poisson channel, and some other natural-looking channels for which the answers turn out to be interest (like the spectrally constrained Poisson channel).

Along the way we delve more deeply into the AWGN in the low SNR regime. We consider in particular coding schemes for the AWGN in the finite block length setting[2] for block lengths inversely related to the SNR (again imagining that we have the freedom to split a fixed power budget between any number of transmissions).

We further present a coding scheme for the AWGN in the low SNR regime that relies on an input distribution very different from the (capacity-achieving) Gaussian distribution. Our input distribution is instead "spikes;" the sender mostly sends nothing and occasionally sends a spike that rises far above the noise. We show that this input distribution is capacity-achieving in the limit of some parameters. We further discuss a communication scheme based on this distribution: the sender encodes the message in the locations of the spikes and the receiver thresholds the channel output to find the locations of the spikes. This is appealing because it potentially means no error correction on the part of the receiver (whereas using a Gaussian input distribution would require increasingly complicated error correction as the SNR goes to 0). We prove some encouraging observations about the feasibility of this communication scheme and point out what needs to be done to complete the proof of feasibility.

This Chapter is organized as follows: Section 5.1 gives a lightning recap of noisy channel coding and establishes notation. Section 5.2 introduces the idea of "channel-splitting:" dividing the total available power between independent, noisier copies of the initial channel, discusses when this might be a natural thing to encounter in practice, and makes some observations for the example of the AWGN. Section 5.3 considers these ideas in the finite block length communication regime. Section 5.4 discusses the related case of "channel-aggregation." Section 5.5 presents some

[2] For finite block lengths, there is a penalty to the maximum achievable rate in terms of the channel dispersion, a measure of the channel noisiness; this can set a limit to how many times we should split a channel into noisier channels in terms of maximizing achievable communication rate.

observations about splitting and aggregation for a broader class of channels. Section 5.6 presents a coding scheme for the AWGN at low power and concludes the Chapter.

5.1 Noisy channel coding in four pages

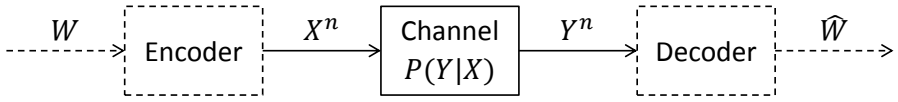

Figure 5.1: Noisy channel coding setup. Our focus in this Chapter is on the channel.

In this Section we remind the reader of channel coding concepts in sufficient detail for our discussion to proceed; the author points the interested reader to (Cover and Thomas, 2006) for an excellent introduction. We have already encountered these concepts earlier in this thesis[3], but state them so that this Chapter can be read independently of the rest.

Recall the setup of communication over a noisy channel, shown in Figure 5.1 in the notation of (Cover and Thomas, 2006). Here $X^n = (X_1, \ldots, X_n)$ and $Y^n = (Y_1, \ldots, Y_n)$ are the channel inputs and outputs, respectively, of *block length* n and W and $\hat{W}$ are the message to be encoded and the message estimated by the decoder, respectively. Let's suppose that the set of possible messages is $W \in \{1, \ldots, M\}$ for some $M \geq 1$ and that each message is equally likely to be sent. The encoder maps the M possible messages W to channel inputs X^n of length n - this mapping is the *codebook*. The communication *rate* is defined as the ratio

$$R = \frac{\log M}{n} \tag{5.1}$$

(throughout, our logarithms are in arbitrary but consistent base).

[3] In Chapter 2 about a photonic LDPC decoder circuit.

If $M = 2^k$, e.g., if the messages W are binary strings of length k, then this reduces to

$$R = \frac{k}{n} \tag{5.2}$$

For example, for the (3,1) Hamming code that encodes "0" with "000" and "1" with "111" to protect from bit flips, the rate is 1/3.

The channel is characterized by a probability distribution of outputs given the inputs $P(Y^n|X^n)$. In what follows we assume further that the channel is memoryless and time-invariant, meaning that the probability distribution of the output depends only on the current channel input:

$$P(y^n|n^n) = \prod_{i=1}^{n} P(y_i|x_i) \tag{5.3}$$

(we denote random variables in capitals, e.g., X, and outcomes in lowercase, e.g., x).

Finally, the decoder takes in channel outputs Y^n and produces estimates $\hat{W}$ of the message W.

The rate R is *achievable* if there exists a sequence of codes (encoder and decoder pairs) indexed by the block length n such that the maximal probability of error when transmitting any of the $M = 2^{nR}$ messages tends to 0 as $n \to \infty$. Suppose we fix a particular encoder (mapping from messages W to channel inputs X^n), which in turn fixes the rate and the distribution $P(X^n)$ and its marginals $P(X)$. Does a decoder exist such that this rate is achievable?

The channel coding theorem due to (Shannon, 1948) answers this question: reliable communication is possible at any rate below the mutual information $I(X;Y)$ and impossible at any rate above $I(X;Y)$. The maximum possible value of the mutual information over all choices of input distribution $P(X)$ is the channel *capacity*[4] C. Below are some equivalent ways of writing the mutual information $I(X;Y)$ between random variables X and Y :

[4] This is the "information" channel capacity, which Shannon established is equal to the "operational" capacity, defined as the supremum of achievable rates.

$$
\begin{aligned}
I(X;Y) &= I(Y;X) && (5.4)\\
&= H(X) - H(X|Y) && (5.5)\\
&= H(Y) - H(Y|X) && (5.6)\\
&= H(X) + H(Y) - H(Y,X) && (5.7)\\
&= D(P_{XY}||P_X P_Y) && (5.8)\\
&= \mathbb{E}_{P_{XY}}\left[\log \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}\right] && (5.9)\\
&\geq 0 && (5.10)
\end{aligned}
$$

Where $H(X)$ is the Shannon entropy

$$
\begin{aligned}
H(X) &= -\sum_{x\in\mathcal{X}} P(x)\log P(x) && (5.11)\\
&= \mathbb{E}\left[\log \frac{1}{P(X)}\right] && (5.12)
\end{aligned}
$$

where $\mathcal{X}$ denotes the alphabet of X. Here we are using sums over discrete alphabets $\mathcal{X}$. Later we will use continuous-valued variables X and replace the sums with integrals. $D(P(X)||Q(X))$ is the Kullback-Leibler (KL) divergence

$$
\begin{aligned}
D(P(X)||Q(X)) &= \sum_{x\in\mathcal{X}} P(x)\log \frac{P(x)}{Q(x)} && (5.13)\\
&= \mathbb{E}_P\left[\log \frac{P(X)}{Q(X)}\right] && (5.14)
\end{aligned}
$$

Finally, the *capacity* C is a property of the channel defined[5] by maximizing $I(X;Y)$ over the set of input distributions $P(X)$ (that is, over the possible choices of the encoder):

$$
C = \max_{P(X)} I(X;Y) \tag{5.15}
$$

[5] See Footnote 4 on the distinction between "information" and "operational" capacity.

Let's denote by $P^*(X)$ the capacity-achieving distribution:

$$P^*(X) = \arg\max_{P(X)} I(X;Y) \tag{5.16}$$

By concavity of the mutual information in $P(X)$ and by convexity of the set of probability distributions $P(X)$, this is a global maximum (rather than a supremum). Reliable communication is thus possible at any rate below capacity and impossible at any rate above. The channel-coding theorem does not tell us how to construct an encoder/decoder pair that achieves a particular rate below capacity[6], but tells us when this is possible.

The *error probability* for a communication scheme is the probability that the decoder output does not match the encoder input:

$$\mathrm{P_e} = P(\hat{W} \neq W) \tag{5.17}$$

where the randomness is due to the use of the noisy channel and the randomness in the encoder input. We can assume a uniform distribution over encoder inputs $P(W)$[7] to define $\mathrm{P_e}$.

Many variations exist on the setup of Figure 5.1 exist (e.g., including feedback to the transmitter), but we work below with this classic version.

5.2 Channel splitting

Given a noisy communication channel, it might be possible to define (and implement!) "splitting" the channel into multiple, noisier — meaning with smaller capacity — channels, as depicted schematically in Figure 5.2. For example, in a flash memory cell, one imagines keeping the total amount of charge deposited in all cells constant while increasing the number of cells, so that individual cells are noisier.

[6] (Shannon, 1948) uses a random codebook and jointly typical decoding, but this can be impractical.

[7] The source-channel separation theorem lets us do this without loss of generality in terms of the distribution of W.

If possible, is this worth doing, say in terms of the capacity of the noisier channels[8]? The answer is positive in some natural-seeming settings. This Section discusses this idea for the AWGN and makes some observations relating the channel splitting operation to the finite block length setting. We further discuss the design of flash memory cells as a possible application of these ideas.

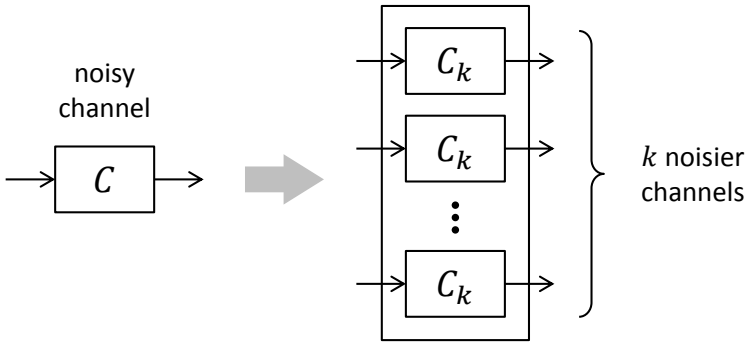

Figure 5.2: Splitting a noisy channel with capacity C into $k > 1$ noisier channels with capacity $C_k < C$.

5.2.1 AWGN with mean power constraint

Consider the additive white Gaussian noise (AWGN) channel, specified by input X, output Y

$$Y = X + Z \tag{5.18}$$

where $Z \sim \mathcal{N}(0, N)$ is the normally-distributed noise with variance N. The channel input X is constrained to satisfy $\mathbb{E}[X^2] = P$, where P is the signal power. Henceforth, we set

$$N = 1 \tag{5.19}$$

and refer to P as the signal to noise ratio (SNR). The capacity is given by

$$C = \frac{1}{2} \log (1 + P) \tag{5.20}$$

[8] In the notation of Figure 5.2, is kC_k larger than C?

Consider the low and high SNR P regimes:

$$C \approx \begin{cases} \frac{P}{2} & : P \ll 1 \\ \frac{1}{2}\log P & : P \gg 1 \end{cases} \tag{5.21}$$

Thus at low SNR ($P \ll 1$) communication over the Gaussian channel is exponentially more power-efficient than in the high power regime. We shall see below that splitting the Gaussian channel can allow us to recover the linear scaling of capacity with input power.

We can think of two ways to define splitting this channel, which are equivalent for our purposes: splitting in time and in frequency. Suppose we arrive at the Gaussian channel by setting some sampling time T_s per transmission and holding the input signal X fixed during each sampling window:

$$Y = \frac{1}{T_\mathrm{s}} \int_0^{T_\mathrm{s}} (X dt + dW_t) \sim \mathcal{N}(X, 1/T_\mathrm{s}) \tag{5.22}$$

where W_t is a Wiener process. Thus, splitting the sampling time into k equal parts of size T_s/k is equivalent to increasing the noise variance $1/T_\mathrm{s}$ by a factor of k. We can also imagine splitting the signal power into different frequency bands while leaving the sampling time per band unchanged at T_s. Suppose our Gaussian channel accepts inputs at some optical frequency. If we can use k different frequencies (that are far enough apart for us to ignore interference at the receiver), we again have k parallel Gaussian channels with the same noise variance, but with sender power reduced to P/k for each channel. This frequency division is also for our purposes equivalent to simply using k wires with power P/k in each. For both time- and frequency-splitting, the SNR is decreased by a factor of k per output sample.

Let's compute the capacity $C_{\mathrm{split}}^{(k)}$ of a Gaussian channel split into k identical channels whose SNR is reduced by a factor of k with capacity C_k. Since the capacities of

independent channels add, we have

$$C_{\text{split}}^{(k)} = kC_k \tag{5.23}$$

$$= k \cdot \frac{1}{2} \log\left(1 + \frac{P}{k}\right) \tag{5.24}$$

$$\geq C = \frac{1}{2} \log\left(1 + P\right) \ \forall k \geq 1 \tag{5.25}$$

Thus the splitting operation always helps in terms of the total capacity of the parallel noisier channels.

Note that (5.24) is almost the expression for the capacity of a band-limited Gaussian channel (as given in (Cover and Thomas, 2006)) $Y(t) = (X(t) + Z(t)) * h(t)$, where the band-pass filter $h(t)$ has 0 amplitude for all frequencies greater than W (aside from a leading factor of 2 and a redefinition of the noise power):

$$C_{\text{band-limited}}^{W} = W \log\left(1 + \frac{P}{WN_0}\right) \tag{5.26}$$

where $N_0/2$ is the noise power spectral density. This is not surprising, since the above expression (5.26) can be derived by computing the capacity in bits per second and setting the sampling time to $1/(2W)$, just as we did when "time-splitting" the Gaussian channel and computing the capacity.

As the number of splits k grows, we recover the linear scaling of capacity with power:

$$C_{\text{split}}^{(\infty)} = \lim_{k \to \infty} C_{\text{split}}^{(k)} = \frac{P}{2} \tag{5.27}$$

The capacity of the AWGN channel (5.20) and the capacity of the split AWGN channel (5.24) are plotted in Figure 5.3. Note that we only gain much from splitting the channel in the inefficient high SNR, $P \gg 1$ regime.

If we have an actual physical channel that we approximate as a Gaussian channel, and imagine splitting it into ever more noisier channels, at some point the approximation of the channel as Gaussian may no longer hold. For example, the channel noise may arise due to discrete photodetection events ("dark counts") at the receiver; if the sampling time is short compared to the typical time between these dark counts,

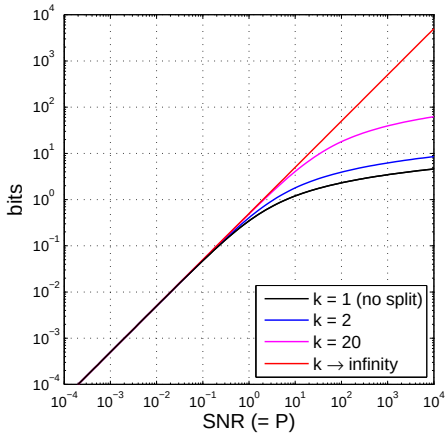

Figure 5.3: (black) capacity of the AWGN channel as a function of SNR ($= P$), (blue) split 2 times, (magenta) split 20 times, (red) in the limit of infinitely many splits. Natural log.

we should no longer treat the noise as a normally-distributed random variable. We address this issue again in Section 5.4.1 when discussing the Poisson channel.

5.3 Coding schemes for split channels in the finite block length setting

Although splitting the AWGN boosts the total capacity of the parallel noisier channels, there is a potential pitfall in terms of performance in the finite block length setting, which potentially limits the use of this operation for an actual implementation. We shall be more precise later, but roughly the trouble is that noisier channels require longer block lengths to allow communication within a given fraction of capacity at a given block error rate. By splitting our channel into k noisier channels, the block length for each of these noisier channels must be longer than the block length for the unsplit channel to achieve the same performance. This increase must then be multiplied by k to get the total new block length, possibly much larger than the block length used for the unsplit channel.

We describe two ways to deal with this issue. The first is to do "nothing:" we compute the potential performance hit in terms of block error probability and nearness to capacity for the split channel scheme. It turns out that the optimal number of channel splits is no longer $k \to \infty$, but some value determined by the unsplit channel power and block length. The second way is to not give each of the noisier split channels its own block length, but to view the noisier channels as together forming the block. Instead of using the parallel channels independently, we will distribute codewords across the noisier channels; this can be useful even if the individual noisier block lengths are equal to 1. This thinking further leads us to quantify the energy budget necessary to communicate a given number of bits at a given block error rate when given the freedom to choose how to distribute the energy between some number of transmissions.

5.3.1 Facts about the finite block length setting

The channel capacity (5.15) sets the supremum of achievable rates in the limit of infinite block length ($n \to \infty$). For finite block length n and a fixed maximum allowable error probability ϵ, the supremum of achievable rates is in general strictly below capacity. The intuition for this is that the mutual information $I(X;Y)$ (which is maximized by the capacity) is the mean of the random variable $i(X;Y)$ defined below:

$$I(X;Y) = \mathbb{E}_{P_{XY}}\left[i(X;Y)\right] = \mathbb{E}_{P_{XY}}\left[\log \frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}\right] \tag{5.28}$$

where the expectation value is taken over the joint distribution $P_{XY}(X,Y) = P_{Y|X}(Y|X)P_X(X)$. For finite n, $i(X;Y)$ fluctuates about its mean, and so $\frac{1}{n}I(X^n;Y^n)$ can dip below the design rate (5.1) set by the encoder. When this happens, it is likely that multiple channel inputs are jointly typical with the output, leading to a high probability of error. For a discussion of this that goes beyond rough intuition, see (Polyanskiy et al., 2010).

Fortunately, quite a few things are known: (Shannon, 1959) established tight bounds for the AWGN with a finite block length, and (Strassen, 1962) and (Polyanskiy et al., 2010) established a relationship between the block length n, the maximum error

probability ϵ, and the degree to which the capacity can be approached in this setting for the discrete memoryless channels and the Gaussian channel:

$$\log M^*(n,\epsilon) = nC - \sqrt{nV}Q^{-1}(\epsilon) + O(\log n) \tag{5.29}$$

here $M^*(n,\epsilon)$ denotes the maximum codebook size achievable at block length n and with maximum error probability ϵ, C is the channel capacity, $Q^{-1}(x)$ is the inverse of $Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt = \frac{1}{2}\left(1 - \operatorname{erf}\left(x/\sqrt{2}\right)\right)$, and V is the *channel dispersion*, which measures the "stochastic variability of the channel" (Polyanskiy et al., 2010). For the case of a discrete memoryless channel, (Strassen, 1962) showed that the expression above holds with $V = \operatorname{Var}[i(X;Y)]$ for X distributed according to the capacity-achieving distribution $P^*(X)$[9], while (Polyanskiy et al., 2010) provides a definition of the channel dispersion V applicable to a broader class of channels. A useful implication of (5.29) due to (Polyanskiy et al., 2010) is a relation for the block length $n^*(\eta,\epsilon)$ necessary to communicate at a fraction η of capacity with maximal probability of error ϵ. Writing

$$\log M^*(n,\epsilon) = n^*(\eta,\epsilon)\eta C \tag{5.30}$$

we subsitute this into (5.29) along with $n \to n^*(\eta,\epsilon)$ to obtain:

$$n^*(\eta,\epsilon) \approx \frac{V}{C^2}\left(\frac{Q^{-1}(\epsilon)}{1-\eta}\right)^2 \tag{5.31}$$

where the approximation denotes discarding the $O(\log n)$ term in (5.29).

For the AWGN channel with mean, peak, and equal-power constraint, (Polyanskiy et al., 2009) established that the channel dispersion is[10]

$$V_{\text{AWGN}} = \frac{P(P+2)}{2(P+1)^2} = \begin{cases} \left(P - \frac{3}{2}P^2 + O\left(P^3\right)\right) & : P \to 0 \\ \left(\frac{1}{2} - \frac{1}{2P^2} + O\left(\frac{1}{P^3}\right)\right) & : P \to \infty \end{cases} \tag{5.32}$$

[9] In the case of multiple capacity-achieving distributions, use one with the smallest (largest) variance if $\epsilon < 1/2$ ($\epsilon > 1/2$) (Polyanskiy et al., 2010).

[10] For arbitrary base logarithms, the expression (5.32) should be multiplied by $\log^2 e$. We work with natural logarithms in this Section.

5.3.2 Split AWGN and finite block length

Let's apply these statements to the split AWGN channel of Section 5.2.1. First, we use the split noisier channels independently, and show that there can be an optimal number of splits in terms of the size of the codebook. Then, we code "across" the sub-channels rather than use them independently and heuristically suggest a relationship between a total energy budget and the number of bits that can be sent with a given block error probability.

Suppose we use the unsplit AWGN with mean power constraint P at a fraction η of capacity with maximum error probability ϵ. Then the block length we are using is given by $n = n^*(\eta, \epsilon)$ (5.31). Now we split this channel to form k parallel AWGN channels with mean power constraint P/k, each with the same block length n [11] (so the total number of transmissions across the noisier channels is nk). Denote the capacity and dispersion of the individual noisier channels by C_k, V_k, respectively. Let's compute using (5.29) the size $M^{*(k)}$ of the codebook for the collection of the k noisier channels. Since we are using them independently, the log size of this code book is k times the log size of the codebook for one of them:

$$\log M^{*(k)} \approx k\left(nC_k - \sqrt{nV_k}Q^{-1}(\epsilon)\right) \tag{5.33}$$

$$= k\frac{n}{2}\log\left(1+\frac{P}{k}\right) - k\sqrt{n\frac{P(P+2k)}{2(P+k)^2}}\,Q^{-1}(\epsilon) \tag{5.34}$$

where the approximation sign means discarding the $O(\log n)$ term in (5.29). For small number of splits k, say that $k \ll P$, the first term of (5.34) grows as $k\log\frac{P}{k}$, while the second term grows as k, but with possibly a smaller numerical prefactor than the first term depending on n and ϵ. For large values of k, say $k \gg P$, the first term limits to a constant, while the second term grows as $\sqrt{k}$. Thus we expect there to be some value of k that maximizes (5.34) for some values of P, ϵ, n.

We could try to get limiting expressions for the optimal number of splits in various regimes, but there are many knobs to turn (P, ϵ, n), the expression is difficult to work

[11] This might be natural if we split the power P into different frequency bands that are measured simultaneously at the receiver while holding the total number of channel uses fixed.

with algebraically, and it's unclear to the author which regimes are the most natural, so let's just look at one example in Figure 5.4. Here the initial, unsplit AWGN has mean power constraint (SNR) $P = 10$, maximum error probability $\epsilon = 1\%$, and block length $n = 100$. The log size of the codebook for these parameters is then given by (5.29) $\log M^*(n, \epsilon) \approx 1100$ (blue line). We plot (solid red line) the log codebook size of the k split channels with SNR P/k using (5.34).

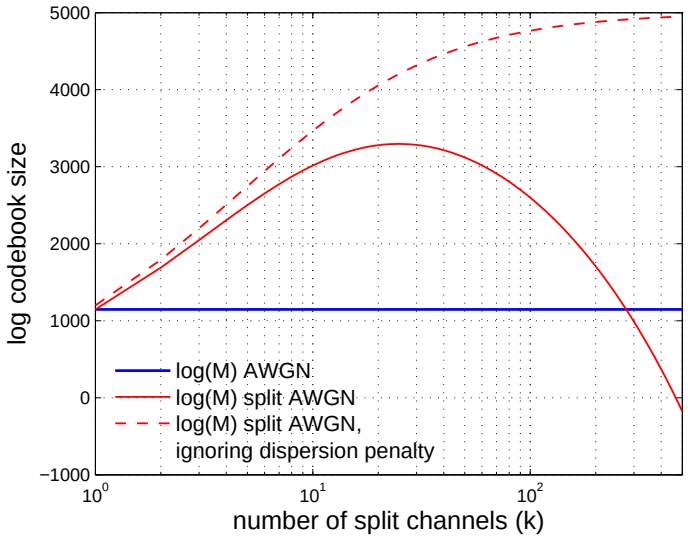

Figure 5.4: (blue line) log codebook size ($\log M$) for the AWGN with SNR $P = 10$, maximum error probability $\epsilon = 1\%$, block length $n = 1000$. (solid red line) log codebook size for the AWGN split into k channels with SNR P/k. (dashed red line) same as solid red line, but ignoring the dispersion penalty (second term in (5.34)). When the log codebook size is negative, the approximation of ignoring the $O(\log n)$ term in (5.29) is no longer valid.

We see in Figure 5.4 that as the number of splits k grows, the codebook size for the split AWGN (solid red line) grows larger than the codebook size for the unsplit AWGN (blue line), peaks for $k \approx 25$, and then dips below the codebook size of the unsplit AWGN for $k \gtrsim 250$. For $k \gtrsim 450$, the expression for the log codebook size becomes negative; at this point the approximation of discarding the $O(\log n)$ term is no longer valid. Note that without the channel dispersion penalty (second term in

(5.29)), the codebook size for the split channels grows for all $k \geq 1$ and asymptotes to $\frac{nP}{2} = 5000$ (red dashed line[12]).

These observations make sense: the AWGN is power-efficient — its capacity is linear in P — in the low SNR regime, so given an AWGN with $P \gg 1$, we should "split" it into about[13] $1/P$ noisier channels so that each is power-efficient. Splitting beyond this point doesn't help much with power-efficiency, but does make the pieces noisier, so we incur a growing penalty in terms of the codebook size as the number of splits increases.

5.3.3 Coding across split blocks

The discussion in the preceding section treated the k noisier split AWGN channels as independent. Now let's see what happens if we view the k split channels as forming the block. That is, instead of using a block length n inherited from the unsplit AWGN for each noisier channel, let's give each of the noisier channels a block length of 1 and code "across" these channels for a block length of k.

Why consider this scenario? Imagine someone gives you a fixed amount of energy (not power) to communicate. How should you use this energy to send the greatest number of bits with a particular probability of error? You are permitted to distribute your energy budget between an arbitrary number of transmissions. Distributing your energy between different transmissions is the same as our channel splitting.

Again denoting by C_k, V_k the capacity and dispersion, respectively, of the AWGN with SNR P/k, let's recall the expression (5.29) for the codebook size, but now set $n = k$ since we are coding across the k split channels:

$$\log M^*(k, \epsilon) = kC_k - \sqrt{kV_k}Q^{-1}(\epsilon) + O(\log k) \tag{5.35}$$

$$= \frac{k}{2}\log\left(1 + \frac{P}{k}\right) - \sqrt{k\frac{P(P+2k)}{2(P+k)^2}}\, Q^{-1}(\epsilon) + O(\log k) \tag{5.36}$$

[12] It doesn't look like it asymptotes on a log scale, but we know it does!

[13] This value depends on P, ϵ, and n and is obtained by maximizing (5.34), which we haven't done.

Let's be clear that we are playing fast and loose with our application of the results of (Polyanskiy et al., 2010) to our problem: for (Polyanskiy et al., 2010), the channel capacity and dispersion are fixed, while for us they vary with the blocklength, potentially invalidating proofs of (Polyanskiy et al., 2010) that rely on approximating the mutual information density with a normal random variable. The hope is that these normal approximations hold at least for a range of k large enough to make our claims, but I have not yet checked these hopes. The author intends to give this more thought in the future. For now let's call this approach a heuristic and proceed.

What value of k maximizes the expression (5.35)? Let's guess that we want k large, $k \gg P$. Then using the low-SNR expressions for the AWGN channel capacity (5.21) and dispersion (5.32), $C_k \to \frac{P}{2k}$ and $V_k \to \frac{P}{k}$ for $k \to \infty$, in which case the expression (5.35) becomes

$$\log M^*(k,\epsilon) = k\frac{P}{2k} - \sqrt{k\frac{P}{k}e}\, Q^{-1}(\epsilon) + P^2 O\left(\frac{1}{k^2}\right) + O(\log k) \tag{5.37}$$

$$\approx \frac{P}{2} - \sqrt{P}Q^{-1}(\epsilon) \tag{5.38}$$

(Note that the dispersion penalty term now tends to a constant, rather than grows with k, as for the independent subchannels case (5.34).) where the approximation is valid for $k \gg P$. We must be careful not to let k get too large, since the first two terms are k-independent, while we discarded the $O(\log k)$ term. The approximation is then valid only for large enough P, such that $\frac{P}{2} \gg \log(k)$ (and $P/2 \gg \sqrt{P}Q^{-1}(\epsilon)$). Therefore the conditions we need for the approximation to hold are

$$1 \ll P \ll k \ll e^{P/2} \tag{5.39}$$

The expression (5.38) is intuitively appealing: working with natural logs, it says that the number of natural bits one can send given an energy budget P with error probability ϵ is approximately $\frac{P}{2} - \sqrt{P}Q^{-1}(\epsilon)$. The way to send this many bits with this error probability is to divide the energy P into k transmissions, where k satisfies the conditions (5.39).

Can we do better than this? We guessed that letting k be large is a reasonable idea, and it's nice that letting $k \to \infty$ results in a constant limit for the codebook size, but we have not solved for k that maximizes (5.35). Numerically we have been able to find cases of finite k maximizing (5.35) for very large $Q^{-1}(\epsilon) = O(\sqrt{P})$, so that the first and second terms of (5.35) are of the same magnitude. This again strays outside the setting of (Polyanskiy et al., 2010), where ϵ is held constant, so that normal approximations hold as the block length grows. So we don't know how to maximize (5.35), and even if we want to, given its deviation from the assumptions of (Polyanskiy et al., 2010).

Using the same intuition as in the previous Section, given a fixed energy $P \gg 1$ budget, we should split this energy enough times so that each individual transmission is in the energy-efficient regime where capacity is linear in the power. There is scant benefit in terms of codebook size to splitting the energy further, and we don't yet understand what the codebook size is when we split many more times anyway, so we keep our comments to about $O(P)$ splits.

5.4 Channel Aggregation

So far we have discussed splitting the power available to a noisy channel into multiple, noisier channel. What about the reverse direction: reducing the number of noisy channels to make a single less noisy one, as depicted schmeatically in Figure 5.5. Is this "aggregation" ever worth doing in terms of the capacity of the single channel[14]? It isn't for the mean power constrained AWGN, since splitting always increases the capacity (see (5.24))[15].

In this Section we apply the aggregation operation to a Poisson channel and show that it boosts capacity. The Poisson channel we focus on has a peak intensity constraint p, which of course means something different than the mean power constraint for the AWGN channel, but is related in that it can be viewed as a constraint on

[14] In the notation of Figure 5.5, is $C^{(k)}$ larger than kC?

[15] In the finite block length regime, splitting enough times can reduce the achievable rate if the subchannels are used independently; see Section 5.3. We focus on the capacity for now and return to the finite block length setting later.

the sender power in some settings (like photon detection). Whereas capacity is linear in power for the AWGN in the low SNR regime, capacity is linear in power for the Poisson channel in the high peak intensity regime.

Where might these channels we are aggregating be coming from? Perhaps the sender has access to multiple optical fibers and is trying to decide how best to distribute available power between them in terms of capacity. Later in Section 5.5 we imagine having access to a large supply of identical channels for aggregation or splitting.

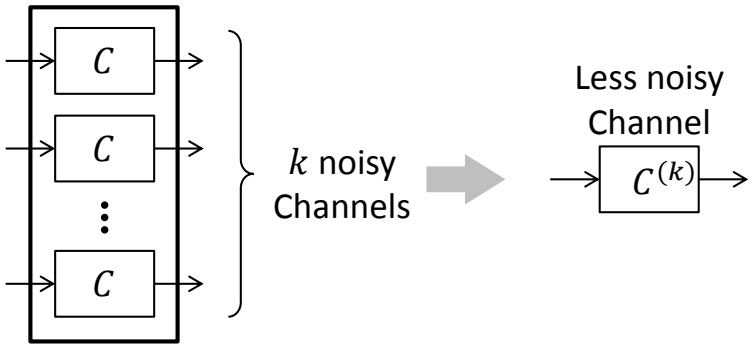

Figure 5.5: Aggregating k noisy channels with capacity C into one less noisy channel with capacity $C^{(k)} > C$.

In Section 5.5 we attempt to provide a unifying perspective on our channel splitting/aggregation observations; the principle that emerges is that capacity "ought to" be linear in power for power-efficient channels. In Section 5.5.3 we provide examples of channels that live in this framework in a less extremal way than the AWGN and the Poisson channel — the spectrally limited Poisson channel, which is power-efficient at some positive input power (rather than 0 or infinity), and one other example.

5.4.1 Poisson channel with peak amplitude constraint

Consider the Poisson channel, whose input is an *amplitude* λ_t in continuous time $t \in [0, T]$ [16] and whose output is a Poisson point process N_t with rate λ_t, that is,

$$N_t \sim \text{Poisson}\left(\int_0^t dt\lambda_t\right) \tag{5.40}$$

The noise is supplied by a "dark" count rate n, meaning $\lambda_t \geq n \ \forall t$. We consider here a peak amplitude constraint p, meaning $\lambda_t \leq n + p \ \forall t$ [17]. Thus

$$\lambda_t \in [n, n + p] \ \forall t \tag{5.41}$$

The sender attempts to modulate the amplitude λ_t within the allowed range to communicate with the receiver via the arrival times of the clicks. An example use of the Poisson channel is shown in Figure 5.6 (see caption for details).

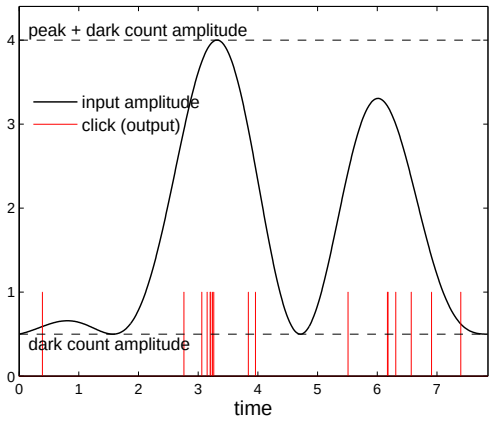

Figure 5.6: Sample use of the Poisson channel with peak amplitude constraint. (black line) input amplitude λ_t. (red lines) output clicks. Dashed black lines denote the amplitude bounds (5.41). The black curve is just some curve, nowhere near what a sample input from a capacity-achieving distribution looks like (it would not vary so smoothly).

[16] T plays the role of the block length in continuous time.
[17] The "input" amplitude λ_t is the sum of the noise amplitude and the sender-supplied amplitude.

The Poisson channel is a model for optical photodetection. The instantaneous click rate is proportional to the amplitude of the electric field at the detector[18] plus a "dark count" rate corresponding to various things (maybe the external field stochastically fluctuates a bit at all times, maybe the electronics in the detector is noisy).

A mean amplitude constraint might be a natural thing to add. The capacity with both a peak and mean constraint was worked out by (Davis, 1980); it turns out that above a certain threshold, the mean constraint is inactive for a capacity-achieving distribution, and the capacity is given by (5.42). Below this threshold, there is another expression in terms of both constraints. This would complicate our story, but is worth keeping in mind for possible extensions.

The capacity[19] was found by (Kabanov, 1978; Davis, 1980):

$$C(p,n) = n\left(\frac{1}{e}\left(1+\frac{p}{n}\right)^{1+\frac{n}{p}} - \left(1+\frac{n}{p}\right)\log\left(1+\frac{p}{n}\right)\right) \tag{5.42}$$

Note that this is not a function of just the ratio p/n, a sort of signal to noise ratio (SNR) for the Poisson channel, while the capacity of the AWGN (5.20) is conveniently a function of just the appropriate SNR (P/N). This is because we are working in continuous time, so $C(p,n)$ has units of bits/s. We can use the noise rate n to set the units of time or just set $n=1$ to obtain the unitless quantity

$$C(p) \equiv \frac{1}{n}C(p,n) \tag{5.43}$$

$$= \frac{1}{e}(1+p)^{1+\frac{1}{p}} - \left(1+\frac{1}{p}\right)\log(1+p) \tag{5.44}$$

We compute the limiting forms of this expression below (5.45) and find that $C(p) \sim p^2$ for $p \ll 1$, $C(p) \sim p \gg 1$, so that the high SNR $p \gg 1$ regime is

[18] The committed reader might recall this model from the quantum optical setting, where the role of the field is played by the collapse operators; see (1.3) and (1.11).

[19] Here to define the capacity C we take $C = \lim_{T\to\infty} \sup \frac{1}{T} I_T(\theta, N)$, where $\{N_t\}_{t\in[0,T]}$ counts the number of clicks up to time t, the supremum is taken over admissible schemes, and admissibility is defined in terms of adapted cadlag processes $\{\theta_t\}_{t\in[0,T]}$. We refer the reader to (Kabanov, 1978; Davis, 1980) for definitions and derivations; our interest in this Section is in the capacity in terms of the parameters.

power-efficient. This may seem counterintuitive, but follows from the convexity of $C(p)$: the $p \gg 1$ regime therefore sets the upper bound on $C(p)$.

Aggregating the Poisson channel: motivation

How could aggregation for the Poisson channel arise "in practice?" We imagine a bundle of optical fibers going to an array of photodetectors, each fiber modeled as a Poisson channel with a peak amplitude constraint p and dark count rate $n = 1$. Given a power budget, how should the sender distribute this power across the fibers if the goal is to maximize capacity? We saw that for AWGN, the answer is to split the power P into $k \to \infty$ "wires" (say, different frequency bands), and we see below that the opposite is true for the Poisson channel: all of the power should go into one fiber.

(Davis, 1980) observed that given a constraint on total energy split between two different polarizations of light, it is better capacity-wise to put the energy in just one of the two polarizations, while (Lapidoth and Shamai, 1998) note that the maximum throughput of the Poisson multiple access channel[20] is linear in the number of users, so our observations are not new. I am using the Poisson channel as an example of our splitting/aggregation framework and as a step to build another channel that lives in this framework in a less extremal way in the "unifying observations" Section 5.5 — the band-limited Poisson channel.

Only using the peak power constraint for the Poisson channel muddies this story, since we have to imagine that aggregating the power from k fibers into one fiber boosts the peak amplitude by a factor of k. Thus, we impose total power conservation by uniformly distributing the peak amplitude budget between all of the fibers we are using. We ignore here the natural-seeming mean amplitude constraint for the sake of having a simple example of channel aggregation.

Another motivating picture is this: suppose the different fibers are being used by independent senders. Should the independent senders instead cooperate and use a

[20] The relationship to the MAC comes up again in Section 5.5.2.

single fiber? Turning many fibers into one can make the decoder more complicated[21], but does it at least help with the capacity? We return to this perspective in Section 5.5.2.

Aggregating the Poisson channel: computation

Consider the low and high "SNR" — meaning peak rate — p regimes:

$$C(p) \approx \begin{cases} \frac{1}{8}p^2 & : p \ll 1 \\ \frac{1}{e}p & : p \gg 1 \end{cases} \tag{5.45}$$

Thus for the high power regime $p \gg 1$, communication over the Poisson channel is quadratically more power-efficent than in the high SNR regime. This statement may seem counterintuitive from (5.45), but, as noted earlier, follows from the convexity of $C(p)$, so that the $p \gg 1$ regime upper bounds $C(p)$ for all p. Let's consider what happens when we aggregate k Poisson channels with SNR (peak power constraint) p and capacity C into a single Poisson channel with SNR kp and capacity $C_{\text{agg}}^{(k)}$. Since the capacities for independent channels add:

$$C_{\text{agg}}^{(k)}(p) = \frac{1}{k}C(kp) \approx \begin{cases} \frac{k}{8}p^2 & : kp \ll 1 \\ \frac{1}{e}p & : kp \gg 1 \end{cases} \tag{5.46}$$

so the aggregation boosted capacity in the low SNR regime $kp \ll 1$ and did not change it in the high SNR regime. We compare this to the case of instead splitting a a single Poisson channel into k noisier channels with SNR p/k with total capacity (sum over the subchannels) $C_{\text{split}}^{(k)}$:

$$C_{\text{split}}^{(k)}(p) = kC\left(\frac{p}{k}\right) \approx \begin{cases} \frac{1}{8k}p^2 & : p/k \ll 1 \\ \frac{1}{e}p & : p/k \gg 1 \end{cases} \tag{5.47}$$

[21] More complicated if we treat the single fiber as a multiple access channel and try to untangle the individual senders at the receiver; we can imagine a single super-sender instead.

so the splitting only hurts capacity for the Poisson channel. We can check[22] that these observations hold exactly - meaning that it is always better to aggregate than to not aggregate or to split - for the Poisson channel capacity expression (5.44):

$$\frac{1}{k}C(kp) \geq C(p) \quad \forall k \geq 1 \tag{5.48}$$

Thus for the Poisson channel, there is a large benefit to aggregation if we start with a low SNR ($p \ll 1$) channel and aggregate until the aggregated channel is in the high SNR, capacity linear with power regime. This is opposite the case of the AWGN from Section 5.2.1, where there is a benefit to splitting a chanel with high SNR ($P \gg 1$) until the noisier subchannels are in the low SNR, capacity linear with power regime.

As the number of aggregations k grows, we recover the linear scaling of capacity with peak amplitude:

$$C_{\text{agg}}^{(\infty)} = \lim_{k\to\infty} C_{\text{agg}}^{(k)} = \frac{1}{e}p \tag{5.49}$$

The capacity of the Poisson channel (5.44) and the capacity of the aggregated Poisson channel (5.46) are plotted in Figure 5.7. Note that we only gain much from aggregating the channel in the inefficient low peak amplitude (SNR), $p \ll 1$ regime.

5.5 A unifying perspective

Is there a way to unify our observations about channel splitting and aggregation so far? Let's recall what we've seen[23]:

- The capacity of the AWGN is linear in the mean power constraint P in the $P \ll 1$ regime. We can split the channel $k \gg P$ times to get there.
- The capacity of the Poisson channel is linear in the peak amplitude constraint p in the $p \gg 1$ regime. We can aggregate the channel $k \gg p$ times to get there.

[22] This follows from the convexity of $C(p)$

[23] We set the noise variance $N = 1$ for the AWGN channel and the dark count amplitude $n = 1$ for the Poisson channel.

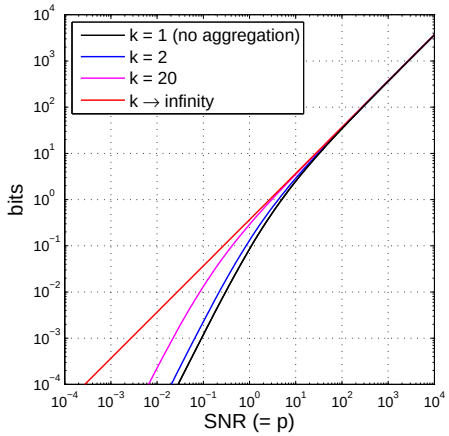

Figure 5.7: (black) capacity of the Poisson channel as a function of peak power constraint/SNR ($= p$), (blue) aggregated 2 times, (magenta) aggregated 20 times, (red) in the limit of infinitely many aggregations. Natural log.

Of course P for the AWGN and p for the Poisson mean different things, but both are constraints on the input power. Moreover the Poisson channel is in continuous time and its capacity has units of bits/time[24], while the AWGN is in discrete time. Let's hold our noses and plot both the AWGN and Poisson channel capacities on the same plot in Figure 5.8.

A feeling we might have from this cartoon is that a channel is power-efficient when its capacity is linear in the power. The AWGN is power-efficient at low power and the Poisson channel is power-efficient at high power.

[24] We used the dark count rate n to set the units of time in Section 5.4.1, so $C(p) = \frac{1}{n}C(p, n)$ has units of bits.

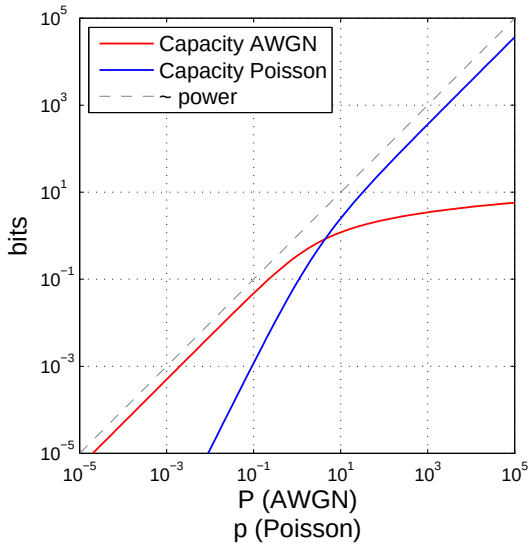

Figure 5.8: (red line) the capacity of the AWGN with SNR P (5.20). (blue line) the capacity of the Poisson channel with peak amplitude (SNR) p (5.44). The dashed line is the "bits" = "power" line.

5.5.1 The geometry of channel splitting and aggregation

Let's get at this intuition by examining the geometry of channel splitting and aggregation. Figure 5.9 shows this. Recall that

$$C_{\text{split}}^{(k)} = \frac{1}{k} C(kP) \tag{5.50}$$

$$C_{\text{agg}}^{(k)} = kC\left(\frac{P}{k}\right) \tag{5.51}$$

Here "P" denotes some kind of "power" constraint that can be split or aggregated between subchannels without reference to a particular channel. When the capacity $C(P)$ is sublinear (superlinear) in P (left (right) subplot), splitting helps (hurts) and aggregation hurts (helps).

This leads us to wonder, what if the capacity $C(P)$ is exactly tangent to the line αP for some $\alpha \geq 0$ at some point P^*? This case is shown in Figure 5.10. For reasons soon stated we call P^* a *fixed point*.

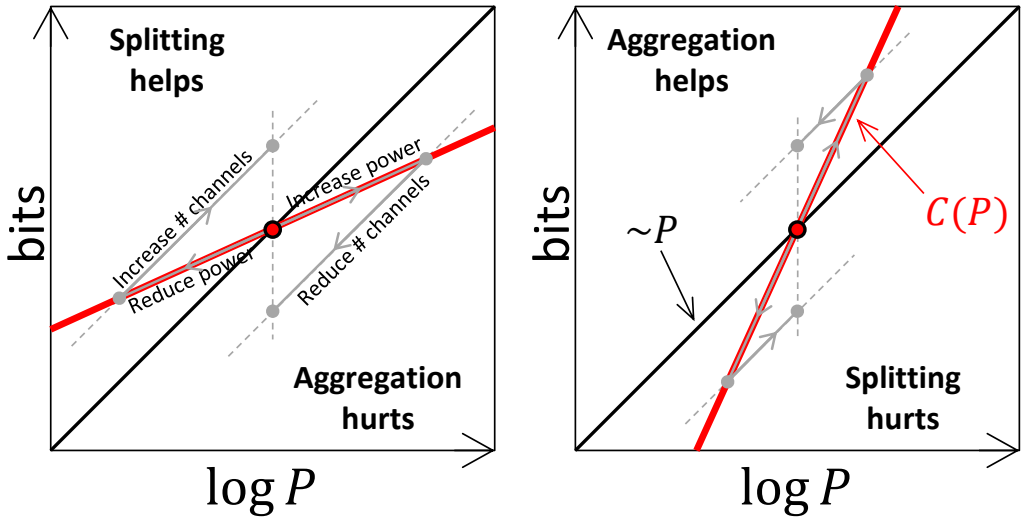

Figure 5.9: (both subplots) The capacity $C(P)$ for some channel is plotted in red. The black line is αP for some $\alpha \geq 0$. Splitting/aggregation corresponds to reducing/increasing the power P by a factor of k, and then increasing/decreasing the number of subchannels by a factor of k. (left/right) capacity sublinear/superlinear in P.

The red capacity curve $C(P)$ does not correspond to any channel that we have in mind. We find channels that exhibit this phenomenon in Section 5.5.3.

We see here that when the capacity curve $C(P)$ is tangent to αP at a fixed point P^* for some α, then for $P < P^*$ ($P > P^*$) the capacity is sublinear (superlinear) in P and so it helps to aggregate (split) the power between subchannels.

Where are all these subchannels coming from? What if we would like to do a non-integer number of splits/aggregations to get to the fixed point P^*? Let's forgo the distinction between splitting and aggregation and parametrize the transformed capacity by a parameter $\kappa > 0$:

$$C^{(\kappa)} \equiv \kappa C\left(\frac{P}{\kappa}\right) \quad \kappa > 0 \tag{5.52}$$

$\kappa > 1$ ($\kappa < 1$) corresponds to channel aggregation (splitting). We imagine perhaps having some huge number of subchannels available to use, so we can approximate the

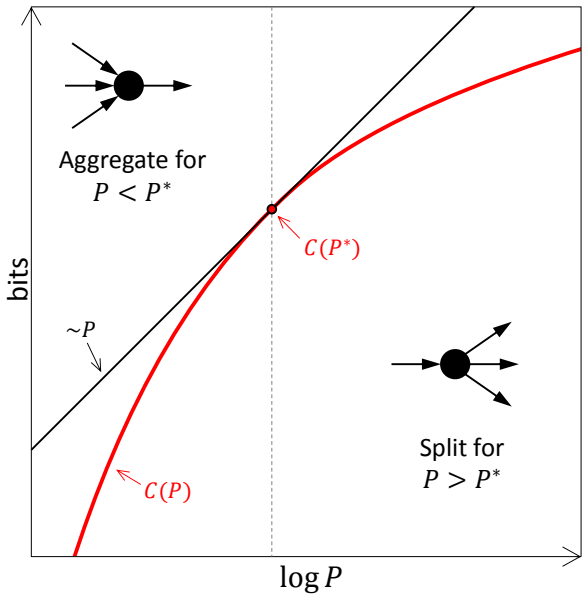

Figure 5.10: (red curve) the capacity $C(P)$ for some channel and power constraint P. (black line) αP for some $\alpha \geq 0$. The black line is tangent to the red curve at P^*. Aggregating (splitting) the power P between the subchannels boosts capacity when $P < P^*$ ($P > P^*$). P^* is the fixed point.

number of aggregations or splits by the real parameter κ or its reciprocal. We note that the transformation (5.52) is called the perspective of $C(P)$.

Let's try to find a fixed point P^* for a given capacity curve $C(P)$. Denote by $C'(P)$ the derivative[25]

$$C'(P) = \frac{d}{dP} C(P) \tag{5.53}$$

We are trying to match the tangent to $C(P)$ at P^* to αP^* for some α. At P^*, the slope is (vertical over horizontal) $\alpha = C(P^*)/P^*$. Thus we must have

$$C'(P^*) = \frac{C(P^*)}{P^*} \tag{5.54}$$

[25] Let's assume everything is well-defined to proceed.

Solving this relation for the AWGN yields $P \to 0$. Solving this for the Poisson channel yields $p \to 0$. We have seen that the Poisson channel is power-inefficient for $p \ll 1$, so what does this fixed point mean?

To answer this, let's define a dynamical system and consider the stability of its fixed points. We imagine that a benevolent engineer is either splitting or aggregating the channel power P whenever P does not satisfy the fixed point relation (5.54). The model is:

$$\frac{d}{dt}P_t = P_t C'(P_t) - C(P_t) \tag{5.55}$$

where t is "time" and $C'(P)$ is the derivative given in (5.53). Then $\frac{d}{dt}P_t = 0$ when $P_t = P^*$ is a fixed point. We evaluate the stability of the fixed points by computing the derivative of the right side of (5.55) with respect to P:

$$\frac{d}{dP}\left(PC'(P) - C(P)\right) = C'(P) + PC''(P) - C'(P) \tag{5.56}$$

$$= PC''(P) \tag{5.57}$$

so the sign of the second derivative $C''(P)$ with respect to P determines the stability of a fixed point, which seems intuitive. For the AWGN, $\lim_{P\to P^*=0} C''|_P = -\frac{1}{2} < 0$, so $P^* = 0$ is stable. For the Poisson channel, the fixed point at $p = 0$ is unstable since $\lim_{p\to p^*=0} C''|_p = \frac{1}{4} > 0$; this makes sense, since we have found that channel aggregation always increases capacity for the Poisson channel for any $p > 0$.

We shall see examples of channels with positive fixed points in Section 5.5.3, but for now we make the cartoon in Figure 5.11. The red capacity curve $C(P)$ once again does not correspond to any channel that we have in mind. Note that we are not using a logarithmic P-axis in this Figure.

This channel has at least two fixed points, identified in Figure 5.11 (maybe there is an unstable one at $P = 0$). Call them $P_1^* < P_2^*$. Using the stability criterion $C''(P) < 0$, we see that P_1^* is stable and P_2^* is unstable. Thus, for $\kappa = 1 + \epsilon$ and $|\epsilon|$ small enough, $C^{(\kappa)} > \kappa C\left(\frac{P_2^*}{\kappa}\right)$ (see (5.52)), so both channel splitting and aggregation increase capacity in the vicinity of P_2^*. Neither channel splitting and aggregation increase capacity in the vicinity of P_1^*, which is stable.

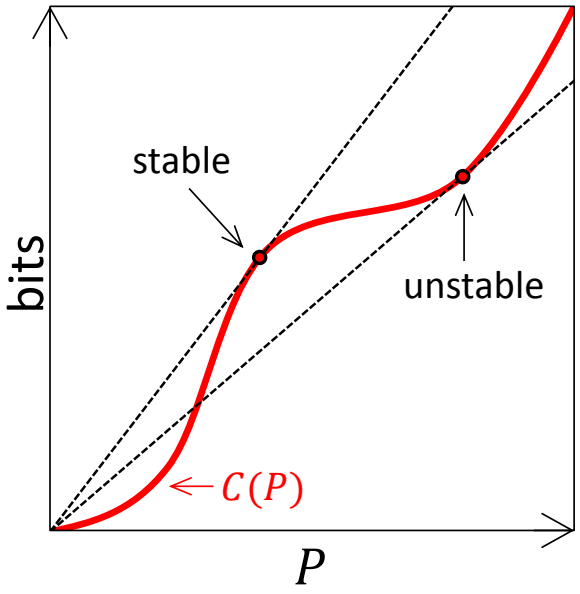

Figure 5.11: (red curve) the capacity $C(P)$ for some channel and power constraint P. Two fixed points are identified and labeled by their stability. Note that we are not using a logarithmic axis for the power P.

This perspective offers a way to decide how to split a given power budget between identical copies of a given channel with capacity $C(P)$: among the set of stable fixed points $\{P^*\}$ choose the one that maximizes the slope $C'(P^*)$. Split or aggregate power so that each subchannel is operating at this power. Thus we think of a channel operating at power P^* power-efficient when its capacity is linear in P near P^* because the freedom we've given ourselves (an arbitrary choice of κ in $C^{(\kappa)} = \kappa C(P/\kappa)$) can not improve upon capacity when capacity is linear in power. The idea of power-efficiency when capacity is linear in the power thus arises because the freedom we've given ourselves (an arbitrary choice of κ in $C^{(\kappa)} = \kappa C(P/\kappa)$) can not improve upon capacity when capacity is linear in power.

5.5.2 Multiple access and broadcast channels

The cartoon in Figure 5.10 invites a loose analogy to multiple access and broadcast channels. When the capacity is superlinear in the power parameter P, we aggregate

multiple channels into a single one to boost capacity, a bit like having a multiple access channel (MAC). When the capacity is sublinear in P, we parcel out the power to multiple channels, a bit like broadcasting. So far we have imagined only a single user and have been interested only in comparing the total capacity of all of the subchannels to the original channel, but we could also phrase things in terms of achievable rates for the individual users. The Poisson MAC was considered by (Lapidoth and Shamai, 1998), who found that the maximum throughput is linear in the number of users, unlike for the Gaussian MAC.

A rough sketch of a possible application for this analogy is this: imagine we are trying to optimize the total throughput, or something else related, of a network of nodes. The nodes talk to each other through a noisy channel with some power-like parameter P and each is supplied with a power budget. The connectivity graph for this network is determined by decisions made at the nodes: each node tries to divide its power budget between a variable number of recipients so that the each node-to-recipient link is in the power-efficient regime. Perhaps the nodes are solar-powered and on sunny days try to talk to more neighbors (rather than talk harder to their current neighbors) to maximize total throughput. So far this is a vague cartoon, but the author intends to give it thought!

5.5.3 Examples

So far we have seen that the AWGN and Poisson channel are power-efficient (capacity linear in SNR) at $P \to 0$ and $p \to \infty$, respectively. It would be fun if the perspective we developed above applied to a less extremal case. In this section we develop two such examples.

5.5.3.1 Spectrally constrained Poisson channel

We understand this example less well than the next one, but the spectrally constrained Poisson channel feels like a more natural channel, so we state it first.

What is it that makes the AWGN power-inefficient at high SNR and the Poisson channel power-efficient at high peak amplitude? For the AWGN with mean power

constraint P and noise variance $N = 1$, the number of distinguishable outputs[26] — roughly the size of the effective output alphabet - grows as $\log P$ for $P \gg 1$. For the Poisson channel with peak amplitude p and "dark" count amplitude $n = 1$, we can see that the size of the effective output alphabet is linear in P using this suboptimal scheme: Divide the total transmission time T into k chunks of size T/k and for each chunk transmit either at amplitude 1 (only the dark count amplitude) or at the maximal amplitude $p + 1$. Then we have a binary symmetric channel for each chunk with some mutual information I per chunk, so that the mutual information over time T between input and output for our scheme is $I_T = kI$. Now suppose we double the peak amplitude constraint p. Now we can afford to use twice as many chunks[27] because we need to wait half as long to resolve that the sender is sending at peak amplitude. Thus for our suboptimal scheme, the mutual information I_T is already linear in p.

So it feels like the Poisson channel in continuous time is letting us divide time more finely as p increases to boost the alphabet size, while the AWGN in discrete time is sticking to fixed time samples. What lives in between these cases?

How about a spectrally constrained Poisson channel? We imagine that the Poisson input amplitude λ_t can't vary faster than a certain bandwidth B. (Shamai, 1993) investigates the spectrally constrained Poisson channel for the case of a second moment bandwidth constraint and a strict bandwidth constraint (a constraint on the spectrum support size) and derives upper and lower bounds for the capacity in these cases. These bounds are not straightforward to apply to a Poisson channel with a peak constraint; we make the heuristic argument below, but intend more careful study.

The heuristic to infer the scaling of the capacity in the low and high bandwidth regime is:

- For $p \ll B$ and $p \ll 1$, the bandwidth constraint is irrelevant and the capacity grows as p^2 (see (5.45)).

[26] Assuming the input distribution is the capacity-achieving Gaussian distribution.
[27] Actually slightly more since the noise amplitude did not double.

- For $p \gg B$, we can no longer split the peak amplitude into $O(1/p)$ time chunks due to the bandwidth constraint; the minimum chunk time is now of order $O(1/B)$.

Thus we expect the capacity to grow superlinearly in p for $p \ll B$, $p \ll 1$ and sublinearly in p for $p \gg 1$. Thus we expect $C(p)$ to have a stable fixed point p^* for a finite p^*.

5.5.3.2 Malfunctioning channels

Here we do a more careful analysis of a particular channel whose capacity has a stable fixed point.

We imagine a channel with some kind of power constraint P that malfunctions increasingly often as P increases. Perhaps, for the Poisson channel with peak amplitude p, the photodetector "burns out" with some probability $\alpha(p)$ and loses all of the data for an entire transmission[28]. We can model this by "augmenting" our channel output alphabet with an erasure symbol e.

Suppose that the channel input (output) alphabet is $\mathcal{X}$ ($\mathcal{Y}$) and the channel matrix is $P(Y|X)$. We construct a new channel whose input is $x \in \mathcal{X}$, whose output is

$$y_{\mathrm{e}} \in \mathcal{Y} \cup \{\mathrm{e}\} \tag{5.58}$$

and whose channel matrix is

$$P(y_{\mathrm{e}}|x) = \begin{cases} \alpha & : y_{\mathrm{e}} = \mathrm{e} \\ \bar{\alpha} P(y|x) & : y_{\mathrm{e}} \neq \mathrm{e} \end{cases} \tag{5.59}$$

where $\bar{\alpha} = 1 - \alpha$ and α depends on the power constraint somehow. Then we leave it as an exercise to the reader to show that the mutual information is

$$I(X; Y_{\mathrm{e}}) = \bar{\alpha} I(X; Y) \tag{5.60}$$

[28] And then gets replaced by a working one in time for the next transmission.

where $I(X;Y)$ is the mutual information for the original non-malfunctioning channel. Thus maximizing the mutual information over the input distribution is the same for the malfunctioning and the non-malfunctioning channel, so the capacity of the malfunctioning channel is

$$C_{\mathrm{e}}(P) = (1 - \alpha(P))C(P) \tag{5.61}$$

where $C(P)$ is the capacity of the non-malfunctioning channel with power constraint P. In continuous time for a transmission of length T, we imagine that the erasure symbol e corresponds to the loss of the entire transmission.

Let's apply this erasure augmentation to the Poisson channel with peak amplitude constraint p. For our erasure model, let's assume that the probability of no malfunction is

$$1 - \alpha_\lambda(p) = e^{-\lambda p} \tag{5.62}$$

This corresponds to the case of a constant malfunction probability per unit time[29] and at least one malfunction event causing the whole transmission to be lost (causing e to be output).

Figure 5.12 plots the capacity for the malfunctioning Poisson channel (see caption for parameters). We see that there is a stable fixed point $p^* \approx 0.75$ (satisfying the relation (5.54)). Maybe counterintuitively, given a fixed power budget to spread between some number of copies of this channel, the total capacity of all of the subchannels is maximized at p^*, rather than the global maximum of the capacity $C(p)$ at $p \approx 1.60$.

[29] We should use the length T of the transmission in (5.62); we don't want to add another parameter, and so subsume T into λ.

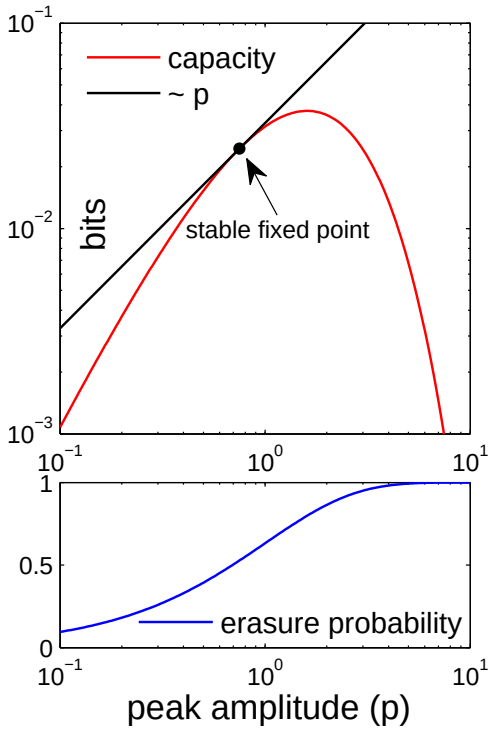

Figure 5.12: (top) (red curve) The capacity $C_{\mathrm{e}}(p)$ (5.61) for the Poisson channel with erasure probability $\alpha_\lambda(p)$ (5.62), $\lambda = 1$ tangent to the line $\frac{c(p^*)}{p^*}p$ (black line). (bottom) the erasure probability $\alpha_\lambda(p)$.

5.6 Spike trains achieve capacity for the AWGN in the low power regime

We have seen in Sections 5.2 and 5.3 that it can be worthwhile in terms of capacity[30] to split the AWGN channel with mean power constraint P and noise variance 1 (SNR P) into k noisier subchannels, each with mean power constraint P/k (SNR lower by a factor of k), and that the savings are potentially large when $P \gg k$ (so that the unpsplit channel's capacity grows only logarithmically with P) and k is at least on the order of P.

[30] and achievable rate in the finite blocklength setting.

A possible difficulty with this capacity-boosting approach is the increased complexity of the encoder and decoder required to use the noisier subchannels: Suppose the encoder is using the capacity-achieving Gaussian distribution on inputs for each of the k noisier subchannels. Then as k grows, the signal in each subchannel becomes increasingly poorly resolvable above the noise, requiring more and more work to decode — error-correct — the output, and so possibly making the channel splitting impractical. One goal of this Section is to describe a communication scheme for the AWGN that requires no error correction as the mean power constraint (SNR) goes to 0 in hopes of overcoming the encoder/decoder complexity difficulty.

The basic idea is to encode messages in the positions of a few "spikes" that rise far above the channel noise; the decoder would threshold the channel output and read off the spike positions. The spikes train input distribution is very different from the capacity-achieving Gaussian distribution, so it is not clear that this has a chance of working. In the following subsection we prove that it is possible to construct a sequence of spike distributions that comes arbitrarily close to capacity as the SNR $P \to 0$ (corresponding to an increasing number of channel splits, $k \to \infty$).

We show that the block error probability for a spike-distributed input to the AWGN vanishes as $P \to 0$ for block lengths that grow as P decreases in a certain way, suggesting the feasibility of the proposed message-to-spike communication scheme. We point out what has yet to be worked out to complete the proof of the feasibility of our proposal; the author intends to keep at it.

It would be fun to consider possible applications of these observations in considering the efficiency and complexity of naturally occurring communication schemes involving spike inputs, like neuron firing.

We note the channel dispersion of the spike train input distribution turns out to be larger than for the Gaussian input distribution, thus placing a smaller limit on the set of achievable rates in the finite block length setting, but possibly with a simpler encoder/decoder. We imagine approaching the low SNR regime by splitting a total energy budget into multiple transmissions (subchannels) and coding across the noisier subchannels (see Section 5.3.3), so our block length goes to infinity as the power per symbol goes to 0.

5.6.1 AWGN: Setup

Recall the additive white Gaussian noise (AWGN) channel with input X, output Y, and average input power constraint P:

$$Y = X + Z \tag{5.63}$$

$$Z \sim \mathcal{N}(0, 1) \tag{5.64}$$

$$\mathbb{E}[X^2] \leq P \tag{5.65}$$

Note that our noise power $\text{Var}[Z]$ is fixed to 1, so that P sets the signal to noise ratio (SNR) of our channel. The capacity $C(P)$ is given by (all logs are natural):

$$C(P) = \frac{1}{2} \log(1 + P) \tag{5.66}$$

Denote the capacity-achieving distribution for the AWGN by $p^*(x; P)$:

$$p^*(x; P) = \frac{1}{\sqrt{2\pi P}} e^{-\frac{x^2}{2P}} \tag{5.67}$$

5.6.2 A different distribution on inputs: spikes

Consider a distribution on AWGN inputs denoted by $p^{\text{d}}(x; P)$ (the 'd' is for "discrete") that has support on two distinct values[31]:

$$p^{\text{d}}(x; P) = p_1 \ \delta(x - K) + (1 - p_1) \ \delta(x) \tag{5.68}$$

[31] It may seem more natural to have a symmetric support on three values, $X \in \{-K, 0, K\}$. This would not alter the fraction of non-zero inputs computed in (5.70), and the sign of the input could be used to communicate at a greater rate. We shall see that this sign contributes a vanishingly small addition to $I(X; Y)$ as $P \to 0$ [32] so that most of the information is contained in the positions in time of the spikes.

so that $X \in \{0, K\}$ with distribution $(1 - p_1, p_1)$. To satisfy the power constraint $\mathbb{E}[X^2] = P$, we have

$$\mathbb{E}[X^2] = p_1 \cdot K^2 + (1 - p_1) \cdot 0^2 = P \tag{5.69}$$

$$\Downarrow$$

$$p_1 = \frac{P}{K^2} \tag{5.70}$$

The idea is to make the spike height K large enough to be easily resolvable in the presence of noise, so that the positions in time of the spikes carry the information. The distribution p^{d} (5.68) differs from p^* (5.67) and is thus not-capacity achieving. The goal is to choose a spike height K such that the rates achievable via the spikes input distribution come arbitrarily close to capacity in the low SNR regime as $P \to 0$. We shall see that if K grows too slowly or too quickly with the SNR P, then the spikes will either fail to be resolvable above the noise, or will be easily resolvable, but occur too rarely to carry enough information to come close to capacity.

5.6.2.1 Choice of spike height

Consider the following dependence of the spike height K on the SNR P:

$$K_{P,\epsilon} = \sqrt{-2(1+\epsilon)\log P} \quad \text{for } P < 1 \tag{5.71}$$

Here $\epsilon > 0$. We shall see below that this scaling gets us to a factor of $1/(1+\epsilon)$ of the AWGN capacity as $P \to 0$.

5.6.2.2 Output quantization

To simplify our analysis, we quantize the output of the AWGN supplied with the spikes input into two values corresponding to the presence or absence of a spike. The quantization can not help in terms of achievable rates, but turns out not to hurt in that we still come arbitrarily close to the capacity of the un-quantized AWGN as $P \to 0$.

Let $\bar{Y} \in \{0,1\}$ denote the quantization of the channel output Y, where $\bar{y} = 1$ (resp. $\bar{y} = 0$) corresponds to the presence (resp. absence) of a received spike. The quantization is accomplished by thresholding the output Y:

$$\bar{y} = \begin{cases} 1 & : y > \alpha K \\ 0 & : \text{otherwise} \end{cases} \quad \text{for } 0 < \alpha < 1 \tag{5.72}$$

where $\alpha \in (0,1)$ is the threshold for our quantizer. (We could use values of α outside this range, but then the result below would not hold.) Figure 5.6.2.2 shows the setup for the AWGN with input X with discrete distribution, quantized output $\bar{Y} \in \{0,1\}$ viewed as a discrete memoryless channel (DMC) between X and $\bar{Y}$.

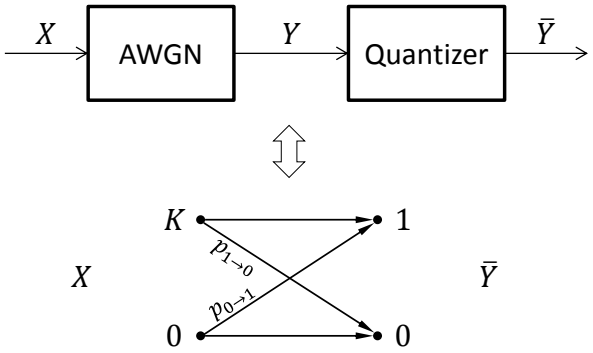

Figure 5.13: (top) AWGN channel with $X \in \{0, K\}$ with quantized output $\bar{Y}$ defined in (5.72) is equivalent to (bottom) DMC with input alphabet $\{0, K\}$, output alphabet $\{0,1\}$, crossover probabilities $p_{1\to 0}$, $p_{0\to 1}$.

Choice of quantization threshold

For the result below to hold it turns out that we need

$$\frac{1}{\sqrt{1+\epsilon}} < \alpha < 1 \tag{5.73}$$

we shall see later that a reasonable choice is

$$\alpha = \frac{1+\epsilon/2}{1+\epsilon} \tag{5.74}$$

so as a possible choice let's set $\alpha = (1+\epsilon)^{-1/4}$. Note that $\alpha \to 1^-$ (from below) as $\epsilon \to 0^+$ (from above). The quantization threshold moves close to the spike height as $\epsilon \to 0$.

Quantized DMC crossover probabilities

We compute the crossover probabilities between X discrete-distributed as p^{d} and the AWGN quantized output $\bar{Y}$:

$$p_{0\to 1} \equiv \mathbb{P}[\bar{Y}=1|X=0] = \frac{1}{2}\mathrm{erfc}\left(\frac{1}{\sqrt{2}}\alpha K_{P,\epsilon}\right) \tag{5.75}$$

$$p_{1\to 0} \equiv \mathbb{P}[\bar{Y}=0|X=K] = \frac{1}{2}\mathrm{erfc}\left(\frac{1}{\sqrt{2}}(1-\alpha) K_{P,\epsilon}\right) \tag{5.76}$$

where $\mathrm{erfc}(x) = 1-\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt$. It is useful to compute asymptotic forms for these in the $P \to 0$ ($K \to \infty$) limit:

$$p_{0\to 1} \xrightarrow{P\to 0} \frac{P^{\alpha^2(1+\epsilon)}}{2\alpha\sqrt{\pi(1+\epsilon)\log\frac{1}{P}}} \tag{5.77}$$

$$p_{1\to 0} \xrightarrow{P\to 0} \frac{P^{(1-\alpha)^2(1+\epsilon)}}{2(1-\alpha)\sqrt{\pi(1+\epsilon)\log\frac{1}{P}}} \tag{5.78}$$

$$\tag{5.79}$$

Moreover, we will later use the fact that $p_{0\to 1}$, $p_{1\to 0}$ are upper bounded by the above limiting expressions.

5.6.2.3 Summary of discrete input distribution

Table 5.1 summarizes the parameters of our discrete distribution for AWGN inputs and the quantized channel output $\bar{Y}$. Figure 5.6.2.3 shows a sample input distributed according to p^{d} for the parameters given in the caption.

$$p^{\mathrm{d}}(x; P, \epsilon) = p_1 \delta(x - K_{P,\epsilon}) + (1 - p_1)\delta(x) \tag{5.80}$$

$$p_1 = P/K_{P,\epsilon}^2 = \frac{P}{-2(1+\epsilon)\log P} \quad (\text{since } \mathbb{E}[X^2] = P\ ) \tag{5.81}$$

$$\bar{y}(y) = \mathbf{1}_{y > \alpha K_{P,\epsilon}} \tag{5.82}$$

$P < 1$	AWGN mean power constraint
$\epsilon > 0$	parameter
$K_{P,\epsilon} = \sqrt{-2(1+\epsilon)\log P}$	Spike height
$\alpha \in \left((1+\epsilon)^{-1/2}, 1\right)$	AWGN output quantization threshold as fraction of spike height $K_{P,\epsilon}$

Table 5.1: AWGN discrete input distribution $p^{\mathrm{d}}(x; P, \epsilon, \alpha)$ and output quantization parameters

5.6.3 Results about the discrete input distribution

Suppose the AWGN input X is distributed according to the discrete pmf $p^{\mathrm{d}}(x; P, \epsilon)$, $\epsilon > 0$, $P < 1$ and the channel output Y is quantized as in (5.82) for a quantization threshold $\alpha \in \left((1+\epsilon)^{-1/2}, 1\right)$, so that there is a DMC between X and the quantized output $\bar{Y}$ (c.f. Section 5.6.2.3). Then

$$\lim_{P \to 0} \frac{I(X; \bar{Y})}{C(P)} = \frac{1}{1+\epsilon} \tag{5.83}$$

where $C(P)$ is the capacity of the AWGN channel (5.66). In the low SNR regime $P \ll 1$ we can thus approximate

$$I(X; \bar{Y}) \approx \frac{P/2}{1+\epsilon} \tag{5.84}$$

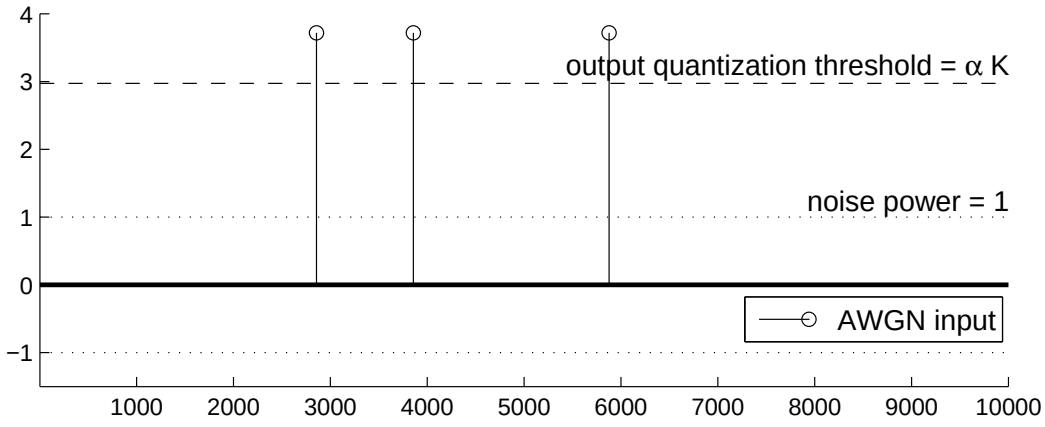

Figure 5.14: Sample spike-distributed AWGN input (circles) with parameters: mean power constraint $P = 0.01$, $\epsilon = 0.5$, spike height $K = \sqrt{-2(1+\epsilon)\log P} \approx 3.7$, quantization threshold $\alpha K = 0.8K \approx 3.0$. The crossover probabilities (5.75,5.76) are $p_{0\to 1} \approx 0.0015$, $p_{1\to 0} \approx 0.23$.

Furthermore, denoting by $\mathrm{P_e}$ the error probability for the point-wise input estimator $\hat{x} = K \cdot \mathbf{1}_{\bar{y}=1} = K \cdot \mathbf{1}_{y>\alpha K_{P,\epsilon}}$,

$$\mathrm{P_e} = \mathbb{P}[\hat{X} \neq X] \tag{5.85}$$

$$\lim_{P\to 0} \frac{\mathrm{P_e}}{p_1} = 0 \tag{5.86}$$

Furthermore,

$$\lim_{P\to 0} \frac{H(X|\bar{Y})}{H(X)} = 0 \tag{5.87}$$

$$\Updownarrow$$

$$\lim_{P\to 0} \frac{H(X)}{I(X;\bar{Y})} = 1 \tag{5.88}$$

where $p_1 = \mathbb{P}[X \neq 0]$ (5.81), so the probability of error vanishes faster than the fraction of spikes as $P \to 0$. The result (5.86) suggests that we might be able to approximate the DMC between X and $\bar{Y}$ as a noiseless DMC and use a simple coding

scheme to communicate at a rate close to $I(X;\bar{Y})$: encode the message in the locations of the spikes. We discuss this idea in the next Section.

We further compute the variance of the mutual information density $\mathrm{Var}[i(X;Y)]$ of the DMC between X and the quantized output $\bar{Y}$ for input distributed according to p^{d} in the limit $P \to 0$:

$$\lim_{P\to 0} \frac{\mathrm{Var}[i(X;Y)]}{V^{\mathrm{d}}} = 1 \tag{5.89}$$

where

$$V^{\mathrm{d}} = \frac{P \log \frac{1}{P}}{2(1+\epsilon)} \tag{5.90}$$

This quantifies the penalty[33] in achievable rate in the finite block length regime (see Section 5.3). Recalling that the channel dispersion for the AWGN in the low SNR regime $P \ll 1$ is $V = P + O(P^2)$ (see (5.32)), we see that by using the spikes distribution p^{d} we incur an extra factor that scales as $\log \frac{1}{P}$. Thus, using the spikes distribution is strictly worse than using the capacity-achieving Gaussian distribution in the finite block length setting. Moreover, the extra penalty in the finite block length regime does not go to 0 as $\epsilon \to 0$ [34], unlike the penalty of $\frac{1}{1+\epsilon}$ in the mutual information $I(X;Y)$ for $X \sim p^{\mathrm{d}}$ relative to AWGN capacity (see (5.83)).

Proofs and numerical results

The proofs of the above claims are given in Appendix 5.B. Some numerical results for the discrete input distribution are given in Appendix 5.A.

5.6.4 Coding schemes

We attempt to construct a communication scheme that uses the spikes input distribution by encoding our message in the locations of the spikes. The spikes distribution

[33] For a DMC, the channel dispersion is given by the variance of the mutual information evaluated at a capacity-achieving distribution. Here we are interested only in the spikes distribution p^{d}.

[34] The penalty factor is $\frac{\log \frac{1}{P}}{2(1+\epsilon)}$, so the penalty actually grows as $\epsilon \to 0$. Since the leading contribution to the supremum of achievable rates for $X \sim p^{\mathrm{d}}$ is $I(X;Y) \approx C_{\mathrm{AWGN}}/(1+\epsilon)$, we don't want to make ϵ too large. Perhaps there is an optimum value of ϵ to use in the finite block length regime. The author intends to give this more thought.

is efficient in the low power $P \ll 1$ regime in the sense that $I(X;\bar{Y}) \approx C(P) \approx P/2$. Thus as $P \to 0$, the achievable rate vanishes, so we need to define carefully what we want achievability to mean in this setting.

Let's define things in terms of achievable codebook sizes instead of rates; we can make up for the vanishing rate by growing the block length n as $P \to 0$. An (M, n, P) code for the quantized AWGN (see Section 5.6.2.2) has a codebook of size M, block length n, and SNR P. Instead of considering sequences of (M, n, P) codes to define achievable rates at fixed P, we consider sequences of $(M(n), n, P(n))$ codes to define achievable codebook sizes, where $P(n) \to 0$ as $n \to \infty$ [35]. Alternately, we can think of $(M(P), n(P), P)$ codes where we choose some vanishing sequence of SNRs (P_i), $P_i \to 0$ as $i \to \infty$.

We can make the block length n arbitrarily large to more than make up in codebook size for any reduction in rate as $P \to 0$, so what $n(P)$ should we use? We motivate two choices. First, a motivation for considering the $P \to 0$ limit is that we imagine having a fixed total power budget to split between an arbitrary number of transmissions. In this case we would set nP constant, so that $n \sim 1/P$.

Second, the spikes input scheme seems easy to use. Perhaps the decoding can be done symbol-wise on the output of the quantized channel. What block length should we use when using this decoding scheme? We turn to this question in the next Section.

5.6.4.1 Symbol-wise decoding (no error correction)

Consider the following coding scheme that uses the spikes input distribution. The message is encoded in the locations of the spikes. The decoder infers the locations of the spikes from the quantized channel output $\bar{Y}$. No error correction is done on the quantized output — the decoding is done symbol-wise. The lack of error correction is potentially (if this works) an appealing feature that might make this scheme practical. In contrast, for the case of the capacity-achieving Gaussian input distribution, decoding can't be done symbol-wise and an increasingly complicated decoding procedure is required as $P \to 0$ to achieve rates close to capacity.

[35] We can make it a sequence by choosing some vanishing sequence of SNRs (P_i).

When using a symbol-wise decoder (i.e. without error correction), the block error probability goes to 1 as the block length $n \to \infty$, so the block length can't grow too quickly as $P \to 0$ for the block error probability to vanish. On the other hand the block length shouldn't grow too slowly as $P \to 0$ in order to have a non-trivial[36] achievable codebook size. We show below that the following conditions need to hold in order to have vanishing block error probability and non-trivial codebook size.

Suppose the block length $n(P)$ satisfies[37] for $P \to 0$

$$n(P) = \Omega\left(\frac{\log(1/P)}{P}\right) \tag{5.91}$$

$$n(P) = o\left(\frac{\log(1/P)^{3/2}}{P^{1+(1+\epsilon)(1-\alpha)^2}}\right) \tag{5.92}$$

$$n(P) = o\left(\frac{\log(1/P)^{1/2}}{P^{(1+\epsilon)\alpha^2}}\right) \tag{5.93}$$

where the channel input X^n is drawn point-wise from the spikes distribution p^{d}, and the decoder uses the point-wise estimator $\hat{x}_i = K \cdot \mathbf{1}_{y_i > \alpha K_{P,\epsilon}}$. Note that the upper bounds never exclude the lower bound since $\alpha \in \left((1+\epsilon)^{-1/2}, 1\right)$. If we use the choice (5.74) $\alpha = (1+\epsilon/2)/(1+\epsilon)$, then the upper bounds (5.92,5.93) become the single upper bound

$$n(P) = o\left(\frac{\log(1/P)^{1/2}}{P^{\frac{(1+\epsilon/2)^2}{1+\epsilon}}}\right) \tag{5.94}$$

Then the block error probability vanishes as $P \to 0$, $n(P) \to \infty$:

$$\lim_{P\to 0} \mathbb{P}\left[\hat{X}^{n(P)} \neq X^{n(P)}\right] = 0 \tag{5.95}$$

This is the sense in which we do not need error correction for this communication scheme. For $n(P)$ satisfying these assumptions, there exists a sequence[38] of $(M(P), n(P), P)$ codes with vanishing error probability and non-trivial codebook size

[36] Meaning $M = 1$ since we can always put at least one codeword in our codebook.

[37] For example, $n(P) = \left[P^{-\frac{1+\epsilon/2}{\sqrt{1+\epsilon}}}\right]$ works (satisfies (5.91) and (5.94)), where [] denotes the integer part.

[38] We choose some vanishing sequence of SNRs (P_i), $P_i \to 0$ as $i \to \infty$. We do this, rather than let n be the index and $P(n)$ vanish, because it's easier to work with $n(P)$ than $P(n)$.

$M(P)$ as $P \to 0$. If $n(P)$ exceeds the upper bounds (5.92, 5.93) and the block error probability vanishes, then $M(P) \to 1$, the trivial codebook.

We would like to use instead $n \sim 1/P$ in order to conserve total power nP, but this doesn't work when we use the point-wise decoder; we need the extra ϵ-dependent scaling in (5.91) for this result. We are threading the needle here in choosing a block length so that the error-correctionless scheme is asymptotically feasible, but at least we know it is possible. As $\epsilon \to 0$

Let's derive the conditions (5.91, 5.92, 5.93). For the lower bound, we want a block length large enough so that the expected number of spikes per n channel uses does not vanish; otherwise the probability of having at least one spike vanishes, and the achievable codebook is trivial as $P \to 0$, $n(P) \to \infty$. The expected number of spikes is

$$\mathbb{E}[\#\text{ spikes}] = n(P)\ p_1 = n(P)\frac{P}{2(1+\epsilon)\log\frac{1}{P}} \tag{5.96}$$

For this not to vanish as $P \to \infty$, we choose $n(P) = \Omega\left(\frac{\log(1/P)}{P}\right)$ [39].

For the upper bound (5.94), we want a block length small enough that the expected number of symbol-wise errors vanishes as $P \to 0$, $n(P) \to \infty$, so that the block error probability vanishes. We computed an upper bound for the symbol-wise error probability (5.111):

$$\mathrm{P_e} = \mathbb{P}[\hat{X} \neq X] = p_1 p_{1\to 0} + (1-p_1)p_{0\to 1} \tag{5.97}$$

$$\leq c_{10}(\epsilon,\alpha)\ P^{1+(1+\epsilon)(1-\alpha)^2}\log\left(\frac{1}{P}\right)^{-3/2} + c_{01}\ (\epsilon,\alpha)P^{(1+\epsilon)\alpha^2}\log\left(\frac{1}{P}\right)^{-1/2} \tag{5.98}$$

$$\overset{(a)}{=} P^{\frac{(1+\epsilon/2)^2}{1+\epsilon}}\left(c_{10}(\epsilon)\ \log^{-3/2}\frac{1}{P} + c_{01}\ (\epsilon)\log^{-1/2}\frac{1}{P}\right) \tag{5.99}$$

where c_{10} and c_{01} depend only on ϵ and α and in (a) we have chosen $\alpha = (1+\epsilon/2)/(1+\epsilon)$ so that the two terms in (5.98) vanish at the same rate $P^{\frac{(1+\epsilon/2)^2}{1+\epsilon}}$. Then we can

[39] We could have a codebook size logarithmic in $n(P)$ by choosing $n(P)$ so that there is one spike on average. For the lower bound (5.91), the codebook size grows faster than logarithmically. In this Section, we only try to show that the codebook size is nontrivial.

upper bound the block error probability

$$\mathbb{P}\left[\hat{X}^{n(P)} \neq X^{n(P)}\right] \leq n(P)\ \mathrm{P_e} \tag{5.100}$$

For this to vanish, we choose $n(P) = o\left(P^{-\frac{(1+\epsilon/2)^2}{1+\epsilon}} \log\left(\frac{1}{P}\right)^{1/2}\right)$ [40].

5.6.4.2 With error correction

We might still be able to use the block length $n \sim 1/P$ (to conserve total power nP) if instead of using a point-wise decoder as in the previous Section, we simply use some decoder (i.e., use error correction). How well can we do? I don't know, but I can at least state what I would like to know.

As we discussed in Section 5.6.4, as $P \to 0$, the rate vanishes, so we instead consider achievable codebook sizes $M(n, P, \beta)$ where $n \sim 1/P$ and $\beta^{(n)}$ is the block error probability. A codebook size $M(\beta)$ is called achievable with block error probability β if a sequence of $(M, n(P), P)$ codes exists such that $\sup_n \beta^{(n)} \leq \beta$ as $n \sim \frac{1}{P} \to \infty$. We do not know if nontrivial codebook sizes are achievable with block error probability $\beta = 0$.

Let's define

$$P_{\mathrm{T}} = nP \tag{5.101}$$

as the total energy budget (constant since $n \sim 1/P$). Now we seek the supremum of achievable codebook sizes:

$$l^*(P_{\mathrm{T}}, \beta) \equiv \sup \log M(\beta) \tag{5.102}$$

where the supremum is taken over all schemes that satisfy the total energy constraint P_{T} [41] and for which $\beta^{(n)}$ is upper bounded by β as $n \to \infty$. The quantity $l^*(P_{\mathrm{T}}, \beta)$ has units of bits per unit energy (rather than per unit time). We can interpret it

[40] Had we chosen some other value of $\alpha \in \left((1+\epsilon)^{-1/2}, 1\right)$, we would get some other upper bound; hence the two upper bounds (5.92,5.92), one for each term in the expression (5.98) for the symbol-wise error probability.

[41] That is, the input x^n for any such scheme satisfies $\sum_{i=1}^n x_i \leq P_{\mathrm{T}}$.

as the maximum number of bits that can be sent using a fixed amount of energy P_{T} with given block error probability β.

Our setting is very similar to the capacity-per-unit-cost setting, where one seeks to communicate near the capacity-cost function $C(P)/P$ (for an introduction to the topic, see (Verdú, 1990)). For the AWGN with mean power constraint, (Golay, 1949) observed that the minimum energy per bit for $P \ll 1$ is $1/2$ (we set the noise variance to $N = 1$), so we expect $l^*(P_{\mathrm{T}}) \approx P_{\mathrm{T}}/2$ for $P_{\mathrm{T}} \gg 1$. Moreover, (Golay, 1949) used a pulse position modulation scheme that involved using all of the power budget on a single transmission communicating $\log_2 N$ bits, similar to our spikes distribution (we do not fix the number of spikes per transmission).

Does a scheme that uses the spikes distribution for inputs $X \sim p^{\mathrm{d}}$ come arbitrarily close (as the parameter $\epsilon \to 0$ after the limit $n \to \infty$) to achieving l^*? We expect that when using $X \sim p^{\mathrm{d}}$ with a fixed energy budget $P_{\mathrm{T}} \gg 1$ and block error probability β, there is some optimal block length $n^*(P_{\mathrm{T}})$ that maximizes the achievable codebook: if the block length is too small, we find ourselves in the inefficient high-SNR regime for the AWGN. If the block length is too large, then there are no spikes at all[42], yielding a trivial codebook. Equivalently we can ask, how does the codebook-size optimal number of spikes per transmission grow with the total energy budget? Does it grow at all, or is the single-pulse scheme of (Golay, 1949) already do as well as we can hope?

5.A Numerical results for discrete input distribution for the AWGN

Figure 5.A (1) shows $I(X;\bar{Y})/C(P)$ for different values of the SNR P vs. ϵ. The AWGN output quantization threshold α (5.82) for these plots is

$$\alpha = \frac{1+\epsilon/2}{1+\epsilon} \tag{5.103}$$

[42] The spike probability is $p_1 = \frac{P}{2(1+\epsilon)\log\frac{1}{P}}$, so $\mathbb{E}[\#\text{ spikes}] = p_1 n \sim p_1/P = O\left(\frac{1}{\log\frac{1}{P}}\right) \to 0$ as $n \sim 1/P \to \infty$.

derived in (5.113) to minimize the error probability $\mathrm{P_e}$ (5.105) as $P \to 0$.

Figure 5.A (2) shows other functions of the discrete AWGN input distribution $p^{\mathrm{d}}(x; P, \epsilon)$ for different values of SNR P and ϵ. See the caption for this figure for discussion.

5.B Proofs of results about the discrete input distribution for the AWGN

The program is to use Fano's inequality to show that $H(X|\bar{Y})/H(X) \to 0$ as $P \to 0$, so that $I(X;\bar{Y}) \approx H(X)$ as $P \to 0$, and then to compute $H(X)$ in the low power limit and show that it matches the capacity of the AWGN channel up to a factor of $1/(1+\epsilon)$.

5.B.1 Upper bound on the error probability

To apply Fano's inequality, let's compute an upper bound for the error probability $\mathrm{P_e}$. Our estimator $\hat{X}$ for X given $\bar{Y}$ is

$$\hat{x} = K_{P,\epsilon} \cdot \mathbf{1}_{\bar{y}=1} \tag{5.104}$$

where $K_{P,\epsilon}$ is given in (5.71). Then

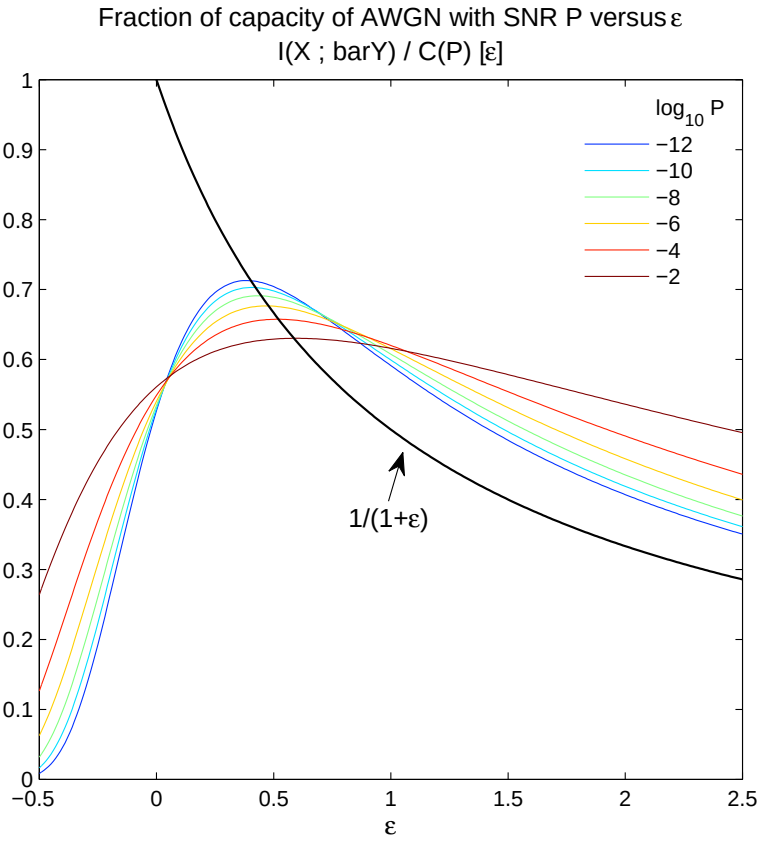

Figure 5.15: $I(X;\bar{Y})/C(P)$ (fraction of capacity of AWGN), where X is distributed according to a discrete input distribution and $\bar{Y}$ is the quantized channel output (c.f. Section 5.6.2.3). The SNR P is held fixed for each curve and ϵ is varied (recall that the spike height is $K_{P,\epsilon} = \sqrt{-2(1+\epsilon)\log P}$). The quantization threshold is $\alpha = (1+\epsilon/2)/(1+\epsilon)$ (5.113). The thick black line $1/(1+\epsilon)$ is the limiting curve for all $\epsilon > 0$ as $P \to 0$.

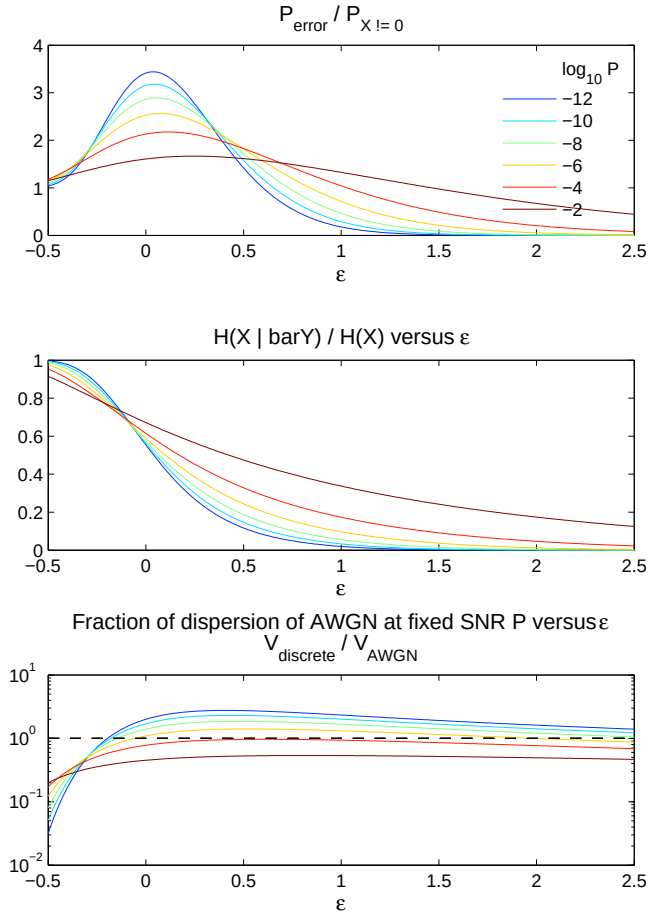

Figure 5.16: **(top)** $\mathrm{P_e}/p_1$ at fixed SNR P (colors) vs. ϵ, where the error probability $\mathrm{P_e}$ and $p_1 = \mathbb{P}[X \neq 0]$ are as given in (5.85,5.81), respectively. This ratio vanishes as $P \to 0$ for all $\epsilon > 0$ (5.86). **(middle)** $H(X|\bar{Y})/H(X)$ (same colors and SNR P as above). This ratio vanishes as $P \to 0$ for all $\epsilon > 0$ (5.87). **(bottom)** The ratio of the dispersion of the AWGN with the discrete input distribution $p^{\mathrm{d}}(x; P, \epsilon)$ to the dispersion of the AWGN using the (Gaussian) capacity-achieving distribution $p^*(x; P)$ (5.67) (same colors and SNR P as above). The former dispersion is computed as $\mathrm{Var}[i(x; \bar{y})]$ for the DMC between X and $\bar{Y}$ (Strassen, 1962; Polyanskiy et al., 2010), the variance of the information density between x and $\bar{y}$. The latter dispersion is given by $V(P) = \frac{1}{2}P(P+2)/(P+1)^2$ (Polyanskiy et al., 2009). This ratio scales as (5.90) $\frac{\log \frac{1}{P}}{2(1+\epsilon)}$ as $P \to 0$. Natural log.

$$\mathrm{P_e} = \mathbb{P}[\hat{X} \neq X] \tag{5.105}$$

$$= \mathbb{P}[X = K, \bar{Y} = 0] + \mathbb{P}[X = 0, \bar{Y} = 1] \tag{5.106}$$

$$\overset{(a)}{=} p_1 p_{1\to 0} + (1 - p_1) p_{0\to 1} \tag{5.107}$$

$$\leq p_1 p_{1\to 0} + p_{0\to 1} \tag{5.108}$$

$$\overset{(b)}{=} \left(\frac{P}{K_{P,\epsilon}^2}\right)\left(\frac{1}{2}\mathrm{erfc}\left(\frac{1}{\sqrt{2}}(1-\alpha)K_{P,\epsilon}\right)\right) + \left(\frac{1}{2}\mathrm{erfc}\left(\frac{1}{\sqrt{2}}\alpha K_{P,\epsilon}\right)\right) \tag{5.109}$$

$$\overset{(c)}{\leq} \left(\frac{P}{K_{P,\epsilon}^2}\right)\left(\frac{e^{-\left((1-\alpha)K_{P,\epsilon}\right)^2/2}}{(1-\alpha)K_{P,\epsilon}\sqrt{2\pi}}\right) + \left(\frac{e^{-\left(\alpha K_{P,\epsilon}\right)^2/2}}{\alpha K_{P,\epsilon}\sqrt{2\pi}}\right) \tag{5.110}$$

$$\overset{(d)}{=} \frac{(1+\epsilon)^{-3/2}}{4\sqrt{\pi}(1-\alpha)} \cdot \left(\log\frac{1}{P}\right)^{-3/2} \cdot \boxed{P^{1+(1+\epsilon)(1-\alpha)^2}} + \frac{(1+\epsilon)^{-1/2}}{2\sqrt{\pi}\alpha} \cdot \left(\log\frac{1}{P}\right)^{-1/2} \cdot \boxed{P^{(1+\epsilon)\alpha^2}} \tag{5.111}$$

where in (a) $p_1 = \mathbb{P}[X \neq 0]$ (computed in (5.81)) and $p_{1\to 0}$ and $p_{0\to 1}$ are the crossover probabilities computed in Section 5.6.2.2. In (b) we substituted the values of p_1, $p_{1\to 0}$, and $p_{0\to 1}$ (5.81,5.75,5.76) for the discrete input distribution. In (c) we used an approximation for $\mathrm{erfc}(x) \leq e^{-x^2}/(x\sqrt{\pi})$ (an approximation for $x \gg 1$, an upper bound for $x > 0$). In (d) we substituted our choice of the spike height $K_{P,\epsilon}$ (5.71).

The boxed terms in (5.111) dominate the behavior of the error probability $\mathrm{P_e}$ for $P \ll 1$, since the other terms are constant or logarithmic in the power P (at fixed ϵ and α). Note that since $\epsilon > 0$ and $0 < \alpha < 1$, the error probability vanishes as $P \to 0$, yielding (5.86) (since p_1, the probability of a spike input, vanishes as well, it is not clear that this input distribution is useful).

We might want for the two contributions to the error probability $\mathrm{P_e}$ to have the same scaling with the power P to minimize $\mathrm{P_e}$:

$$P^{1+(1+\epsilon)(1-\alpha)^2} = P^{(1+\epsilon)\alpha^2} \tag{5.112}$$

$$\Downarrow$$

$$\alpha = \frac{1+\epsilon/2}{1+\epsilon} \tag{5.113}$$

This is the motivation for the choice of α given in Section 5.6.2.2 and (5.74).

5.B.2 Some lemmas

We shall need the following lemmas.

Lemma 1: Given two functions $q : \mathbb{R}^+ \to \mathbb{R}^+$, $p : \mathbb{R}^+ \to \mathbb{R}^+$, suppose that

$$\lim_{x\to 0^+} q(x) = \lim_{x\to 0^+} \frac{q(x)}{p(x)} = 0 \tag{5.114}$$

Then

$$\lim_{x\to 0^+} \frac{H(q(x))}{H(p(x))} = 0 \tag{5.115}$$

where $H(x) = -x\log x - (1-x)\log(1-x)$ is the binary entropy function. Note that this lemma only takes some work to prove in the case that $\lim_{x\to 0^+} p(x) = 0$.

Lemma 1.1: Given $x > 0$, $y > 0$ and x and y small enough, the following identity holds:

$$H(xy) < H(x)H(y) \tag{5.116}$$

It is sufficient to have $x, y < x^*$, where $x^* \approx 0.416$ is the unique solution of $H(x^2) = H(x)^2$ on the open interval $x \in (0, 1)$.

Proof of Lemma 1.1: Suppose that $x < 1/2$ and suppose without further loss of generality that $y = \alpha x$, $\alpha \in (0, 1]$. Then

$$\frac{H(xy)}{H(x)H(y)} = \frac{H(\alpha x^2)}{H(x)H(\alpha x)} \overset{\text{(a)}}{\leq} \frac{H(\alpha x^2)}{H(\alpha x)^2} \overset{\text{(b)}}{\leq} \frac{H(\alpha x^2)}{\alpha^2 x^2 \log^2(\alpha x)} \tag{5.117}$$

where (a) follows from the monotonicity of $H(x)$ for $x \in [0, 1/2]$ and (b) follows from keeping only one of the two terms in the binary entropy function. Now because

$$\lim_{x\to 0} \frac{(1-x)\log(1-x)}{x\log x} = 0 \tag{5.118}$$

then for small enough x, we can crudely upper bound the binary entropy function by twice the term:

$$H(x) \leq -2x\log x \quad \text{for } x \text{ small enough} \tag{5.119}$$

Substituting this into (5.117) we obtain

$$\frac{H(xy)}{H(x)H(y)} \leq -2\frac{\alpha x^2 \log(\alpha x^2)}{\alpha^2 x^2 \log^2(\alpha x)} \quad \text{for } x \text{ small enough} \tag{5.120}$$

$$= -\frac{4}{\alpha}\frac{\log(\alpha x)}{\log^2(\alpha x)} \tag{5.121}$$

$$\to 0 \text{ as } x \to 0 \tag{5.122}$$

Thus for x small enough and for $y \leq x$ we have $H(xy) < H(x)H(y)$. Switching x and y completes the proof. □ We looked at a plot to check that it is sufficient to have $x, y < x^*$, where $x^* \approx 0.416$ is the unique solution of $H(x^2) = H(x)^2$ for $x \in (0, 1)$.

Proof of Lemma 1: Let's define

$$r(x) \equiv \frac{q(x)}{p(x)} \tag{5.123}$$

Then

$$\frac{H(q(x))}{H(p(x))} = \frac{H(r(x)p(x))}{H(p(x))} \tag{5.124}$$

$$\overset{(a)}{<} \frac{H(r(x))H(p(x))}{H(p(x))} \quad \text{for } x \text{ small enough} \tag{5.125}$$

$$= H(r(x)) \quad \text{for } x \text{ small enough} \tag{5.126}$$

where in (a) we used the assumptions (5.114) to find x small enough to apply Lemma 1.1. $r(x)$ vanishes as $x \to 0^+$ by the assumptions (5.114), so Lemma 1 follows. □

Lemma 1 will be useful when q and p are as given in Lemma 2:

Lemma 2: For the discrete input distribution $p^{\mathrm{d}}(x; P, \epsilon)$ for the AWGN channel, let the probability of error $\mathrm{P_e}$ and $p_1 = \mathbb{E}[X \neq 0]$ be as given in (5.105, 5.81), respectively. Furthermore, let the output quantization threshold be $\alpha \in \left(\frac{1}{\sqrt{1+\epsilon}}, 1\right)$. Then

$$\lim_{P \to 0} \frac{\mathrm{P_e}}{p_1} = 0 \tag{5.127}$$

Proof of Lemma 2:

$$\frac{\mathrm{P_e}}{p_1} \overset{(a)}{=} \left(\frac{2(1+\epsilon)\log\frac{1}{P}}{P}\right)\mathrm{P_e} \tag{5.128}$$

$$\overset{(b)}{\leq} \frac{(1+\epsilon)^{-1/2}}{2\sqrt{\pi}(1-\alpha)} \cdot \left(\log\frac{1}{P}\right)^{-1/2} \cdot \boxed{P^{(1+\epsilon)(1-\alpha)^2}} + \frac{(1+\epsilon)^{1/2}}{\sqrt{\pi}\alpha} \cdot \left(\log\frac{1}{P}\right)^{1/2} \cdot \boxed{P^{(1+\epsilon)\alpha^2-1}} \tag{5.129}$$

In (a) we substituted (5.81). In (b) we we used the inequality (5.111). As in (5.111), the boxed terms dominate the behavior of the expression as $P \to 0$. The first boxed term vanishes as $P \to 0$ with ϵ and α fixed by the assumptions $\epsilon > 0$, $0 < \alpha < 1$. The second boxed term vanishes as $P \to 0$ when

$$(1+\epsilon)\alpha^2 - 1 > 0 \Rightarrow \alpha > \frac{1}{\sqrt{1+\epsilon}} \tag{5.130}$$

Since $\alpha < 1$ by assumption (otherwise the quantization threshold would be greater than or equal to the spike height) we find

$$\alpha \in \left(\frac{1}{\sqrt{1+\epsilon}}, 1\right) \tag{5.131}$$

This is the condition on α we stated without motivation in Section 5.6.2.2. This completes the proof of Lemma 2. □

Let's put these lemmas together to obtain Lemma 3

Lemma 3: For $\mathrm{P_e}$ and p_1 as in Lemma 2,

$$\lim_{P\to 0} \frac{H(\mathrm{P_e})}{H(p_1)} = 0 \tag{5.132}$$

Proof of Lemma 3: Apply Lemma 1 with $q = \mathrm{P_e}$, $p = p_1$. The conditions of Lemma 1 (5.114) follow from Lemma 2. □

We shall also use apply Fano's inequality as stated in Lemma 4.

Lemma 4 (Fano's inequality): Let X have the discrete distribution $p^{\mathrm{d}}(x; P, \epsilon)$ and $\bar{Y}$ be the quantized AWGN output defined in (5.72). Then

$$H(X|\bar{Y}) \leq H(\mathrm{P_e}) \tag{5.133}$$

where $\mathrm{P_e}$ is the error probability defined in (5.105).

Proof of Lemma 4: We use the form of Fano's inequality given in Cover & Thomas (2.140):

$$H(X|\bar{Y}) \leq H(\mathrm{P_e}) + \mathrm{P_e} \log\left(|\mathcal{X}| - 1\right) \tag{5.134}$$

$$= H(\mathrm{P_e}) \tag{5.135}$$

where the equality follows from input alphabet $\mathcal{X} = \{0, 1\}$, $|\mathcal{X}| = 2$. □

5.B.3 Proofs of main results

Let's apply these lemmas to derive the main results stated in Section 5.6.3, starting with the limits (5.87,5.88)

Proof of (5.86): This follows from Lemma 2 since $p_1 \in [0, 1]$. □

Proof of (5.87,5.88):

We write

$$\frac{H(X|\bar{Y})}{H(X)} \overset{\text{(a)}}{\leq} \frac{H(\mathrm{P_e})}{H(X)} \tag{5.136}$$

$$\overset{\text{(b)}}{=} \frac{H(\mathrm{P_e})}{H(p_1)} \tag{5.137}$$

$$\overset{\text{(c)}}{\to} 0 \text{ as } P \to 0 \tag{5.138}$$

where (a) follows from Lemma 4 (Fano's inequality), (b) follows since X is distributed according to a discrete distribution with input alphabet $\mathcal{X} = \{0, 1\}$, and (c) follows from Lemma 3. Thus (5.87) follows. (5.88) is equivalent to (5.87), writing $I(X; \bar{Y}) = H(X) - H(X|\bar{Y})$ and rearranging the expression. □

Finally, let's prove the result (5.83).

Proof of (5.83): First, let's compute $H(X)$ in the low power limit:

$$\frac{H(X)}{C(P)} \overset{(a)}{=} \frac{H(p_1)}{\frac{1}{2}\log(1+P)} \tag{5.139}$$

$$\overset{(b)}{=} \frac{H\left(\frac{P}{-2(1+\epsilon)\log P}\right)}{\frac{1}{2}\log(1+P)} \tag{5.140}$$

$$\overset{(c)}{=} \frac{\log\frac{1}{P} + \log\log\frac{1}{P} + 1 + \log(2(1+\epsilon))}{(1+\epsilon)\log\frac{1}{P}} + O(P) \tag{5.141}$$

$$= \frac{1}{1+\epsilon} + O\left(\frac{\log\log\frac{1}{P}}{\log\frac{1}{P}}\right) \tag{5.142}$$

$$\to \frac{1}{1+\epsilon} \text{ as } P \to 0 \tag{5.143}$$

where (a) follows from the discrete distribution of X with support size of 2. In (b) we substituted the value of $p_1 = \mathbb{P}[X \neq 0]$ (5.81). In (c) we computed the power series about $P = 0$. Note that our $O(f(P))$ notation corresponds to the case $P \to 0$, rather than $P \to \infty$.

Finally, we derive (5.83)

$$\lim_{P\to 0} \frac{I(X;\bar{Y})}{C(P)} \overset{(a)}{=} \lim_{P\to 0} \frac{H(X)}{C(P)} \tag{5.144}$$

$$\overset{(b)}{=} \frac{1}{1+\epsilon} \tag{5.145}$$

where (a) follows from (5.88) and (b) follows from (5.143).

where in (a) $p_1 = \mathbb{P}[X \neq 0]$ (computed in (5.81)) and $p_{1\to 0}$ and $p_{0\to 1}$ are the crossover probabilities computed in Section 5.6.2.2. In (b) we substituted the values of p_1, $p_{1\to 0}$, and $p_{0\to 1}$ (5.81,5.75,5.76) for the discrete input distribution. In (c) we used an approximation for $\text{erfc}(x) \leq e^{-x^2}/(x\sqrt{\pi})$ (an approximation for $x \gg 1$, an upper bound for $x > 0$). In (d) we substituted our choice of the spike height $K_{P,\epsilon}$ (5.71). □

Let's derive the limiting expression for the variance of the mutual information density, $\text{Var}[i(X;Y)]$ of the DMC between $X \sim p^{\text{d}}$ and $\bar{Y}$ in the limit $P \to 0$. Recall

from the discussion in Section 5.3.1 that this quantifies the penalty in the supremum of achievable rates in the finite block length setting.

Proof of (5.95): Using sloppy methods we guess that the limiting expression for $\mathrm{Var}[i(X;\bar{Y})]$ is as given in (5.89)

$$V^{\mathrm{d}} = \frac{P \log \frac{1}{P}}{2(1+\epsilon)} \tag{5.146}$$

Let's check that it works. First, let's compute $\mathbb{E}[i(X;\bar{Y})^2]$:

$$\mathbb{E}[i(X;\bar{Y})^2] = \sum_{x,\bar{y}} p(\bar{y}|x)p(x) \log^2 \left(\frac{p(\bar{y}|x)}{p(x)} \right) \tag{5.147}$$

Let's recall the asymptotic forms for the DMC crossover probabilities $p_{1\to 0}$, $p_{0\to 1}$ in the limit $P \to 0$ (5.77,5.78):

$$p_{0\to 1} = O\left(\frac{P^{\alpha^2(1+\epsilon)}}{\sqrt{\log \frac{1}{P}}} \right) \tag{5.148}$$

$$p_{1\to 0} = O\left(\frac{P^{(1-\alpha)^2(1+\epsilon)}}{\sqrt{\log \frac{1}{P}}} \right) \tag{5.149}$$

and for the spike probability $p_1 = p(x = K)$ (5.81):

$$p_1 = \frac{P}{2(1+\epsilon)\log \frac{1}{P}} \tag{5.150}$$

Thus all of the $\log^2$ terms in the expression (5.147) — except for the one corresponding to $x = \bar{y} = 0$ — are $\sim \log^2 P^\gamma$ for some γ, so to establish which terms of (5.147) are dominant we must look at the prefactors $p(\bar{y}|x)p(x)$. Let's first consider the cross

terms $(x = 0, \bar{y} = 1)$ and $(x = K, \bar{y} = 0)$:

$$\frac{p_{0\to1}p_0}{V^{\mathrm{d}}} = O\left(P^{\alpha^2(1+\epsilon)-1}\log^{\beta_{01}}\left(\frac{1}{P}\right)\right) \tag{5.151}$$

$$\frac{p_{1\to0}p_1}{V^{\mathrm{d}}} = O\left(P^{(1-\alpha)^2(1+\epsilon)}\log^{\beta_{10}}\left(\frac{1}{P}\right)\right) \tag{5.152}$$

$$\frac{p_{0\to0}p_0}{V^{\mathrm{d}}} = O\left(P^{-1}\log^{\beta_{00}}\left(\frac{1}{P}\right)\right) \tag{5.153}$$

where β_{01}, $\beta 10$ are constants. We assumed that $\alpha \in \left((1+\epsilon)^{-1/2}, 1\right)$ in the construction of the spikes distribution, so both of the expressions above tend to 0 as $P \to 0$. We can approximate the $x = \bar{y} = 0$ term by expanding the $\log^2$ for small P:

$$\log^2\left(\frac{p_{0\to0}}{p_0}\right) = \left(\frac{1-p_{0\to1}}{1-p_1}\right) = O\left(P^{\alpha^2(1+\epsilon)}\log^{-1/2}\frac{1}{P}\right) \tag{5.154}$$

Therefore the ratio of this term to V^{d} vanishes as $P \to 0$:

$$\frac{p_{0\to0}p_0\log^2\left(\frac{p_{0\to0}}{p_0}\right)}{V^{\mathrm{d}}} = O\left(P^{\alpha^2(1+\epsilon)-1}\log^{-1/2}\frac{1}{P}\right) \xrightarrow{P\to0} 0 \tag{5.155}$$

(where we again used $\alpha > (1+\epsilon)^{-1/2}$) The only remaining term corresponds to $(x = K, \bar{y} = 1)$. Let's compute its ratio to V^{d} as $P \to 0$:

$$\lim_{P\to0}\frac{p_{1\to1}p_1\log^2\left(\frac{p_{1\to1}}{p_1}\right)}{V^{\mathrm{d}}} \overset{\text{(a)}}{=} \lim_{P\to0}\frac{p_{1\to1}\log^2\left(p_{1\to1}\frac{2(1+\epsilon)\log\frac{1}{P}}{P}\right)}{\log^2\frac{1}{P}} \tag{5.156}$$

$$\overset{\text{(b)}}{=} \lim_{P\to0}\frac{\log^2\left(\frac{\log\frac{1}{P}}{P}\right)}{\log^2\frac{1}{P}} \tag{5.157}$$

$$= 1 \tag{5.158}$$

In (a) we substituted p_1 (5.150) and V^{d} (5.146). In (b) we used $p_{1\to1} \xrightarrow{P\to0} 1$ and expanded the $\log^2$ term. Thus we have found

$$\lim_{P\to0}\frac{\mathbb{E}[i(X;\bar{Y})^2]}{V^{\mathrm{d}}} = 1 \tag{5.159}$$

So far we have only said something about $\mathbb{E}[i(X;\bar{Y})^2]$. To compute the variance, we compute $\mathbb{E}[i(X;\bar{Y})]^2 = I(X;\bar{Y})^2$. We already found (5.84) that $I(X;\bar{Y}) \approx (P/2)/(1+\epsilon)$ for $P \ll 1$. Thus $I(X;\bar{Y})^2 = O(P^2)$, so

$$\frac{I(X;\bar{Y})^2}{V^{\mathrm{d}}} = O\left(P \log \frac{1}{P}\right) \tag{5.160}$$

and the ratio to V^{d} vanishes as $P \to 0$. We thus have

$$\lim_{P\to 0} \frac{\mathrm{Var}[i(X;\bar{Y})]}{V^{\mathrm{d}}} = \lim_{P\to 0} \frac{\mathbb{E}[i(X;\bar{Y})^2] - I(X;\bar{Y})^2}{V^{\mathrm{d}}} = 1 - 0 = 1 \tag{5.161}$$

This completes the proof of (5.95). □

Chapter 6

Object coding

This Chapter is a sketch of an idea for a communication scheme using whatever one finds at hand. The motivation is as follows: sometimes one finds at one's disposal many degrees of freedom to use in a memory or communication scheme, but has little theoretical understanding of what an optimal scheme might look like (optimal, say, in terms of error probability). We have so far encountered setups consisting of multiple optical modes subject to photodetection, atomic systems with an internal state, and briefly mentioned flash cell memories (in the previous Chapter). Characterizing an associated noisy channel might be hard and of limited usefulness in practice - the actual realization of the channel might differ from our idealization, requiring different schemes or at least rendering what we find nonoptimal. We must make do with the setups we find, but it would be nice if something about our analysis outlasts a particular setup. Is there something we can say about a broad class of hard-to-characterize communication setups without wedding ourselves too closely to a particular channel? This Chapter offers a perspective on communication with an ill-characterized channel and suggests some ideas for constructing reasonable schemes in more specific settings once details are filled in.

The idea is roughly to create a random list of binary properties that the channel input (we'll call it an "object" below) might have, so that the ill-characterized noise corrupting the object approximately turns into a binary symmetric channel on the list of properties. Section 6.1 offers an introductory example of what we mean and defines

the setup more concretely. Section 6.2 applies these ideas to the case of random linear properties — e.g., whether a certain random linear combination of vector components exceeds a threshold. We make some observations about the possible existence of a feasibility phase transition in this setting and conclude.

6.1 The object channel

6.1.1 An example

Suppose we are given an object with many degrees of freedom, but little idea of the distribution of these objects, what a typical one looks like, and how the object evolves in time or due to "noise." For concreteness, let's suppose that our object is a pepperoni pizza (see Figure 6.1). The pizza has many degrees of freedom - its shape, weight, positions of the pepperonis and so on - and we suspect that it could enable quite a bit of information to pass from sender to receiver via the pizza delivery channel, but it's hopeless to attempt to construct a channel matrix, compute entropies, or do anything like an optimal scheme.

Let's try to capture the information in the pizza by establishing a correspondence between the pizzas and binary strings. Each entry of the string corresponds to a binary property of the pizza - whether there are more pepperonis on the left on the right, whether its weight is greater than something, whether it is delicious and so on. The properties are not all independent of one another, but so long as their number is not too great, there is probably[1] at least one pizza that has all of them. The sender encodes a message in this list of properties, attempts to make a pizza that satisfies all of the properties, and gives it to the delivery person.

What is the action of the delivery channel on the pizza? This is difficult to describe in terms of the pizza, but easier to describe in terms of the binary string of properties. In the example in Figure 6.1, some pepperonis fell off or were jostled around to the right side of the pizza, violating the first property, but the pizza remained large and delicious, so the binary string of properties changed in only its first bit. From the

[1] "Probably" meaning, say, assuming the entries are sampled Bernoulli(1/2).

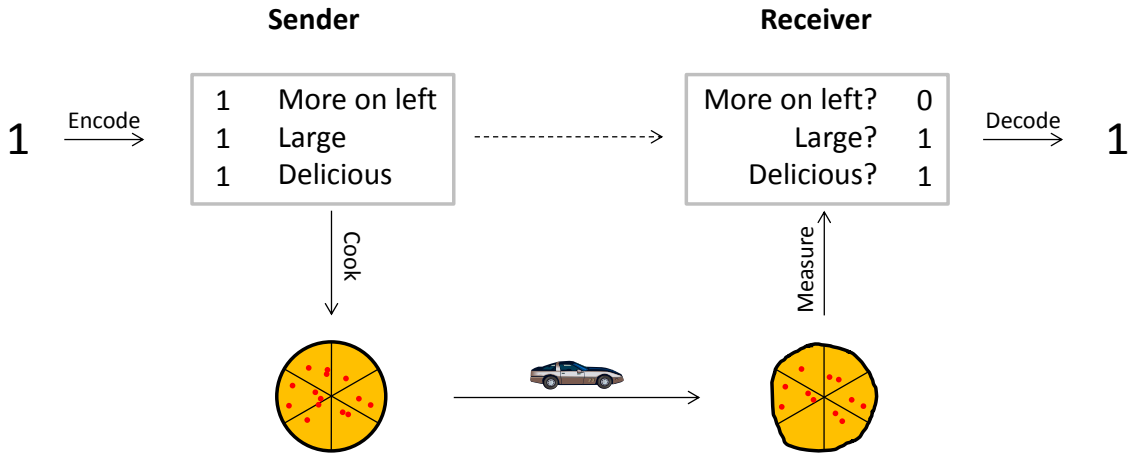

Figure 6.1: The pizza channel. The encoder uses a 3-bit Hamming code ((3,1)) to prepare a binary list of properties. The cook then cooks some pizza that conforms to these properties and gives the pizza to the delivery person. The receiver measures the pizza's properties and outputs the nearest codeword in Hamming distance. The dashed arrow is the induced binary channel between sender and receiver.

perspective of the sender and receiver, there is thus an induced binary channel (dashed arrow). What can we say of this induced channel? If the choice of pizza properties is a good one, then hopefully the violation of one property doesn't say much about another. Better yet, the binary properties should be robust against the kind of noise affecting the pizza[2].

How can we correct errors induced in the pizza delivery channel? This is hard to describe for the pizza, but easier for the binary list of properties: we can try using some error-correcting code suitable for a binary symmetric channel. In the example of Figure 6.1, the sender makes pizzas that correspond to lists of properties that are codewords for the (3,1) Hamming code (000 or 111). Thus error correction is achieved through the outer code. No error correction is done on the pizza itself since we are not interested in reconstructing the input pizza, only the corresponding list of properties. Additionally, if the cook fails to prepare a pizza that conforms to all of the properties,

[2] e.g., a bit corresponding to the property "the weight is an even number of grams" would probably flip with probability a half, so we would not want to use this property too much.

this can be lumped into the action of the channel, and hopefully error-corrected by the outer code.

We could even use what we learn about the induced binary channel to make estimates about the size of the set of typical pizzas and of how noisy the channel is. For example, we could try encoding ever longer Bernoulli(1/2)-distributed lists of properties (the properties would have to be randomly generated somehow), until we begin to fail to cook corresponding pizzas. We could measure how many bits flip per channel use for these randomly generated lists of properties.

So far, this is a loose set of thoughts. Things become a bit more concrete in the next Section, and we present some explicit schemes in Section 6.2.

6.1.2 A block diagram

Figure 6.2 shows the schematic diagram corresponding to our object coding scheme, formalizing and setting up notation for the preceding example. There is an outer code intended for use on a binary symmetric channel implemented by the encoder/decoder pair. The encoder maps messages W to binary lists of properties X^m. The channel accepts object S as input and produces a noisy object $\tilde{S}$ as output according to channel matrix $P(\tilde{S}|S)$, so we must somehow prepare an object that corresponds to the list of properties X^m. The receiver measures those same properties to produce binary string Y^m, which is then given to the decoder to produce an estimate $\hat{W}$ for the intended message.

The ingredients needed for this scheme to work are:

- A way of choosing a "random" set of binary properties. The properties should not be too many (so that at least one object possesses all of them), not be too redundant (so changing one is informative about many others), and not be too susceptible to the actual channel noise.

- A way of preparing an object that corresponds to a random list of properties - this is a random constraint satisfaction problem.

- A way of measuring the set of properties of the received noisy object.

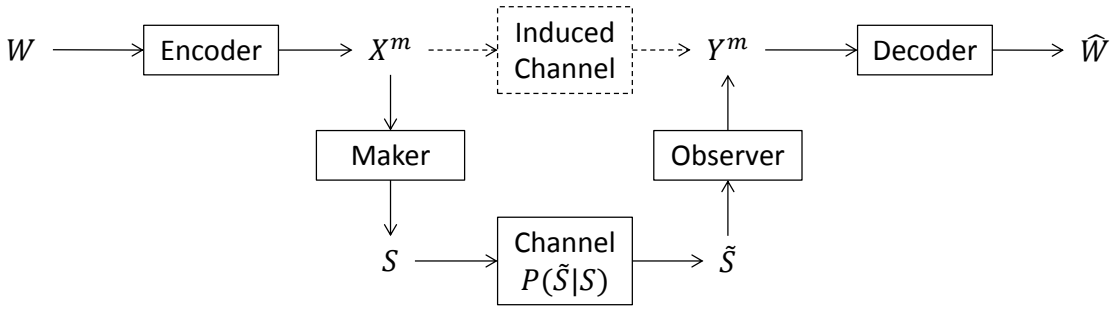

Figure 6.2: Object coding. The encoder maps the message W into a binary string X^m corresponding to an assignment of properties of an object. An object S is prepared that corresponds to this set of properties and is sent through the channel to produce a noise-corrupted object $\tilde{S}$ according to the channel matrix $P(\tilde{S}|S)$. The receiver measures the same set properties of $\tilde{S}$ that the sender used and produces binary string Y^m, which is passed to the decoder to obtain a guess $\hat{W}$ for the intended message.

It would also be nice if it were easy to extend the list of properties by perturbing the object; if we use some message-passing scheme to make objects that correspond to a list of properties, it might be possible to run the scheme a bit longer after adding a few more constraints until all are satisfied. It might also be ok if no object is found that satisfies all of the properties, but only most of them, as this case can be lumped into a noisier induced binary channel between lists of properties and our measurements of those properties.

Acting randomly and using random constraints to achieve good performance for information processing is nothing new: there are efficient random constructions for LDPC codes (the properties are parity check constraints that include a uniformly randomly drawn set of bits), for compressed sensing (the properties are random linear functions of the object), for the construction of probabilistic data structures like Bloom filters (the random properties are a collection of hash functions), and for many other things.

Our contribution here is an amalgamation of some of these ideas into the perspective of Figure 6.2. In a random LDPC code, a set of randomly-drawn constraints must all have the same value (the parity checks must evaluate to 0); in our case,

we think of the parity checks as carrying a message — the parity checks can be randomly assigned to 0 or 1 and the bits are responsible for taking on values that correspond to this message. Our observer makes a set of measurements of the same randomly-generated properties that the maker used to prepare the object; unlike in compressed sensing, our goal is not to reconstruct the underlying object from these measurements, but to recover the binary list of properties that was encoded in the object. The hash functions used in a Bloom filter are ideally as brittle with respect to variations in the input data as possible to minimize false positive probability; for our purpose, we would like to instead find properties that are as insensitive to channel noise as possible.

The next Section presents an explicit construction that follows this approach.

6.2 Linear properties

In this Section we consider a particular instance of the coding schemes outlined above suitable for objects that are distributions or more generally collections of numbers. We aim to create random descriptions of our objects using linear constraints. The object preparer in this case becomes a linear program solver. The object measurer becomes a matrix multiplication followed by thresholding.

6.2.1 Setup

Imagine we have the following setup: Our object is a collection of n wires or bins that each carry some amount of stuff. If these are optical channels, then perhaps we split some total amount of power between them. If the bins are cells in a flash memory, then we are loading some number of electrons in each bin. The observation model depends on the particular system we are imagining: for optical channels, the observations in a fixed unit of time might be a Poisson-distributed number of clicks, with mean determined by the power in the channel plus some background noise rate; for cells in a flash memory, we might have something more complicated as we measure

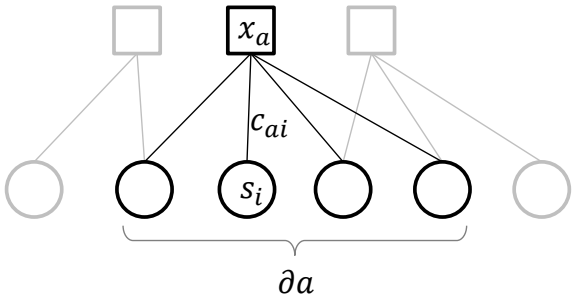

Figure 6.3: Factor graph for the setup in 6.3. The factor a corresponds to a linear constraint on the variable nodes in ∂a with coefficients c_{ai}.

the charge in each bin. We will make more concrete assumptions as we need them, but for now let's set some notation:

$$s_i : \text{ amount of stuff in bin } i \quad \text{for } i \in \{1, \ldots, n\} \tag{6.1}$$

$$\tilde{s}_i : \text{ observation of bin } i \tag{6.2}$$

The assumptions we will need are that $s, \tilde{s}_i \in \mathbb{R}$ and that the expectation $\mathbb{E}[\tilde{s}_i]$ is linear in s_i.

We define things with respect to the bipartite graph shown in Figure 6.3. Here the circles correspond to our n bins and the squares correspond to m constraints. We follow the notation of (Mézard and Montanari, 2009), where $i \in \{1, \ldots, n\}$ indexes the bins, $a \in \{1, \ldots, m\}$ indexes the constraints, and $\partial a \subset \{1, \ldots, n\}$ is the set of bins connected to the a-th constraint (similarly, ∂i is the set of constraints that include bin i).

The constraints are some functions $f_a(\partial a, x_a)$ of the bins ∂a and the constraint *setting* x_a. Recalling the discussion of Section 6.1.2, in our scheme we think of the constraint settings $(x_1, \ldots, x_m)$ as encoding a message. The function f_a must evaluate to a value in some acceptance set for the constraint to be considered satisfied. For example, in the parity check code setting, the constraints are $f_a(\partial a, x_a) = \left(\bigoplus_{i \in \partial a} s_i\right) \oplus x_a = 0 \pmod 2$ and the settings are restricted to $x_a = 0 \forall a$.

We impose linear constraints on our object (the collection of n bins). Let the a-th constraint be of the form

$$x_a \cdot \left(\sum_{i \in \partial a} c_{ai} s_i \right) > 0 \tag{6.3}$$

where

$$x_a \in \{-1, +1\} \tag{6.4}$$

$$c_{ai} \in \mathbb{R} \tag{6.5}$$

where x_a is the setting of the a-th constraint, and $\{c_{ai}\}_{i \in \partial a}$ is a set of coefficients (we shall draw them randomly as described below). Recall from the discussion in Section 6.1.2 that the sender shall seek to exhibit an assignment of the bins s^n that satisfies the m constraint settings x^m. The receiver receives a noisy measurement y^m and forms an estimator for the constraint settings by measuring the object — computing and thresholding the constraint values:

$$y_a = \text{sign} \left(\sum_{i \in \partial a} c_{ai} \tilde{s}_i \right) \tag{6.6}$$

where we have to define sign(0) somehow.

So long as $\mathbb{E}[\tilde{s}_i]$ is proportional to s_i and there is not much noise, we expect y_a to be be a good estimator for x_a. We shall assume a particular channel (AWGN) in Section 6.2.6. How do we find the s^n corresponding to all constraints? This involves solving a linear program as described in Section 6.2.3.

6.2.2 Robustness to noise

Suppose we have values s^n that satisfy the constraint settings x^m in (6.3). Leaving for now aside the issue of finding this assignment, how robust is it to noise in the channel? That is, how likely is it that the measured constraint setting $y_a = x_a$ (6.6)? We have not said what the channel is yet, but we can make this observation: since the constraints are linear, we can scale our $s_i \to \alpha s_i$ for any positive α to produce another satisfying assignment and to satisfy the power constraint (6.20) for arbitrarily

low P. As $\alpha \to 0$, the scheme becomes increasingly vulnerable to noise in that the probability $P(y_a \neq x_a) \to 1/2$. We shall make this observation more precise once we fix a particular observation channel (we shall use the AWGN in Section 6.2.6).

We can add robustness to noise to our scheme by insisting that not only are the constraints (6.3) satisfied, but satisfied by some positive margin b, which we shall try to make as large as possible without violating the power constraint. That is, we seek an assignment s^n that in addition to (6.3) satisfies for all constraints $a \in \{1, \ldots, m\}$

$$\left| \sum_{i \in \partial a} c_{ai} s_i \right| \geq b \tag{6.7}$$

Of course once we have some assignment of the bin values s^n that satisfies all of the linear constraints (6.3), we can set $s' = \alpha s$ for α large enough so that the margin constraint (6.7) is satisfied for any positive b. To deal with this, we must impose another constraint to bound the maximum size of the bin values. A constraint on the L_1 norm of s is straightforward to add:

$$\frac{1}{n}\|s\|_1 = \frac{1}{n}\sum_{i=1}^{n} |s_i| \leq L \tag{6.8}$$

If we are using a power-constrained AWGN with input s, we want instead a constraint on the L_2 norm of s, but this would not fit neatly into a linear program solver. We discuss this issue in Section 6.2.3. We could also instead use a peak-value linear constraint like $|s_i| \leq L \ \forall i$, but stick to the L_1 norm constraint in the discussion that follows.

Figure 6.4 shows this setup. Our object preparation task is loosely like the inverse of classification by a support vector machine (SVM). An SVM tries to find a linear boundary that separates a set of labeled points with the largest possible margin. We start with the labels (x^m) and a fixed boundary (at 0) and try to move the points around to opposite sides with the largest possible margin. There are many possible variations on this (e.g., more than binary constraint settings), but we stick with this setup in the following discussion.

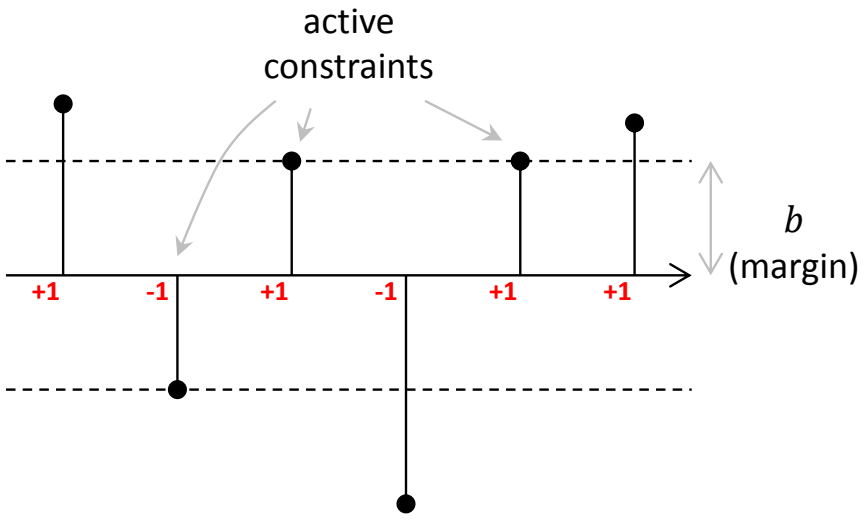

Figure 6.4: Linear constraints (values shown as black dots) satisfied with margin $b > 0$ (dashed lines). Red numbers indicate constraint settings. Three of the constraints (2, 3, and 5) are active — satisfied with equality.

6.2.3 Optimization problem

We are now ready to state the optimization problem to be solved by our encoder, which maps constraint settings x^m to bin values s^n.

6.2.3.1 Optimization problem P1

Let X denote the $m \times m$ diagonal matrix $X_{aa} = x_a \in \{-1, +1\}$, C denote the $m \times n$ matrix formed by the constraint coefficients, $C = (c_{ai})$, $\mathbf{s} \in \mathbb{R}^n$ denote the vector s^n, and the scalar b denote the constraint satisfaction margin (P1):

$$\arg\max_{\mathbf{s},b} b \tag{6.9}$$

subject to

$$XC\mathbf{s} \geq \mathbf{0}_{m\times 1} \tag{6.10}$$

$$|C\mathbf{s}| \geq b\ \mathbf{1}_{m\times 1} \tag{6.11}$$

$$\frac{1}{n}\|\mathbf{s}\|_1 \leq L \tag{6.12}$$

where $\mathbf{0}$ and $\mathbf{1}$ denote vectors of all 0s, 1s, respectively, of the sizes indicated and the inequalities and absolute value are meant point-wise for all elements when comparing two vectors. Appendix 6.A shows how to turn (P1) into a linear program in standard form.

For the natural case of the mean-power-constrained AWGN, we would instead want to solve another optimization problem:

6.2.3.2 Optimization problem P2

In the notation of optimization problem (P1),

$$\arg\max_{\mathbf{s},b} b \tag{6.13}$$

subject to

$$XC\mathbf{s} \geq \mathbf{0}_{m\times 1} \tag{6.14}$$

$$|C\mathbf{s}| \geq b\ \mathbf{1}_{m\times 1} \tag{6.15}$$

$$\frac{1}{n}\|\mathbf{s}\|_2^2 \leq P \tag{6.16}$$

We have replaced the linear constraint $\|\mathbf{s}\|_1 \leq nL$ with a quadratic constraint $\|\mathbf{s}\|_2^2 \leq nP$. The quadratic constraint appears to be a fly in the ointment because it prevents us from applying a linear program solver to this problem[3]. It doesn't change the convexity of the problem[4], however, and there exist methods to solve (P1) exactly (e.g., (Martein and Schaible, 1987)). We deal with this issue using an approximate method outlined below.

We would really like to just use a linear program solver to keep the computations simpler and easy to code up (and possibly easier to implement in a device that used this scheme), so we resign ourselves to an approximate method: we adapt the solution to (P1) to obtain an approximate solution to the power-constrained problem (P2).

[3] The absolute value constraint (6.11) and the L_1 norm constraints aren't linear either, but can be massaged into a linear form using a standard trick, as shown in Appendix 6.A.

[4] Because $\{s : \|s\|_2^2 \leq nP\}$ is a convex set that we intersect with the simplex formed by the linear constraints.

Here "approximate" means that the solution satisfies all of the linear constraints and the power constraint, but the margin b might not be maximal.

Method to solve (P2) approximately:

> **Input**: Constraint matrix C, constraint settings x^m, and power constraint P.
>
> **Output**: Solution to (P2) (if it is feasible): bin values $s^n (= \mathbf{s})$ satisfying the linear constraints and the power constraint with (possibly-not-maximumally large) margin $b > 0$.
>
> 1. Solve (P1) using a linear program solver to obtain solutions b' and $\mathbf{s}'$ (if it is feasible).
>
> 2. Find $\alpha > 0$ s.t. $\|\alpha \mathbf{s}'\|_2^2 = P$.
>
> **Return** $\mathbf{s} = \alpha \mathbf{s}'$, $b = \alpha b'$.

That is, we solve the L_1-constrained problem (P1) and then scale the solution until it satisfies the power constraint of (P2). How do we know that scaling $\mathbf{s} = \alpha \mathbf{s}'$ yields another set of assignments that satisfies all of the linear constraints? This follows from the linearity and homogeneity of the constraints (6.10) and is shown in Appendix 6.A. How close is this approximate solution to the solution of (P2)? We don't know, but we offer some observations for estimating this in Appendix 6.A.

It is possible that (P2), (P1) are infeasible (have no solution) [5], in which case we declare an error. We expect that if the number of constraints is too large, a "typical" set of constraints is infeasible. We investigate this issue in Section 6.2.5 and observe evidence of a feasibility phase transition in the number of constraints.

6.2.4 Constraint matrix ensemble

Let's make things concrete by fixing an ensemble of constraint satisfaction problems for our encoder to solve. For the imagined benefit of practicality, we don't want these

[5] Since $\mathbf{s} = 0$, $b = 0$ is always a solution, we call a problem infeasible if it is infeasible with margin $b > 0$. We use non-strict inequalities in (P1) and (P2) since inputs for linear program solvers are in this form (this way we can find a maximum over a closed set).

problems to be too hard for a linear solver to solve, so let's use an ensemble of sparse constraints. That is, only a few of the coefficients c_{ai} in each constraint (6.3) are non-zero, so the bipartite factor graph of Figure 6.3 is a sparse graph. There are two things to specify in order to fix the constraint ensemble:

- The constraints factor graph — which constraints are non-zero.
- The values of the constraints coefficients c_{ai}.

There are two popular ways to define ensembles of sparse graphs. The one we use is: given n variable nodes ("bins") and m constraint nodes, we fix the degree of each constraint node to some value k, and for each constraint node a draw the boundary ∂a uniformly randomly from the set of subsets of $\{1, \ldots, n\}$ of size k (there are $\binom{n}{k}$ such subsets). Another way to define a sparse graph ensemble is to define a fixed probability for each of the k-subsets to correspond to a constraint, and then draw the constraints independently with this probability; here the total number of constraints is binomially distributed[6]. We stick to the former method in the discussion that follows. We set

$$|\partial a| = k \ \forall a \in \{1, \ldots, m\} \tag{6.17}$$

for some integer k. We then uniformly randomly draw k distinct variable nodes and connect them to the a-th constraint. How do we set the number m of constraints? We discuss this question in Section 6.2.5.

What of the coefficient values c_{ai}? Let's use the popular choice (e.g., from compressed sensing) to sample from the standard normal distribution:

$$c_{ai} \sim \mathcal{N}(0, 1) \text{ iid } \forall a, i \tag{6.18}$$

Another reasonable choice is to sample $c_{ai} \in \{+1, -1\}$ iid $\sim$ Bernoulli$(1/2)$ $\forall a, i$. The Bernoulli ensemble might seem more suitable than the Gaussian ensemble for a practical implementation, since then checking whether the constraints are satisfied or not (or measuring the channel output as in (6.6)) involves only two possible edge values

[6] We encountered another ensemble in Chapter 2 in our photonic scheme to decode LDPC codes. There, the degree of each variable and constraint node was fixed, and the edges were drawn uniformly.

± 1, possibly simplifying a circuit implementation. We provide numerical results for both the Gaussian and Bernoulli ensembles in Section 6.2.6.

Recall that the actual constraint matrix used in the optimization problem (P1), (P2) is XC, where X is a diagonal matrix of constraint settings, $x_{aa} \in \{+1, -1\}$ iid $\sim$ Bernoulli(1/2) $\forall a$. Thus the distribution of values c_{ai} should be invariant under sign change, $c_{ai} \to -c_{ai}$, as the Gaussian and Bernoulli distributions are.

6.2.5 Evidence of a feasibility phase transition

We have now specified the ensemble of constraints, but have not yet fixed the number of constraints. We proceed to investigate the feasibility of our linear constraint satisfaction problem as a function of the number of constraints in this Section. We can consider this question without reference to any communication scheme at all (though this motivation led us here); below we put this question in the context of related work.

Let's consider what happens when we increase the ratio m/n of the number of constraints to the number of bins. Perhaps as the ratio m/n grows beyond some threshold the probability of feasibility decreases dramatically. This happens, for instance, for random constraint satisfaction problems like K-SAT (see (Mézard and Montanari, 2009)) and for randomly drawn LDPC codes[7]. (Gamarnik, 2004) proved the existence of such a phase transition for random linear constraint satisfaction problems, but drawn from a different ensemble[8] than the one we are interested in.

Figure 6.5 shows this scenario for several conditions (see caption; A nice thing about linear programs is that we can establish feasibility in time polynomial in the problem size[9] in contrast to, e.g., number partitioning[10]). We see that when the constraint size $|\partial a|$ is large enough (4 appears sufficient), then most randomly drawn linear programming programs (P1) are feasible for $m/n < 2$ and not feasible for $m/n > 2$. The case of a $|\partial a| = 2$ seems to have a smaller threshold near $m/n = 1$.

[7] Drawing too many constraints results in a code space that contains only the all 0s codeword.
[8] (Gamarnik, 2004) studied the linear programming relaxation of K-SAT.
[9] Though we entrusted our fate to Matlab's implementation of a linear program solver.
[10] This is the problem of determining whether it is possible to partition a set of numbers into two sets with equal sum. For a discussion of the feasibility transition see (Mézard and Montanari, 2009).

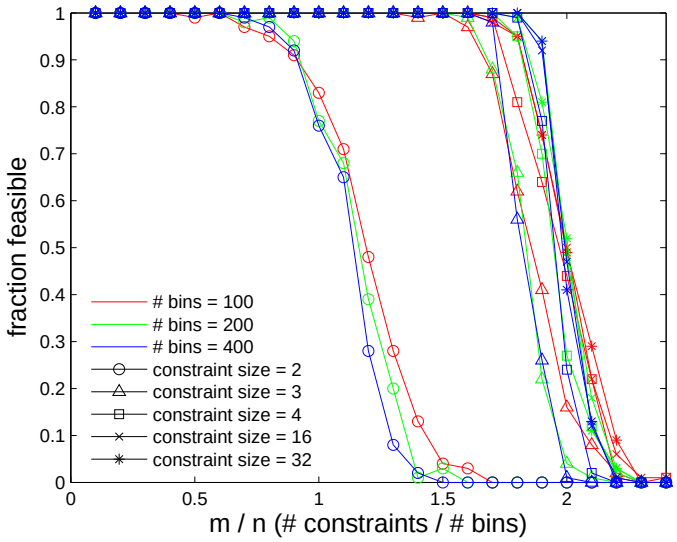

Figure 6.5: Fraction of linear programming problems (P1) that are feasible. The problems are randomly drawn from the ensemble of Section 6.2.4 ($c_{ai} \sim \mathcal{N}(0,1)$). Parameters are indicated in the legend. We sampled 100 problems per parameter setting. We used Matlab's linear program solver.

We observe that, encouragingly for our interpretation of this as a feasibility phase transition, the crossover from feasibility to infeasibility becomes steeper as n (the number of bins) grows near $m/n = 2$ (for $|\partial a| \geq 4$). Figure 6.6 shows the same thing as Figure 6.5, but for the $c_{ai} \in \{+1, -1\}$ ensemble. The transition to infeasibility for $|\partial a| = 2$ seems less sharp than for the $c_{ai} \sim \mathcal{N}(0,1)$ ensemble (Figure 6.5).

How can we understand these phenomenona?

The large $|\partial a|$ case

A heuristic argument for the case of $|\partial a|$ large enough is as follows: at least one solution to the LP problem is at a vertex of a convex polytope of dimension n [11]. The

[11] Actually $n+1$, since we are treating the margin b on the same footing as the n bin values; See Appendix 6.A.

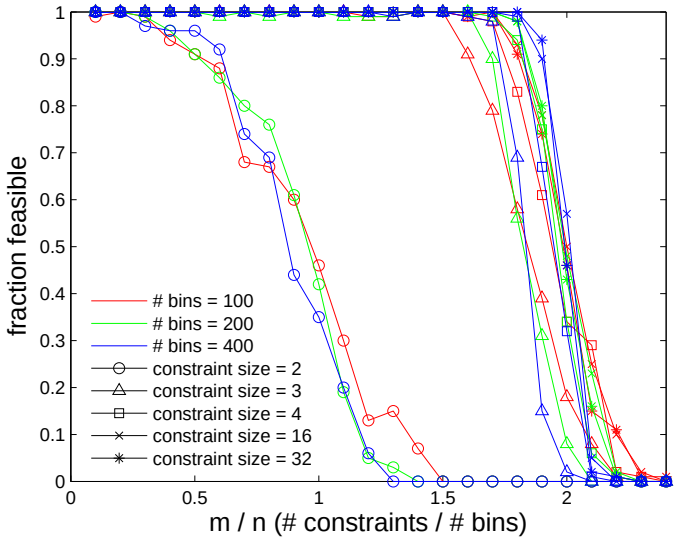

Figure 6.6: Same as Figure 6.5, but with the constraint coefficients drawn Bernoulli(1/2) $c_{ai} \in \{+1, -1\}$.

number of faces meeting at a vertex is n [12], so there are n "active" constraints at this solution — inequality constraints that are satisfied with strict equality.

Now suppose we have some assignment of bin values and impose one more randomly-drawn linear constraint from the ensemble of Section 6.2.4. With probability 1/2 this constraint is already satisfied by the current bin assignment. The other half of the time we must change our assignment in order to satisfy the new constraint, so this new constraint becomes active. So we expect to be able to add about $2n$ constraints so that n of them are active. Adding more constraints would require more than n active constraints at a solution, which happens with probability 0. Thus we expect to exhibit an assignment of n bins that satisfies the m constraints for $m < 2n$, and for no such assignment to exist for $m > 2n$.

[12] More than n faces can meet at a vertex, but this happens with probability 0 for randomly drawn constraints, as in our case.

The case $|\partial a| = 2$

What about the case $|\partial a| = 2$? There appears to be a smaller threshold, perhaps neear $m/n = 1$, for the transition between feasibility and infeasibility (see Figure 6.5 for the Gaussian ensemble; the transition for the Bernoulli ensemble appears less sharp in Figure 6.6). We have concocted another post-hoc heuristic for this case: When each constraint includes exactly two bins, then we can replace the bipartite factor graph of constraints (see Figure 6.3) with another graph G that has only the n variable nodes and connects two variable nodes whenever these two nodes appear in the same constraint. So long as G is a tree, we can find an assignment of the variable nodes (bins) that satisfies all of the constraints — for each connected component, take a leaf, and then work outwards towards the other leaves, making sure each constraint is satisfied[13].

What if G has a cycle of size c? The cycle corresponds to c constraints on c variables, so there is at least one satisfying assignment of the variables. If G contains another distinct cycle involving the same c variables, then we have overconstrained these variables and with probability 1 there is no satisfying assignment. How many edges can G have before it has two distinct cycles on the same set of variables? Let's recall a result about Erdős-Rényi random graphs (Erdős and Rényi, 1960): suppose $m/n > 1$, then G has $O(n)$ cycles with probability tending to 1 as $n \to \infty$. If $m/n < 1$, then G has no cycles with probability tending to 1 as $n \to \infty$. We moreover want to find distinct cycles in the same connected component, but this follows from another property of Erdős-Rényi graphs: if $m/n > 1$, then G has a giant component ($O(n)$ vertices) with probability tending to 1 as $n \to \infty$. This giant component includes $O(n)$ of the cycles, so it contains distinct cycles. Thus we expect to exhibit an assignment of n bins that satisfies the m constraints for $m < n$, and for no such assignment to exist for $m > n$.

[13] e.g., suppose s_1 and s_2 are connected via the constraint $-3s_1 + 2s_2 > 0 \Rightarrow s_1 < \frac{2}{3}s_2$. Then set $s_1 = 0$ and $s_2 = 1$ and move on to the neighbors of s_2. The procedure might only fail if we make our way back to s_1, but this can not happen since G is a tree by assumption.

We have given heuristics for computing a feasibility threshold at $m/n = 2$ for large-enough $|\partial a|$ and at $m/n = 1$ for $|\partial a| = 1$ and presented numerical evidence for this phenomenon (see Figures 6.5 and 6.6). Of course an actual proof would be nice!

6.2.6 Additive white Gaussian noise

We have examined the feasibility of random linear constraint satisfaction problems in the previous Section, but not yet talked about using these constraints as part of a communication scheme. Let's make our scheme more concrete by assuming a particular observation model, the additive white Gaussian noise (AWGN) channel. That is,

$$\tilde{S}_i = S_i + Z_i \quad \text{where } Z_i \sim \mathcal{N}(0,1) \ \forall i \tag{6.19}$$

where we are using noise variance $N = 1$. We can endow our AWGN with a mean power constraint on any set of settings s^n used as the channel input:

$$\frac{1}{n}\sum_{i=1}^{n} s_i^2 \leq P \tag{6.20}$$

The mean power constraint presents a difficulty for us in that it is not linear in S^n. We shall discuss some ways of circumventing this difficulty later.

There are several reasons why applying our object coding idea to the AWGN is probably a poor idea. The AWGN doesn't well model the photon counting channel we discussed in Chapter 4 (the number of clicks per wire is approximately normal for many clicks, but can not be negative), and there are more accurate models for the flash memory cell readout channel (Bez et al., 2003). Moreover, since so much is known about the AWGN, it is unclear why anyone would ever use our funny random linear constraints scheme. We use the AWGN results to see if there is any promise to our scheme, to evaluate its performance, and to gain some intuition about the

linear constraint setting through an explicit example that might be relevant for other channels[14].

Beyond some numerical observations, the story stops here. First, given an assignment of bin values s^n that satisfies $|C\mathbf{s}| \geq b > 0$, for each constraint a we can estimate the probability $P(y_a \neq x_a)$ that the measured constrained setting does not match the input constraint setting. We can do this by recalling that (6.6)

$$y_a = \text{sign}\left(\sum_{i \in \partial a} c_{ai}\tilde{s}_i\right) \tag{6.21}$$

The sum is normally distributed since each term $c_{ai}\tilde{s} \sim \mathcal{N}(s, c_{ai}^2)$, so we can straightforwardly compute error probabilities. This computation is less useful for estimating a mean error probability over all constraints because the measured values of the constraints are not independent, since the measurements y_a are correlated due to the constraints graph structure. In our numerical experiments we estimate the mean error probabilities by random sampling.

The bigger each constraint size ($|\partial a|$), the more terms are in the sum (6.21), increasing the probability of an error. On the other hand, as $|\partial a|$ grows it becomes easier to satisfy a constraint by a larger margin b without violating other constraints because there are more total bins in the support of each constraint to adjust. Which of these effects wins? First let's define a performance measure for our scheme.

Let's define the mean error probability averaged over the all m constraints :

$$\mathrm{P_e} = \frac{1}{m}\sum_{a=1}^{m} P(y_a \neq x_a) \tag{6.22}$$

[14] Besides, in the process we ended up gaining some intuition about coding schemes for the AWGN in the low signal to noise (SNR) regime ($P \ll N = 1$) that lead to the coding scheme presented in Section 5.6

Next let's pretend that we have a memoryless binary symmetric channel[15] with channel matrix

$$P(y|x) = \begin{cases} 1 - \mathrm{P_e} & : y = x \\ \mathrm{P_e} & : y \neq x \end{cases} \tag{6.23}$$

Let's approximate the maximum throughput rate of our object coding scheme by the capacity of this BSC and call this approximation R_{obj}:

$$R_{\text{obj}} = 1 - H(\mathrm{P_e}) \tag{6.24}$$

where $H(\cdot)$ is the binary entropy function. If there are m constraints, then the number of bits we are sending through using the linear constraints coding scheme is

$$N_{\text{obj}} = (\#\text{ bits sent by random linear constraints scheme}) \approx mR_{\text{obj}} \tag{6.25}$$

There is some non-vanishing probability of error since $m < \infty$, but we ignore this issue here. Finally, let's compare this to the maximum number of bits we can send using any scheme at all for the AWGN with mean power constraint P:

$$N_{\text{AWGN}} = (\max\ \#\text{ bits sent over AWGN}) \approx nC_{\text{AWGN}}(P) = \frac{n}{2}\log\left(1 + \frac{P}{2}\right) \tag{6.26}$$

The ratio $N_{\text{obj}}/N_{\text{AWGN}}$ is one measure of performance. Given that with some probability our linear constraint problems turn out to be infeasible, let's treat the infeasibility case as an "erasure," so the receiver has no information about the constraint settings x^m. Then the rate for our linear constraints scheme is reduced by the complement of the probability of an erasure. Thus the performance metric we use is

$$\frac{N_{\text{obj}}\mathrm{P_{feas}}}{N_{\text{AWGN}}} \tag{6.27}$$

where $\mathrm{P_{feas}}$ is the probability that a randomly drawn constraint satisfaction is feasible for some parameters. Figure 6.7 plots this measure of performance for the same

[15] It isn't memoryless because the y_a measurements are correlated due to the constraints factor graph structure (see Figure 6.3).

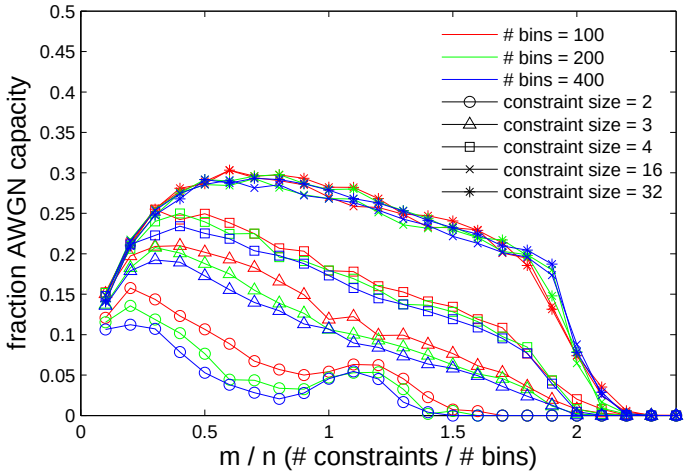

Figure 6.7: Estimated fraction of AWGN with SNR $P = 1$ capacity, measured by $\frac{N_{\text{objP}_{\text{feas}}}}{N_{\text{AWGN}}}$ (6.27). The problems are randomly drawn from the ensemble of Section 6.2.4 ($c_{ai} \sim \mathcal{N}(0,1)$). Parameters are indicated in the legend. We sampled 100 problems per parameter setting. We used Matlab's linear program solver.

parameters as Figure 6.5. We are using the AWGN with SNR $P = 1$. Figure 6.8 shows the same thing as Figure 6.7, but for the $c_{ai} \in \{+1, -1\}$ ensemble.

We see for the $c_{ai} \sim \mathcal{N}(0,1)$ ensemble (Figure 6.7) that we come closer to AWGN capacity as the constraint size $|\partial a|$ grows. The optimal ratio of constraints to bins seems to be near $m/n \approx 1/2$. For the $c_{ai} \in \{+1, -1\}$ Bernoulli(1/2) ensemble (Figure 6.8), we see that small constraint sizes seem to do better than large constraint sizes for $m/n < 0.5$, while the curves for the large constraint sizes look similar to the curves in Figure 6.7.

These figures appear discouraging: simply using quantized ± 1-valued inputs to the AWGN channel with SNR $P = 1$ gets us within about 0.74 of capacity, while none of the curves in Figures 6.7 and 6.8 rise above about 0.4.

The motivation for the random linear constraints coding scheme is to make some not-too-inefficient communication over ill-understood channels, rather than the AWGN, so this outcome is perhaps not surprising. We could do better if the receiver attempted error correction whenever the received constraint was satisfied by less than the expected margin b. Finding the nearest margin-satisfying set of bin values to some

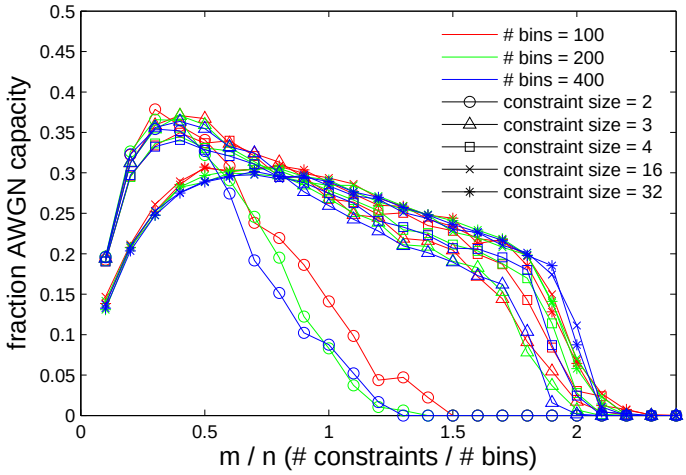

Figure 6.8: Same as Figure 6.7, but with the constraint coefficients drawn Bernoulli(1/2) $c_{ai} \in \{+1, -1\}$.

other set of bin values is a non-convex optimization problem[16], so we cannot easily use another linear program solver for this task. Perhaps some message-passing passing scheme on the constraints factor graph can be devised to implement this error correction (though having error correction conflicts with the picture of leaving error correction to the outer code of Figure 6.2).

Finally, let's take a look at a histogram of bin values that a linear program solver found to satisfy some randomly-drawn linear constraints. Figures 6.9 and 6.10 show this for the Gaussian and Bernoulli(1/2) constraint coefficients ensembles, respectively. We chose to set the number of constraints and variables equal ($n = m = 200$) based on our previous observations — the problem is probably feasible and does reasonably well in terms of fraction of AWGN capacity versus other parameter settings (see Figures 6.7 and 6.8).

We see that a large number of all variables (about a third) is fixed at 0. As we increase the number of constraints, more and more constraints become active and the peak at bin value $s = 0$ decreases. Eventually the problems become infeasible.

[16] To see this, suppose our only constraint is $|s_1| \geq 1$ and we receive $\tilde{s}_1 = 0$. Then we can satisfy the margin by either increasing or decreasing $\tilde{s}_1$ by 1. The sum of these solutions leaves $\tilde{s}_1$ unchanged and the margin remains unsatisfied, so the set of solutions is non-convex.

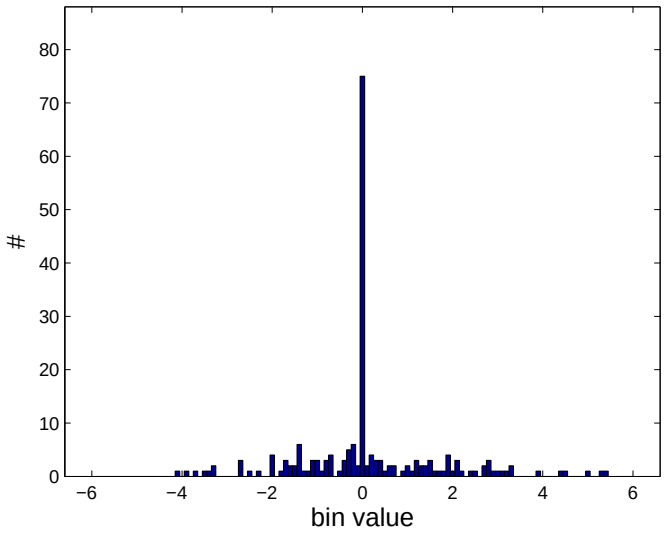

Figure 6.9: Histogram of s^n — bin values satisfying a random constraint problem drawn from the ensemble of Section 6.2.4 ($c_{ai} \sim \mathcal{N}(0,1)$). Here $m = n = 200$, $|\partial a| = 16\ \forall a$, and $\frac{1}{n}\|x\|_1 = 1$. For this sample $\frac{1}{n}\|x\|_2 \approx 2.52$. We used Matlab's linear program solver.

These observations and optimism about this scheme inspired us to try (Section 5.6 the discrete input distribution as a capacity-achieving input for the AWGN in the low SNR regime.

6.A Maximizing the constraint margin with linear programming

This Appendix shows how to map the optimization problem (P1) in Section 6.2.3 into a linear programming problem in a standard form. We also make some observations about the solutions of (*P*1) that justify linearly scaling these solutions to approximately solve (P2) in Section 6.2.3.

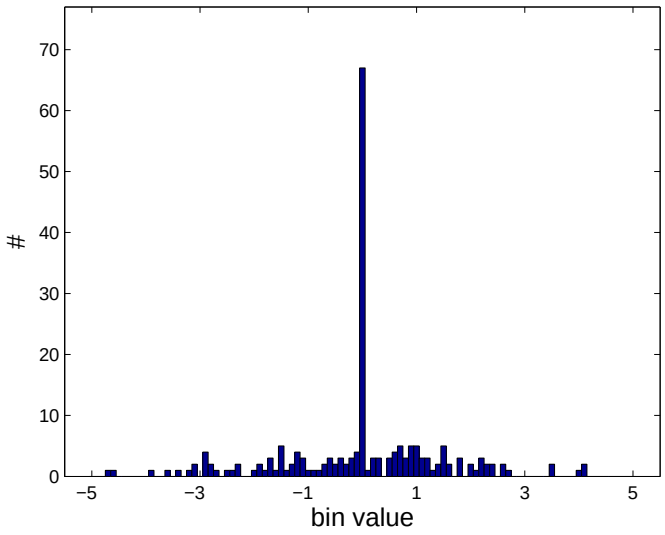

Figure 6.10: Same as Figure 6.9, but with the constraint coefficients drawn Bernoulli(1/2) $c_{ai} \in \{+1, -1\}$. For this sample $\frac{1}{n}\|x\|_2 \approx 2.24$.

6.A.1 Arranging (P1) in standard form

Let's recall the statement of optimization problem (P1): Given constraint settings $x^m \in \{+1, -1\}^m$, constraint coefficients $c_{ai} \in \mathbb{R}$, and $L > 0$:

$$\begin{aligned} &\text{find } s, b \text{ that} \\ &\text{maximize } b \\ &\text{subject to} \\ &\quad XCs \geq \mathbf{0}_{m\times 1} && (6.28) \\ &\quad |Cs| \geq b\ \mathbf{1}_{m\times 1} && (6.29) \\ &\quad \frac{1}{n}\|s\|_1 \leq L && (6.30) \end{aligned}$$

where X is a $m \times m$ diagonal matrix $X_{ii} = x_i$, C is a $m \times n$ matrix formed by the constraint coefficients $C = (c_{ai})$, and s denotes s^n viewed as a column vector.

The standard form of a linear program is: Given $\underline{A}_{m\times n}$, $\underline{b}_{m\times 1}$, and $\underline{c}_{n\times 1}$:

$$\begin{aligned} &\text{find } x \text{ that} \\ &\text{maximizes } \underline{c}\cdot\underline{x} \\ &\text{subject to} \\ &\underline{Ax} \le \underline{b}_{m\times 1} \qquad (6.31) \\ &\underline{x} \ge \mathbf{0}_{n\times 1} \qquad (6.32) \end{aligned}$$

where the "standard" quantities are underlined to avoid confusion with our variables.

There are two standard tricks to apply to convert (P1) into standard form. The first is to turn the absolute value constraint (6.29) into a linear constraint. This can be done by writing:

$$\begin{aligned} |s_i| &\le b \qquad (6.33) \\ &\Updownarrow \\ s_i &= s_i^+ - s_i^- \qquad (6.34) \\ 0 &\le s_i^+ \le b \qquad (6.35) \\ 0 &\le s_i^- \le b \qquad (6.36) \end{aligned}$$

so we have broken each bin value $s_i \to (s_i^+, s_i^-)$ into a positive and negative part that satisfy linear constraints.

Second, the problem (P1) seeks to maximize the margin b in (6.29). We can handle this by appending the margin b to the list of bin values s^n. The vector to maximize $\underline{c}$ is then 0 for every entry except the one corresponding to the margin b, where it is positive. We must also modify the constraints matrix A to have a rightmost column of 1s for the margin variable b. Below we state explicitly what these sentences mean.

Putting these two tricks together, we map to a linear programming problem in standard form thus:

$$\underline{A}_{(m+1)\times(2n+1)} = \left[\begin{array}{ccc|ccc|c} & -XC & & & XC & & \begin{matrix}1\\ \vdots \\ 1\end{matrix} \\ \hline 1 & \cdots & 1 & 1 & \cdots & 1 & 0 \end{array}\right] \tag{6.37}$$

$$\underline{x}_{(2n+1)\times 1} = \left[\begin{array}{ccc|ccc|c} s_1^+ & \cdots & s_n^+ & s_1^- & \cdots & s_n^- & b \end{array}\right]^{\mathrm{T}} \tag{6.38}$$

$$\underline{b}_{(m+1)\times 1} = \left[\begin{array}{ccc|c} 0 & \cdots & 0 & nL \end{array}\right]^{\mathrm{T}} \tag{6.39}$$

$$\underline{c}_{(2n+1)\times 1} = \left[\begin{array}{ccc|ccc|c} 0 & \cdots & 0 & 0 & \cdots & 0 & 1 \end{array}\right]^{\mathrm{T}} \tag{6.40}$$

where the $\pm XC$ blocks act on the positive and negative parts of s. The $-XC$ block is first because in (P2) we have a lower bound constraint (6.28), while in the standard form for linear programming we have an upper bound constraint (6.31). The last row of $\underline{A}$ corresponds to the constraint (6.29) $\|s\|_1 \leq nL$. We can now use a linear program solver to solve this problem and pull out the solutions s^n and b to the problem (P1).

6.A.2 Scaling the solution to (P1) to obtain an approximate solution to (P2)

Next, we'd like to show that rescaling the solution s', b' to (P1) to satisfy the quadratic power constraint in (P2) produces an approximate solution of (P2), meaning that the linear constraints (6.28). This follows from the form of (6.28) and (6.29) in (P1): scaling $s \to \alpha s'$ results in an equivalent problem with $b \to \alpha b$, $L \to \alpha L$ (this would not be the case if the constraint (6.28) were not homogeneous - if we did not use a 0 lower bound). Thus, scaling the solution of (P1) is equivalent to finding the solution to (P1) with some other upper bound for the L_1 constraint (6.30). In particular, we can scale the solution of (P1) so that it satisfies the power constraint of (P2). It would be nice to estimate how close the solution of (P1) is to (P2).

Bibliography

R. F. Bailey. Error-correcting codes from permutation groups. *Discrete Mathematics*, 309:4253–4265, 2009. 107

A. Barchielli. *Open Quantum Systems III: Recent Developments*. Springer-Verlag, New York, NY, 2006. 6, 29

P. Barclay, K.-M. Fu, C. Santori, and R. G. Beausoleil. Hybrid photonic crystal cavity and waveguide for coupling to diamond nv-centers. *Optics Express*, 17:9588, 2009. 42

A. Barg and A. Mazumdar. Codes in permutations and error correction for rank modulation. *IEEE Transactions on Information Theory*, 56:3158–3165, 2010. 107

M. Bayindir, B. Temelkuran, and E. Ozbay. Photonic-crystal-based beam splitters. *Applied Physics Letters*, 77:3902, 2000. 31

J. O. Berger, J. M. Bernardo, and D. Sun. The formal definition of reference priors. *The Annals of Statistics*, 37(2):905–938, 04 2009. doi: 10.1214/07-AOS587. 83

J. M. Bernardo. Reference posterior distributions for bayesian inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41:113–147, 1979. 83

R. Bez, E. Camerlenghi, A. Modelli, and A. Visconti. Introduction to flash memory. *Proceedings of the IEEE*, 91:489–502, 2003. 197

I. F. Blake. Permutation codes for discrete channels. *IEEE Transactions on Information Theory*, 20:138–140, 1974. 107

I. F. Blake, G. Cohen, and M. Deza. Coding with permutations. *Information and Control*, 43:1–19, 1979. 107

T. Botter, D. W. C. Brooks, N. Brahms, S. Schreppler, and D. M. Stamper-Kurn. Linear amplifier model for optomechanical systems. *Physical Review A*, 85:013812, 2012. 9

L. Bouten, R. V. Handel, and A. Silberfarb. Approximation and limit theorems for quantum stochastic models with unbounded coefficients. *J. Func. Anal.*, 254: 3123–3147, 2008. 49

D. Burshtein and G. Miller. Expander graph arguments for message passing algorithms. *IEEE Transactions on Information Theory*, 47:782–790, 2001. 27

H. J. Carmichael. Quantum trajectory theory for cascaded open systems. *Physical Review Letters*, 70:2273–2276, 1993. 5, 29

H. D. Chadwick and L. Kurz. Rank permutation group codes based on Kendall's correlation statistic. *IEEE Transactions on Information Theory*, 15:306–315, 1969. 107

T. M. Cover and J. A. Thomas. *Elements of Information Theory, Second Edition*. John Wiley & Sons, Hoboken, New Jersey, 2006. 79, 91, 92, 93, 123, 129

M. H. A. Davis. Capacity and cutoff rate for Poisson-type channels. *IEEE Transactions on Information Theory*, 26:710–715, 1980. 140, 141

P. Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journ. Royal Stat'l Soc. Series B (Methodological)*, 39:262–268, 1977. 108, 109

P. Erdős and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5:17, 1960. 196

A. Faraon, I. Fushman, D. Englund, N. Stoltz, P. Petroff, and J. Vuckovic. Coherent generation of non-classical light on a chip via photon-induced tunnelling and blockade. *Nature Physics*, 4:859–863, 2008. 29

R. Gallager. Low density parity check codes. *IEEE Transactions on Information Theory*, 8:21–28, 1962. 25

R. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, Cambridge, MA, 1963. 25

D. Gamarnik. Linear phase transition in random linear constraint satisfaction problems. *Probab. Theory Relat. Fields*, 129:410–440, 2004. 193

C. W. Gardiner. Wave-function quantum stochastic differential equations and quantum-jump simulation methods. *Physical Review A*, 46:4363–4381, 1992. 6

C. W. Gardiner. Driving a quantum system with the output field from another driven quantum system. *Physical Review Letters*, 70:2269–2272, 1993. 5, 29

C. W. Gardiner and M. Collett. Input and output in damped quantum systems: Quantum stochastic differential equations and the master equation. *Physical Review A*, 31:3761–3774, 1985. 9

M. J. E. Golay. Note on the theoretical efficiency of information reception with ppm. *Proceedings of the I.R.E.*, 37:1031, September 1949. 167

J. Gough and M. R. James. Quantum feedback networks: Hamiltonian formulation. *Commun. Math. Phys*, 287:1109–1132, 2009. 60

J. M. R. Gough, J. Quantum feedback networks: Hamiltonian formulation. *Commun. Math. Phys.*, 287:1109–1132, 2008. 5, 29

J. M. R. Gough, J. The series product and its application to quantum feedforward and feedback networks. *IEEE Transactions on Automatic Control*, 54:2530–2544, 2009. 5, 29

R. Hamerly and H. Mabuchi. Advantages of coherent feedback for cooling quantum oscillators. *Phys. Rev. Lett.*, 109:173602, Oct 2012. doi: 10.1103/PhysRevLett.109.173602. 9

M. H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001. 83

R. L. Hudson and K. R. Parthasarathy. Quantum Ito's formula and stochastic evolutions. *Communications in Mathematical Physics*, 93:301–323, 1984. 5, 6, 29

A. A. Jiang, M. Schwartz, and J. Bruck. Error-correcting codes for rank modulation. In *ISIT 2008*, pages 1736–1740, Toronto, Canada, July 2008. 107

A. A. Jiang, R. Mateescu, M. Schwartz, and J. Bruck. Rank modulation for flash memories. *IEEE Transactions on Information Theory*, 55:2659–2673, 2009. 107, 111

J. Johansson, P. Nation, and F. Nori. Qutip: An open-source python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 183 (8):1760 – 1772, 2012. ISSN 0010-4655. doi: 10.1016/j.cpc.2012.02.021. 11

J. Johansson, P. Nation, and F. Nori. Qutip 2: A python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234 – 1240, 2013. ISSN 0010-4655. doi: 10.1016/j.cpc.2012.11.019. 11

Y. M. Kabanov. The capacity of a channel of the Poisson type. *Theory Prob. Appl.*, 23:143–147, 1978. 140

M. G. Kendall. *Rank correlation methods*. Griffin, London, 4th edition edition, 1970. 108

J. Kerckhoff. *Quantum Engineering with Quantum Optics*. PhD thesis, Stanford University, 2011. 6

J. Kerckhoff, H. I. Nurdin, D. S. Pavlichin, and H. Mabuchi. Designing quantum memories with embedded control: Photonic circuits for autonomous quantum error correction. *Phys. Rev. Lett.*, 105:040502, Jul 2010. doi: 10.1103/PhysRevLett.105.040502. 14, 22, 49

J. Kerckhoff, D. S. Pavlichin, H. Chalabi, and H. Mabuchi. Design of nanophotonic circuits for autonomous subsystem quantum error correction. *New Journal of Physics*, 13:055022, 2011. 22

C. Kimberling. Qbest lower and upper approximates to irrational numbers. *Elemente der Mathematik*, 52:122–126, 1997. 62

A. Lapidoth and S. Shamai. The Poisson multiple-access channel. *IEEE Transactions on Information Theory*, 44:488–501, 1998. 141, 150

T. Liu, A. Zakharian, M. Fallahi, J. V. Moloney, and M. Mansuripur. Design of a compact photonic-crystal-based polarizing beam splitter. *IEEE Photonics Technology Letters*, 17:1435–1437, 2005. 31

M. Luby, M. Mitzenmacher, and M. Shokrollahi. Improved low-density parity-check codes using irregular graphs. *IEEE Transactions on Information Theory*, 47:585–598, 2001. 25

H. Mabuchi. Cavity-qed models of switches for attojoule-scale nanophotonic logic. *Phys. Rev. A*, 80:045802, 2009. 23, 29, 30, 42, 49

H. Mabuchi. Nonlinear interferometry approach to photonic sequential logic. *Appl. Phys. Lett.*, 99:153103, 2011. 23, 29, 49, 65

D. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory*, 45:399–431, 1999. 25

D. MacKay and R. Neal. Near Shannon limit performance of low density parity check codes. *Electronics Letters*, 33:457–458, 1997. 25

A. Majumdar, M. Bajcsy, D. Englund, and J. Vuckovic. All optical switching with a single quantum dot strongly coupled to a photonic crystal cavity. *IEEE Journal of Selected Topics in Quantum Electronics*, 18:1812–1817, 2012. 29

L. Martein and S. Schaible. On solving a linear program with one quadratic constraint. *Rivista di matematica per le scienze economiche e sociali*, 10(1-2):75–90, 1987. ISSN 1593-8883. doi: 10.1007/BF02090479. 190

M. Mézard and A. Montanari. *Information, Physics, and Computation.* Oxford University Press Inc., New York, 2009. 186, 193

J. von Neumann. *Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components.* Princeton University Press, Princeton, New Jersey, 1956. 3, 74

A. E. B. Nielsen and J. Kerckhoff. Efficient all-optical switch using a lambda atom in a cavity qed system. *Physical Review A*, 84:043821, 2011. 29

K. R. Parthasarathy. *An Introduction to Quantum Stochastic Calculus.* Birkh 1992. 6

D. S. Pavlichin and H. Mabuchi. Photonic circuits for iterative decoding of a class of low-density parity-check codes. *pre-print*, 2013. 4, 5, 13, 15, 21, 23

K. Pinn and C. Wieczerkowski. Number of magic squares from parallel tempering monte carlo. *International Journal of Modern Physics C*, 9:541–546, 1998. 114

Y. Polyanskiy, H. V. Poor, and S. Verdú. Dispersion of gaussian channels. In *ISIT*, pages 2204–2208, Seoul, Korea, June 28 - July 3 2009. 132, 170

Y. Polyanskiy, H. V. Poor, and S. Verdú. Channel coding rate in the finite blocklength regime. *IEEE Trans. Inf. Theory*, 56:2307–2359, 2010. 131, 132, 136, 137, 170

T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47:599–618, 2001. 25

J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978. ISSN 0005-1098. doi: http://dx.doi.org/10.1016/0005-1098(78)90005-5. 83

I. Sanov. On the probability of large deviations of random variables. *Matematicheskii Sbornik*, 42:11–44, 1958. 91

S. Shamai. Bounds on the capacity of a spectrally constrained Poisson channel. *IEEE Transactions on Information Theory*, 39:19–29, 1993. 151

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423,623–656, 1948. 124, 126

C. E. Shannon. Probability of error for optimal codes in a gaussian channel. *The Bell System Technical Journal*, 38:611–656, 1959. 131

M. Sipser and D. A. Spielman. Expander codes. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 566–576, 1994. 22, 26

M. Sipser and D. A. Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42:1710–1722, 1996. 22, 26, 27, 28, 34, 35, 41, 43, 47

D. Slepian. Permutation modulation. *Proceedings of the IEEE*, 53:228–236, 1965. 107

C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15:72–101, 1904. 108

V. Strassen. Asymptotische abschätzungen in Shannon's informationstheorie. In *Trans. 3rd Prague Conf. Inf. Theory*, pages 689–723, Prague, 1962. 131, 132, 170

S. M. Tan. A computational toolbox for quantum and atomic optics. *Journal of Optics B: Quantum and Semiclassical Optics*, 1:424–432, 1999. 11, 14

R. Tanner. A recursive approach to low complexity codes. *IEEE Transactions on Information Theory*, 27:533–547, 1981. 24

N. Tezak, A. Niederberger, D. S. Pavlichin, G. Sarma, and H. Mabuchi. Specification of photonic circuits using quantum hardware description language. *Phil. Trans. Roy. Soc. A*, 370:5270–5290, 2012. 13, 69

S. Verdú. On channel capacity per unit cost. *IEEE Trans. Inf. Theory*, 36:1019–1030, 1990. 167

B. Vigoda. *Analog Logic: Continuous-Time Analog Circuits for Statistical Signal Processing*. PhD thesis, Massachusetts Institute of Technology, 2003. 3

Z. Wang. *Coding for Information Storage*. PhD thesis, California Institute of Technology, 2013. 107, 108

Z. Wang and J. Bruck. Partial rank modulation for flash memories. In *ISIT 2010*, pages 864–868, Austin, Texas, USA, June 2010. 107

H. Wiseman. Quantum trajectories and quantum measurement theory. *Quantum Semiclass. Opt.*, 8:205–222, 1996. 39

H. M. Wiseman. *Quantum Measurement and Control*. Cambridge University Press, Cambridge, UK, 2010. 5, 9, 13, 39