# POSTMODERN CODING THEORY

By

**M. Ryan Julian Jr.**

A dissertation submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

(Mathematics)

at the

**UNIVERSITY OF WISCONSIN – MADISON**

2019

Date of final oral examination: August 16, 2019

The dissertation is approved by the following members of the Final Oral Committee:

N. Boston, Professor, Mathematics, Electrical and Computer Engineering

J. Ellenberg, Professor, Mathematics

M. Gupta, Assistant Professor, Computer Sciences

D. Papailiopoulos, Assistant Professor, Electrical and Computer Engineering

# Abstract

In this work, we formalize the essential aspects of a (channel) coding theory in order to generalize the subject area to cover a broader collection of mathematical settings. After reviewing the classical theory within this new framework, we cover generalizations in three different directions.

We examine block codes over nonabelian groups, where we prove that there are no nonabelian groups whose subgroup block codes support a duality theory, solving an open problem of Dougherty, Kim, and Solé [22]. This result can be rephrased in terms of group theory to say that given a finite group $G$, $G^n$ has a self-dual subgroup lattice for all $n$ if and only if $G$ is abelian.

We define three different permutation coding theories under the Hamming, Kendall tau, and Ulam metrics. In each case, we review what is known about the Hamming and Singleton bounds in these coding theories. For the Kendall tau and Ulam metrics, we provide new insights into the weight enumerators of various subgroups, and demonstrate that for non-normal subgroups the weight distributions can vary significantly under conjugation. We will then study the relabeling that arises from this conjugation action, giving formulas for the minimum and maximum Kendall tau and Ulam weights of a permutation with a given cycle type, and we will apply these theorems to understand the minimum distances of conjugates of subgroup codes under these metrics. We will also introduce a sequence of $W_n^k$ numbers giving the maximum number of inversions in a permutation on $n$ symbols with a single $k$-cycle and $n-k$ fixed points.

Finally, we'll present an entirely new theory of coding curves inspired by recent

research into time-of-flight cameras as well as a more intuitive question of packing ropes into boxes. We will produce several basic bounds in this setting including a Hamming-type bound, and we will provide examples of lattice coding curves that approach these bounds in low dimensions. We also present work on feasible regions, connections to sphere packing, and Manin-type curves. We will finish this section with a brief discussion of several different directions to extend this theory.

# Acknowledgements

This work is the result of not only my own efforts, but also the support and inspiration of many mentors, colleagues, teachers, friends, family, and loved ones.

I'd like to first thank my advisor, Nigel Boston. I am grateful for your years of mentorship and guidance. Your breadth of mathematical knowledge was an invaluable resource for me to draw on, and your workflow has helped to shape how I think as a mathematician.

I'd also like to thank Jordan Ellenberg, Mohit Gupta, and Dimitris Papailiopoulos. In addition to serving on my committee, each of you has inspired me through our collaborations and discussions.

Many of my teachers over the years are especially deserving of thanks. Becky Gill inspired me early on to see the beauty of mathematics and the value in asking the right questions. Judy Zielinski taught me how to think abstractly and encouraged me to explore mathematics independently. She also gave me my first graduate-level coding theory textbook[1], which traveled with me in my library until I finally read it as a graduate student. Paul Sally challenged me with the rigor of modern mathematics and was unwavering in his belief that I could master it. Working with him and observing up-close the tireless work of a living legend was truly inspirational.

Thank you to all of my friends and classmates who have made my time in Madison so enjoyable. The boardgames group provided a much-needed weekly break, Vlad shared with me his fascinating and unique perspectives on math and education over our many

[1]Steven Roman's Coding and Information Theory [73]

pool games, and Christina was always a phone-call away to share grad school stories and advice.

My family has been inspiring me and supporting me from the very beginning. A huge thanks to my parents Ryan and Susan for always encouraging my interests and ensuring that I had the freedom to explore them. I'd also like to thank my sisters Michelle, Karen, and Megan. Through our conversations, you have often sparked interesting mathematical questions, even when you didn't mean to.

Finally, I'd like to thank my partner, Mali. Your love and support have helped me more than I can express. Your adventurous spirit inspires me every day, and I hope we have many more adventures still to come.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

*'I am very seldom interested in applications. I am more interested in the elegance of a problem. Is it a good problem, an interesting problem?'*

(Claude Shannon)

Coding theory arose in the 1940's as the slightly more pragmatic sibling of information theory. Although the preliminary notions of information theory were already being laid out in the development of both wired and wireless communication networks in the years leading into World War II and were rapidly driven forward by Alan Turing and other cryptanalysts during the war, the moment that marked the true birth of the field is certainly Claude Shannon's 1948 article 'A mathematical theory of communication' [76].

In that single paper, Shannon formalizes a mathematical language for discussing communication systems, defines the fundamental information entropy function $H(X) = -\sum_i P(x_i) \log P(x_i)$, and proves both the noiseless coding theorem and the noisy-channel coding theorem. In essence, he both introduced and settled the most fundamental theorems of a nascent mathematical theory in just a few pages. But far from ending the study, these theorems threw down a left and right gauntlet that have been picked up by separate lines of researchers who have been working for decades to meet his challenge.

## 1.1 Historical Development

Shannon's noiseless coding theorem gives a theoretical limit to data compression, but it does not provide a practical method for achieving this limit. More precisely, it says that a sequence of $N$ i.i.d. symbols from a distribution with entropy $H(X)$ can be losslessly compressed into no fewer than $NH(X)$ bits. This result sparked the creation of source coding, which can be loosely defined as the study of compression algorithms that attempt to efficiently meet this entropy limit.

The first reply to this challenge was given by Fano even before Shannon's article had been published. Shannon mentions Fano's method in his 1948 paper and Fano published it himself a year later [25], so this method is typically called Shannon-Fano coding. Just a couple years later, David Huffman, a graduate student in one of Fano's earliest information theory classes, improved on his method with the now-famed Huffman coding [44], which is both fast and optimally compressed amongst codes that encode symbols separately. In the 1970's, this was improved upon with the development of arithmetic coding [70] which encodes a message all at once instead of symbol-by-symbol. It's a bit slower than Huffman coding, but it was the first coding method that met the entropy limit in general. Finally, within the last decade, Jarek Duda introduced asymmetric numeral systems [23], which provide the compression guarantees of arithmetic coding with a speed similar to that of Huffman coding. Together these techniques form the basis for most compression algorithms in use today including those used in the common file formats .zip, .png, .tiff, and many many others.

Where Shannon's noiseless coding theorem dealt with data compression, his noisy-channel coding theorem provides an optimum target for data transmission. Intuitively

this theorem states that for any discrete memoryless channel with noise there exists a channel capacity $C$ with the property that for any rate of information transmission less than $C$ and any error rate $\epsilon > 0$ there exists some sufficiently long code that can achieve this transmission rate with an error rate less than $\epsilon$. Moreover, information transfer above this rate $C$ is not achievable without increasingly large errors. Once again, Shannon's proof is an existence proof that does not give a practical means to actually achieve this bound. Instead decades of research in the area now known as channel coding have been necessary to provide useful methods for achieving these transmission rates.

Just as it was with source coding, the initial reply came quickly. In 1950, Shannon's one-time office mate Richard Hamming published 'Error detecting and error correcting codes' [40], containing his famous Hamming code, which encodes 4 bits of information as a 7 bit string in such a way that single bit errors in transmission can not only be identified, but they can also be corrected[1]. Hamming's title perhaps more clearly states the goals of channel coding. Errors in transmission cannot be avoided. Instead, channel coding seeks to manage and correct those errors by adding in enough redundancy to recover the missing or altered information on the receiving end.

In the years following Hamming's discovery, more and more channel coding methods were developed with closer and closer ties to mathematics. In 1949, Marcel Golay published his Golay code [30], which encodes 12 bits of information in a 23 bit block in such a way that any 3 errors can be corrected. For mathematicians, this code is particularly notable for the fact that it can be constructed from the Mathieu group $M_{24}$ and can be used to construct the Leech lattice. A decade later a trio of papers by Reed and Solomon [68],

[1]In fact, Hamming had developed this code in 1947, before Shannon's paper was published, but publication was delayed in order to protect patent rights.

Bose and Ray-Chaudhuri [6], and Hocquenghem [41] used polynomial algebra over finite fields to develop Reed-Solomon and BCH coding schemes where the error-correcting ability of the codes could actually be tuned according to the application. At around the same time, Gallager [28] introduced low-density parity-check (LDPC) codes, a graph-based coding scheme which finally promised capacity-approaching codes. Unfortunately, due to implementation difficulties, LDPC codes remained almost unknown until they were revived and brought into more common use in the 1990's. Nowadays LDPC codes compete for popularity with turbo codes developed in the 1990's [4] and polar codes from the last decade [1], which each have a strong engineering flavor to their implementation.

Although these channel coding schemes are somewhat less visible to the average technology user than the source coding methods we mentioned earlier, they are in fact ubiquitous. Golay codes, Reed-Solomon codes, and turbo codes have all been trusted by various NASA missions to send back color images from distant reaches of our solar system. Closer to home, LDPC codes handle the errors arising in WiFi transmissions, turbo codes have been employed in the 3G and 4G mobile networks, and polar codes are expected to be used alongside LDPC and turbo codes in the upcoming 5G networks. Error-correcting codes are also essential in cases where the transmission channel is actually storing data on a physical medium for later retrieval. Reed-Solomon codes help to ensure that CD's, DVD's, and Blu-ray disks can survive scratches and still be read with no difficulties apparent to the consumer. The flash memory chips in thumb drives typically use Hamming, Reed-Solomon, or BCH codes. Even the bar codes on our groceries have built in error-detection. All told, the average person probably uses error-control codes thousands of times each day whether they're aware of it or not.

## 1.2 Textbook Channel Coding

In this dissertation, we will focus on channel coding, so for simplicity any future uses of the phrase "coding theory" will refer to channel coding.

As we can see from the above progression, coding theory began as an engineering question that was answered through mathematical means. Even though the subject area has developed to the point that there are a number of very good mathematically rigorous coding theory textbooks available [60, 69, 73, 84], the engineering questions remain primary and in all cases the presentation remains rooted in Shannon's original language. In fact, nearly all textbooks on coding theory include some version of the following diagram from Shannon's paper somewhere within the first few pages[2].

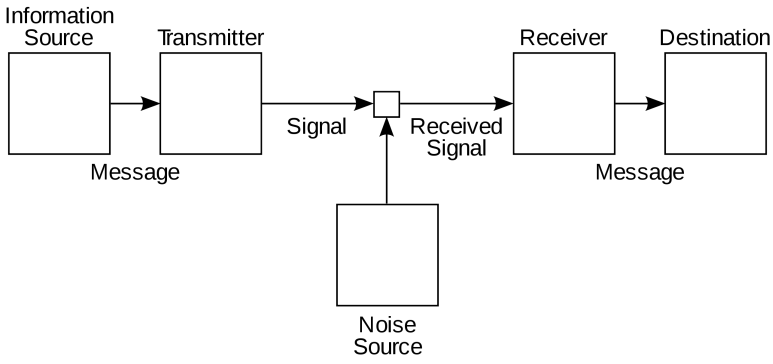


Figure 1: Schematic of a general communication system.

In coding theory textbooks, codewords are nearly always thought of as a sequence of symbols from some alphabet $\Sigma$, and codes are simply subsets of $\Sigma^*$. Most presentations specialize still further by focusing on block codes, which are subsets of $\Sigma^n$ for some $n \in \mathbb{N}$. Structures are imposed on top of this in order to exploit algebraic and geometric symmetry in order to narrow the search for good codes while also providing clever and

[2]We include it here to continue that well-established trend.

efficient encoding and decoding algorithms.

When it comes to defining what question(s) coding theory seeks to answer, the usual response is the one found in Guruswami, Rudra, and Sudan's 'Essential Coding Theory' [37]:

**Question 1.1.** *How much redundancy do we need to correct a given amount of errors?*

This question is posed in the language we've just discussed. 'Redundancy' means adding more than the necessary number of symbols in order to represent a given message. 'Errors' are discretely counted as the number of symbols in the received message that are incorrect or have been altered by the noise in the channel. Throughout, the assumption is that codewords are sequences of symbols being sent over a channel.

In this work, we seek to step away from many of the assumptions implicit in this standard treatment. We will rephrase this standard question in a way that escapes the confinement of the language of sequences and block codes.

## 1.3 Motifs of Coding Theory

We begin with the observation that all of the examples above and the standard textbook presentations contain variations on just three main motifs: space, size, and distance.

### 1.3.1 Space

The space $S$ is the set of potential codewords we're working with. For block codes, this space is $\Sigma^n$ for some $n \in \mathbb{N}$, and for the many more mathematically refined coding methods above, the set of symbols $\Sigma$ is taken to be $\mathbb{F}_2$ or more generally $\mathbb{F}_q$.

In fact, this space can be any space at all from the mathematical pantheon. Groups are fair game, as are sequences over them (i.e. direct products of these groups). In particular, this means that permutation groups form a perfectly valid space for coding[3]. Even spaces of manifolds are reasonable places to perform coding theory. This last example may seem unlikely to find practical applications, but we'll show later that this is actually not the case.

Just as it was with the traditional block codes, a code $C$ will be defined as a (possibly structured) subset of the space $S$. We'll denote the set of all acceptable codes as $\mathcal{C}$.

### 1.3.2 Size

On the set $\mathcal{C}$ we must have a function $\mathfrak{s} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ to represent the size of a code. In the block coding setting the size of a code is simply the number of codewords, but other size functions are also valid. For example, the size of a linear code can also be expressed by the dimension of the subspace forming the code.

The size of a code provides a measure of redundancy. When a code is very large, there is very little redundancy. For example, if our code consists of every binary string of length 7, then there is no redundancy, since every element we're able to represent is already a message. Smaller codes have more redundancy. The repetition code of length 7 contains only two codewords, 0000000 and 1111111, so there's a lot of redundancy. The Hamming code of length 7 falls somewhere in the middle[4].

[3]This example has seen some attention already.

[4]In these examples, we can certainly be more precise about how much redundancy there is (6 bits of redundancy for each 1 bit message in the repetition code and 3 bits of redundancy for every 4 bit message for the Hamming code), but we will encounter examples later in this work that illustrate how size is a more natural notion.

We'll take a quite general stance here and simply require that $\mathfrak{s}(C) \leq \mathfrak{s}(C')$ when $C \subseteq C'$ and $\mathfrak{s}(\emptyset) = 0$.

### 1.3.3 Distance

Finally, we'll require that $\mathcal{C}$ has a distance function $\mathfrak{d} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ to represent how 'self-avoiding' the code is. In traditional coding theory, Hamming distance is used to measure the number of symbols that differ between two codewords, and the distance measure for a code $C$ is the minimum Hamming distance between pairs of distinct codewords from $C$. In this setting the minimum distance determines the number of errors that can be corrected, since a code with minimum distance $d$ can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors. We will consider some settings where it doesn't make sense to ask how many errors can be corrected, so we argue that the distance function is more general.

We will only require that the distance function satisfies $\mathfrak{d}(C) \geq \mathfrak{d}(C')$ when $C \subseteq C'$ and $\mathfrak{d}(\emptyset) = \infty$.

## 1.4 The Fundamental Question

With these notions in mind, we establish the following definition.

**Definition 1.2.** *A* coding theory *consists of a set of acceptable codes $\mathcal{C}$ on the space $S$, a size function $\mathfrak{s}$, and a distance function $\mathfrak{d}$.*

The most important difference between the size function and the distance function is that as a code gets larger (in terms of subset containment), the size increases while the distance decreases. These two functions can be seen as fighting against each other,

and good codes are those that are near the threshold where the size cannot be increased without decreasing the distance and the distance cannot be increased without decreasing the size. This gives us our new fundamental question of coding theory, stripped of the original language of sequences and errors:

**Question 1.3.** *Given a coding theory* $(\mathcal{C}, \mathfrak{s}, \mathfrak{d})$*, how large can* $\mathfrak{s}(C)$ *be with a given* $\mathfrak{d}(C)$*?*

## 1.5 Outline

In the subsequent chapters, our goal is to formalize the intuition developed in this chapter, and establish examples showing the utility of this broader perspective on coding theory. We'll explain how traditional coding theory can be expressed in our new terminology. We'll show a limitation in trying to simply expand on the algebra behind traditional block coding settings by proving that block codes over nonabelian groups do not support a duality theory. From there, we'll move on to developing new coding theories designed for specific modern applications. We'll discuss permutation codes and their applications to flash memory. Finally, we'll develop an entirely new theory of coding curves with applications to time-of-flight cameras.

In chapter 2, we will formally define what we mean by a *coding theory* and explain the fundamental question of coding theory. We'll then briefly summarize what this looks like in the traditional block coding setting. We'll also take some time to summarize the key theorems of block coding theory that we will be generalizing and playing off of in later chapters. These include the Hamming and Singleton bounds, distance and weight enumerator polynomials, the interactions between weight enumerators and duality measured by the MacWilliams identity, and the Manin curve.

In chapter 3, we present our work solving an open problem of Dougherty, Kim, and Solé [22]. They asked whether there can be a duality theory and a MacWilliams identity for codes over nonabelian groups. We proved that, in fact, there do not exist any finite nonabelian groups $G$ with the property that $G^n$ has a self-dual subgroup lattice for all natural numbers $n$. In fact, our results can be seen as giving a criterion distinguishing abelian and non-abelian groups. That is, a finite group $G$ is abelian if and only if $G^n$ has a self-dual subgroup lattice for all $n$. We have previously published the work in chapter 3 in [50].

The remaining chapters will present new coding theories beyond the traditional block coding settings. We will cover three different permutaton coding theories in chapter 4, varying the metric used to measure the distance between permutations. We will cover the Hamming, Kendall tau, and Ulam metrics, summarizing for each how they relate to the application of rank modulation coding for flash memory and presenting the Hamming and Singleton bounds in those settings. In the case of the Hamming bound for the Kendall tau metric, a clearly stated version of this bound along with a formula for producing the sizes of Kendall tau balls has not yet appeared in the coding theory literature, so our presentation cleans up this gap in the literature. We will present a variety of weight enumerator polynomials in the Kendall tau and Ulam metrics along with other computational data we've prepared, showing the impact of relabeling on the weight distributions of codes in these two metrics. Our key theoretical contributions in this chapter are new formulas giving the maximum and minimum Kendall tau and Ulam weights of permutations under the relabeling operation given their cycle type. Included in this section is a triangle of numbers $W_n^k = 2\left\lfloor \frac{k}{2} \right\rfloor (n-k) + \left\lceil \frac{(k-1)^2}{2} \right\rceil$ counting the maximum number of inversions in a permutation consisting of a single $k$ cycle and

$n-k$ fixed points, which we believe are of somewhat general interest. We will also apply our theorems to give the minimum weights under relabeling of cyclic group codes under these metrics.

We will present an entirely new coding theory that we've developed in chapter 5. Inspired by research in the area of time-of-flight cameras as well as a more intuitive question about packing ropes into boxes, we've developed a theory of coding curves. We will borrow a definition of the *reach* of a curve from knot theory, and ask what is the maximum total length of curves of reach $r$ that can be packed into a unit cube in $n$ dimensions. We will use this chapter as a model for developing a new coding theory by establishing a formal definition that matches the problem we've presented, establishing Hamming bounds for this setting and checking for other standard bounds like the Singleton bound, and finding families of examples to test how close we can get to the bounds we've established for this coding theory. We will also prove a pair of theorems on the feasible regions for the shapes of these coding curves, and present a family of unknown curves similar to the Manin curve along with a handful of other open problems in this setting.

Finally, we will give a few final thoughts on the future of coding theory in chapter 6.

# Chapter 2

# Classical Theory

*'Clarity of vision, focus of intent, rectitude of thought: these will help in mathematics. If you don't have these three things, you're doomed.'*

(Paul Sally)

In this chapter, we'll briefly summarize the classical theory in the language we've introduced. We'll begin by defining the terms necessary for a mathematical discussion of coding theory. Then we'll provide a trivial example followed by some standard examples from traditional coding theory. Finally we'll provide a few of the standard theorems in those settings. The results we highlight here are those that we will prove new variations of later.

## 2.1 Basic Definitions

Recall the following definition from the introduction.

**Definition 2.1.** *A* coding theory *consists of a triple* $(\mathcal{C}, \mathfrak{s}, \mathfrak{d})$*, where*

- $\mathcal{C}$ *is a set of acceptable codes over some space* $S$*, i.e.* $\mathcal{C} \subseteq \mathcal{P}(S)$*.*
- *The size function* $\mathfrak{s} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ *has the properties that* $\mathfrak{s}(\emptyset) = 0$ *and* $\mathfrak{s}(C) \leq \mathfrak{s}(C')$ *when* $C \subseteq C'$*.*

- *The distance function* $\mathfrak{d} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ *has the properties that* $\mathfrak{d}(\emptyset) = \infty$ *and* $\mathfrak{d}(C) \geq \mathfrak{d}(C')$ *when* $C \subseteq C'$.

In cases where the triple is clear from context, we will refer to the coding theory simply by the name $\mathcal{C}$.

In the vast majority of traditional examples, the distance function is actually inherited from a distance function on $S$ as the minimum distance between distinct codewords. We'll refer to these as standard coding theories.

**Definition 2.2.** *A* standard coding theory *is a coding theory in which there is some metric* $d : S \times S \to \mathbb{R}_{\geq 0}$ *with the property that* $\mathfrak{d}(C) = \min_{\substack{x,y \in C \\ x \neq y}} d(x, y)$.

To better express the fundamental question of coding theory, we'll introduce one more piece of notation.

**Definition 2.3.**

$$A(\mathcal{C}, d) = \sup_{\substack{C \in \mathcal{C} \\ \mathfrak{d}(c) \geq d}} \mathfrak{s}(C)$$

In other words, $A(\mathcal{C}, d)$ is the largest code size that can achieve a distance of $d$. In many cases, we'll develop families of codes so that $\mathcal{C}$ depends on some other parameter, which we'll refer to as the dimension. So the values of $A(\mathcal{C}, d)$ should be seen as measuring the trade-offs of these three parameters: dimension, size, and distance. With this notation, we can now rephrase the fundamental question of coding theory as follows.

**Question 2.4** (Fundamental Question of Coding Theory)**.** *What is* $A(\mathcal{C}, d)$*?*

Astute readers will no doubt have noticed by this point that we've said almost nothing about actually correcting errors. We'll remedy that omission now.

**Definition 2.5.** *Given a code $C$ consisting of codewords from the space $S$, a* decision function *is a function $D : R \to C$, where $R \subseteq S$.*

Note that this definition doesn't actually say anything explicit about correcting errors. It simply says that there's some collection of receivable messages $R$ that can be mapped to elements of the code $C$. In practice, however, the most interesting and useful decision functions are those that actually do (under certain assumptions) have the effect of correcting the errors resulting from the noise of the channel. The choice of decision function depends not only on the structure of the code, but also on the structure of the noise. Many important secondary questions of coding theory arise from trying to find decision functions that have a high probability of decoding correctly for a given noise model and can also be implemented efficiently.

From this description, it may sound like the error-correcting ability of a code is just determined by the decision function, and has nothing to do with the distance function $\mathfrak{d}$. So what was the point of defining a distance function in the first place? The point is that the distance function should be chosen based on the noise model. A well-chosen distance function will have the property that when $\mathfrak{d}(C)$ is large, high magnitude noise in the current noise model can be tolerated well, and there exists a decision function that realizes this[1]. In many settings, there are theorems saying that minimum distance decoding is equivalent to maximum likelihood decoding; this can be interpreted as saying that the distance function chosen for that setting is accurately representing the noise model.

[1]Our phrasing here is necessarily a bit vague, since we no longer want to assume that 'tolerating noise well' means correcting errors perfectly. We'll encounter settings later where that highlight this distinction.

## 2.2 A Trivial Example

Not all coding theories are hard to understand. To emphasize this, our first example will be a coding theory for which we can easily solve the fundamental question.

Let $S = \{1, \ldots, n\}$, and let our set of codes be $\mathcal{C} = \mathcal{P}(S)$, the power set of $S$. We will define $\mathfrak{s}(C) = |C|$ and $\mathfrak{d}(C) = \begin{cases} \infty & \text{if } C = \emptyset \\ 1 & \text{otherwise} \end{cases}$.

**Theorem 2.6.** *For this trivial coding theory,* $A(\mathcal{C}, d) = \begin{cases} 0 & \text{if } 1 < d \\ n & \text{if } 0 \leq d \leq 1 \end{cases}$.

*Proof.* If $d > 1$, then the only code $C$ with $\mathfrak{d}(C) > d$ is $C = \emptyset$, which has $\mathfrak{s}(C) = 0$. On the other hand, if $0 \leq d \leq 1$, then every code $C \in \mathcal{C}$ has $\mathfrak{d}(C) \geq d$, so $A(\mathcal{C}, d) = n$, since the largest code $C = S$ has size $n$. □

## 2.3 Block Codes

For a much more practical and traditional example, we'll now turn to block codes.

**Definition 2.7.** *A* block coding theory *is a standard coding theory where*

- $S = \Sigma^n$ *for some* $n \in \mathbb{N}$ *and some finite alphabet of symbols* $\Sigma$.
- $\mathfrak{d}(C) = \min_{\substack{x,y \in C \\ x \neq y}} d(x, y)$, *where* $d$ *is the Hamming distance, i.e.* $d(x, y)$ *is the number of coordinates where* $x$ *and* $y$ *differ.*
- $\mathfrak{s}(C) = |C|$, *i.e. the size of a code is just the number of codewords.*

Some authors do not include the Hamming metric in the definition of block codes, and in fact there are many other metrics such as the Lee metric[2] and the Levenshtein metric[3] that are equally valid and useful in certain circumstances. Our decision to include the Hamming metric in our definition will help to simplify our language a bit. When we say 'block code' we will always be referring to codes with the Hamming metric. This definition will also help us avoid a possible pitfall. If block codes are defined only as a set of elements in $\Sigma^n$, then every code would be a block code for some sufficiently broad choice of alphabet $\Sigma$. This is similar to how every cipher can be thought of as a monoalphabetic substitution cipher for some sufficiently large alphabet. It's true, but it's not a useful perspective and it confuses the very important distinctions between different settings that our choice of terminology is meant to clarify.

Even after choosing an alphabet $\Sigma$, the parameter $n$ still allows us to vary the length of our codes. As we mentioned earlier, since $n$ is now a third parameter to balance along with the size and the distance of a code, in this context we'll often write $A(n, d)$ instead of $A(\mathcal{C}, d)$ when the family of codes is clear. To say much more, it is useful to impose an algebraic structure on $\Sigma^n$.

**Definition 2.8.** *A* linear block coding theory *is a block coding theory where* $\Sigma = \mathbb{F}_q$ *and* $\mathcal{C}$ *contains only linear subspaces of* $\mathbb{F}_q^n$.

Since the number of codewords in a linear block code of dimension $k$ is always $q^k$, we could also choose to measure the size of a linear block code by it's dimension. Setting

[2]This takes into account how far an individual symbol has been shifted when the alphabet is $\mathbb{F}_q$, and agrees with the Hamming metric when $q = 2$ or $q = 3$.

[3]This measures the number of single-symbol edits (insertions, deletions, and substitutions) required to change one string into another.

$\mathfrak{s}(C) = k$ for a linear block code of dimension $k$ would lead to an equivalent definition of linear block coding theory[4].

**Definition 2.9.** *An* abelian group block coding theory *is a block coding theory where* $\Sigma = G$ *for some abelian group* $G$ *and* $\mathcal{C}$ *contains only subgroups of* $G^n$.

We'll refer back to these two examples as we explore some of the standard theorems of coding theory. The remainder of this chapter will summarize some of the most important results in the theory of block codes and especially linear block codes. In the subsequent chapters, we will see variations on each of these theorems in other coding theories.

## 2.4 Standard Bounds

For the most part, it is extremely difficult to nail down the exact values of $A(\mathcal{C}, d)$ for a sufficiently complex coding theory. Instead, much of the effort to understand $A(\mathcal{C}, d)$ has been redirected to determining upper and lower bounds. Every example of a code $C$ with a known $\mathfrak{d}(C)$ and $\mathfrak{s}(C)$ gives a lower bound for $A(\mathcal{C}, d)$. Upper bounds, on the other hand, come from a variety of different and often insightful arguments. We'll highlight a couple of the most basic bounds here.

### 2.4.1 Hamming Bound

Since the distance function of a block code comes from a metric on pairs of codewords, we can study how balls around codewords under this metric interact. If a block code

[4]To be precise with the size function, we should not allow the empty set to be a linear block code in this context.

$C$ has $\mathfrak{d}(C) = d$, then this means that balls of radius $\lfloor \frac{d-1}{2} \rfloor$ around the codewords of $C$ must be disjoint. The number $\lfloor \frac{d-1}{2} \rfloor$ is known as the *packing radius* of the code. Counting the total size of these balls and comparing it to the size of the space $\Sigma^n$ gives us the Hamming bound for block codes:

**Theorem 2.10** (Hamming Bound)**.** *For a block code on an alphabet $\Sigma$ with $|\Sigma| = q$,*

$$A(n, d) \leq \frac{q^n}{\sum_{k=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{k} (q-1)^k}.$$

Because of the geometric intuition of this bound, it's often referred to as the sphere-packing bound. In fact, for any standard coding theory and many other coding theories where the distance function depends on a metric on $S$, there is some sphere-packing or volume-based bound on the size of $A(\mathcal{C}, d)$. We'll see other examples of this later, but for now we'll give a general form for standard coding theories.

**Theorem 2.11** (Hamming Bound)**.** *Let $(\mathcal{C}, \mathfrak{s}, \mathfrak{d})$ be a standard coding theory with $\mathfrak{s}(C) = |C|$ and $\mathfrak{d}(C) = \min_{\substack{x,y \in C \\ x \neq y}} d(x, y)$. Suppose that every ball of radius $r$ contains the same number of points*[5]*, and call this number $B(r)$. Then*

$$A(\mathcal{C}, d) \leq \frac{|S|}{B\left(\lfloor \frac{d-1}{2} \rfloor\right)}.$$

Codes that meet this Hamming bound are referred to as *perfect codes*. We've already mentioned two of these. The Hamming code and the Golay code are perfect codes. In fact, there is an entire family of higher dimensional Hamming codes with $n = 2^m - 1$, $d = 3$, and $\mathfrak{s}(C) = 2^{2^m - m - 1}$ and a ternary Golay code with $n = 11$, $d = 5$, and $\mathfrak{s}(C) = 3^6$

[5]This is true, for example, of (left- or right-) translation invariant metrics.

that are also perfect. Tietäväinen [83] was able to show that those are the only non-trivial[6] perfect codes over finite fields.

Each of these bound-attaining codes tells us a value of the $A(n, d)$ function that we've established as the primary target of coding theory. For binary linear block codes, these perfect codes give us the following values.

| Perfect code | $A(n, d)$ |
|---|---|
| $\{0^n\}$ | $A(n, \infty) = 1$ |
| $\mathbb{F}_2^n$ | $A(n, 1) = 2^n$ |
| Odd-length repetition codes | $A(2m+1, 2m+1) = 2$ |
| Hamming codes | $A(2^m - 1, 3) = 2^{2^m - m - 1}$ |
| Binary Golay code | $A(23, 7) = 2^{12}$ |

Table 1: $A(n, d)$ values for the theory of binary linear block codes given by perfect codes.

### 2.4.2 Singleton Bound

Another strategy one can take to bound $A(\mathcal{C}, d)$ is to begin by mapping a code $C$ with distance at least $d$ into a smaller space. If we can guarantee that the images of the codewords will be distinct, or if we can control the number of preimages, then we can count the size of this – hopefully much smaller – space and find a restriction on the size of $C$. Bounds of this type are called Singleton bounds after the first use of this strategy for block codes by Richard Singleton [77].

The block code Singleton bound is obtained by puncturing the code by removing the first $d - 1$ coordinates. Since the distance between the original codewords was at least $d$, their images are still distinct, but they now sit within a space that only has $q^{n-d+1}$

[6]The trivial perfect codes are the codes consisting of just a single codeword, the code that contains the entire space $\Sigma^n$, and odd-length repetition codes over $\mathbb{F}_2$.

points. Later we will see a version for permutation codes that uses a similar strategy to obtain a very different looking bound.

**Theorem 2.12** (Singleton Bound)**.** *For a block code on an alphabet* $\Sigma$ *with* $|\Sigma| = q$*,*

$$A(n, d) \leq q^{n-d+1}.$$

As we mentioned earlier, the size of a linear block code can equivalently be measured by the dimension of the code as a linear subspace of $\mathbb{F}_q^n$. With this size function, the Singleton bound for linear block codes takes the slightly simpler form $A(n, d) \leq n-d+1$.

Codes that meet the Singleton bound are referred to as *maximum distance separable (MDS)* codes[7]. Over a binary alphabet, the only MDS codes are the code containing all of $\mathbb{F}_2^n$, single-parity-check codes, and repetition codes [85]. There are other MDS codes, including the Reed-Solomon codes we mentioned earlier, over larger alphabets.

Once again, these bound-attaining examples tell us certain values of the $A(n, d)$ functions. The MDS binary linear block codes give us the following.

| MDS code | $A(n, d)$ |
|---|---|
| $\mathbb{F}_2^n$ | $A(n, 1) = 2^n$ |
| Single-parity-check codes | $A(n, 2) = 2^{n-1}$ |
| Repetition codes | $A(n, n) = 2$ |

Table 2: $A(n, d)$ values for the theory of binary linear block codes given by MDS codes.

[7]In other coding theories, codes meeting Singleton bounds are typically referred to as *Singleton optimal*.

## 2.5 Distance and Weight Enumerators

The bounds of the previous section help us to understand the values of $A(n, d)$, which we have presented as the primary goal of coding theory. The standard codes we've been discussing so far in this chapter, however, have many more distances between codewords beyond just the minimum distance for the code. In fact, more complete information about the distances between the codewords is often much more useful than just a minimum distance, since a code with a few close codewords but otherwise very well spread out can still have very good error-correcting capabilities on average. The full spectrum of the pairwise distances between the codewords of a standard code is typically represented by an enumerator polynomial.

**Definition 2.13.** *The* distance enumerator polynomial *of a finite standard code $C$ is*

$$D_C(x, y) = \sum_{i=0}^{N} A_i x^i y^{n-i},$$

*where $A_i = \frac{1}{|C|}\#\{(c_1, c_2) \in C \times C | d(c_1, c_2) = i\}$ and $N$ is the maximum pairwise distance between elements of $S$.*

For linear block codes, it's unnecessary to independently compute all of the pairwise distances between codewords, since the same information can be determined by simply computing the distances between each codeword and the origin.

**Definition 2.14.** *The* weight *of a codeword $c$ in a linear block code $C$ is $w(c) = d(c, 0)$ where $d$ is the Hamming distance.*

Observe that $d(c_1, c_2) = d(c_1 - c_2, 0) = w(c_1 - c_2)$. So for linear block codes we can rewrite the distance enumerator polynomial as follows.

**Definition 2.15.** *The* weight enumerator polynomial *of a linear block code $C$ of length $n$ is*

$$W_C(x, y) = \sum_{i=0}^{n} A_i x^i y^{n-i},$$

*where $A_i = \#\{c \in C | w(c) = i\}$.*

The weighting factor $\frac{1}{|C|}$ we used earlier in the definition of distance enumerator polynomials was chosen so that the distance enumerator polynomial for a linear block code is equal to the weight enumerator polynomial.

While there are a variety of useful theorems involving the weight enumerator polynomials that can help us understand the weight distributions of codes with various properties, we will highlight only one: the MacWilliams theorem. This theorem gives us an identity connecting the weight enumerator of a linear block code with the weight enumerator of its dual code. So let's turn now to briefly summarize duality for linear block codes.

## 2.6 Duality

Since linear block codes are simply vector subspaces of $\mathbb{F}_q^n$ and $\mathbb{F}_q^n$ already has a built-in bilinear form, $\langle a, b \rangle = \sum_{i=1}^{n} a_i b_i$, it's natural to define a duality relation as follows.

**Definition 2.16.** *Given a linear block code $C \subseteq \mathbb{F}_q^n$, the* dual code *to $C$ is the code*

$$C^{\perp} = \{x \in \mathbb{F}_q^n | \langle x, c \rangle = 0 \ \forall c \in C\}.$$

Let's examine how this relation applies to the binary linear block codes of length 3. In Figure 2, we can see the 16 binary linear block codes of length 3. The edges in the

lattice represent the containment relationships between the codes, and we've arranged the diagram so that the codes that lie symmetrically above and below each other are dual to each other. For example, the code generated by 001 is dual to the code generated by 100 and 010, and the code $\mathbb{F}_2^3$ is dual to the code $\{0\}$.

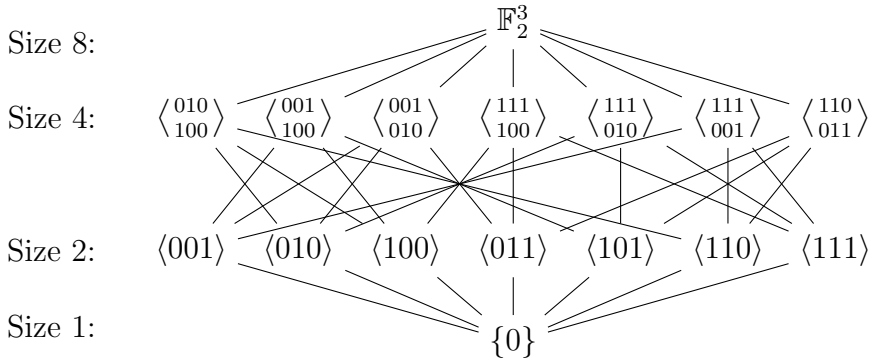


Figure 2: The linear block codes of length 3 over $\mathbb{F}_2$

Although the duality relationship was determined by the bilinear form on the vector space $\mathbb{F}_q^n$, it's important to see the full lattice of length 3 linear block codes over $\mathbb{F}_2$ in order to understand that the lattice structure is essential for allowing a duality theory to exist. In general, for a coding theory to admit a duality theory, it's necessary that the lattice of codes (related by containment) is self-dual, in the sense that the lattice structure obtained by reversing all of the containments is the same. Given this structure, the mapping that realizes the self-duality of the lattice also determines the dual pairs of codes.

Block codes over abelian groups also have self-dual containment lattices, so dual codes also exist in that setting. In fact, the duality theory for block codes over an abelian group $G$ is also given by an inner product using essentially the same definition as above, but the inner product used is actually an inner product on the characters of

the group $G$.

We will see more on the importance of self-dual lattices in chapter 3.

## 2.7 MacWilliams Theorem

In 1963, Jessie MacWilliams showed that there is a remarkable relationship between the weight distributions of dual linear block codes [59]. Nowadays it is common to write her theorem as a statement on the weight enumerator polynomials as follows.

**Theorem 2.17** (MacWilliams)**.** *Let $C$ be a linear block code of length $n$ over $\mathbb{F}_q$. Then*

$$W_{C^\perp}(x,y) = \frac{1}{|C|} W_C(y-x,(q-1)x+y).$$

This identity provides a powerful tool for understanding weight distributions of codes. In addition to providing a straightforward way of determining the weights of large codes that are dual to much simpler and smaller codes, it also provides restrictions on the weight structures of self-dual codes.

We can see the usefulness of the MacWilliams identity for determining weight distributions of large codes using the example of the Hamming codes. Although the full weight enumerators of the Hamming codes can be somewhat complicated, the Hamming codes are dual to a very simple family of codes, the simplex codes, $\Sigma_m$, with $n = 2^m - 1$, $d = 2^{m-1}$, and $\mathfrak{s}(C) = 2^m$. Unlike the Hamming codes, the simplex codes always have a very simple weight distribution: every codeword other than the identity has exactly the same weight! So the weight enumerator for the $m^{\text{th}}$ simplex code is just

$$W_{\Sigma_m}(x,y) = (2^m-1)x^{2^{m-1}}y^{2^{m-1}-1} + y^{2^m-1}.$$

We can then produce the full (and much more complicated) weight enumerator for the Hamming code, $H_m$ dual to to $\Sigma_m$ by just applying the MacWilliams identity.

$$W_{H_m}(x,y) = \frac{1}{2^m} W_{\Sigma_m}(y-x, x+y).$$

We never have to actually compute the weights of any individual Hamming codewords.

Many other forms of the MacWilliams identity have been discovered in the last few decades. These include a version for formal duals of non-linear codes [58] and versions for codes over abelian groups and modules over finite rings [89]. As an example of these generalizations, here's the version for abelian group block codes[8].

**Theorem 2.18.** *Let $C$ be an abelian group block code of length $n$ over $G$. Then*

$$W_{C^\perp}(x,y) = \frac{1}{|C|} W_C(y-x, (|G|-1)x+y).$$

As you can see, this generalized form of the theorem is extremely similar looking to MacWilliams' original version. Although most versions of the MacWilliams theorem take very similar forms, any theorem providing a linear transformation relating the weight distributions of pairs of dual codes is typically thought of as a MacWilliams identity.

## 2.8 Manin Curve

In order to compare block codes of different lengths, it's useful to look simultaneously at the *rate* of the code $R = \frac{k}{n}$ and the *relative distance* $\delta = \frac{r}{n}$. This allows us to plot a point for each block code (regardless of its length) on a single $(\delta, R)$ graph. Suppose that

[8]The form of the theorem we use here is based on the exposition of Wood [89], but Wood attributes this theorem in an earlier form to Delsarte [16].

for some alphabet size, $q$, we plot a point for every single block code over that alphabet. Manin was able to show that for every $q$ there is some continuous decreasing function $R = \alpha_q(\delta)$ with the property that the points corresponding to block codes are dense below $\alpha_q$ with only isolated points above [61, 87]. One of the key questions of coding theory is to identify this function. Goppa has conjectured that in the binary case this curve is $\alpha_2(\delta) = 1 - H(\delta)$ for $\delta < \frac{1}{2}$, where $H$ is the binary entropy function, but this remains an open question [35].

# Chapter 3

# Codes over Nonabelian Groups

*'The mathematicians have been very much absorbed with finding the general solution of algebraic equations, and several of them have tried to prove the impossibility of it. However, if I am not mistaken, they have not as yet succeeded. I therefore dare hope that the mathematicians will receive this memoir with good will, for its purpose is to fill this gap in the theory of algebraic equations.'*

(Niels Henrik Abel)

In the spirit of repeatedly generalizing block codes, we'll continue by expanding the types of groups that we might form block codes over.

**Definition 3.1.** *A* nonabelian group block coding theory *is a block coding theory where* $\Sigma = G$ *for some nonabelian group* $G$ *and* $\mathcal{C}$ *contains only subgroups of* $G^n$.

Of course, all the usual questions about $A(n, d)$ and decision functions for various noise models can be asked in this setting. For example, the Hamming and Singleton bounds from section 2.4 certainly still hold in this setting, since nonabelian group block codes are still block codes. We can also ask whether the duality theory and the MacWilliams identity developed for various block codes in section 2.6 and section 2.7 can be generalized in this setting. As we saw, the traditional MacWilliams identity was developed for linear block codes, but it's been shown that a version exists for block codes

over abelian groups as well. In an open problems paper in 2015, Dougherty, Kim, and Solé [22] asked whether there exists a duality theory and MacWilliams identity for block codes over nonabelian groups or at least for some class of nonabelian groups. We will show that, in fact, there are no nonabelian groups over which a block coding theory can have a duality theory or a MacWilliams identity [50].

## 3.1 Formulating a Strategy

When we encountered duality for linear block codes and for block codes over abelian groups in section 2.6, we were able to produce dual codes by using an inner product. Given a code $C \in \mathcal{C}$, we would define $C^{\perp} = \{x \in S | \langle x, c \rangle = 0 \ \forall c \in C\}$, where the inner product was taken to be the usual inner product for vectors over $\mathbb{F}_q^n$ for linear block codes and it was the inner product for characters of $G$ for block codes over an abelian group $G$. With a duality theory in hand, we could then ask whether there was a MacWilliams identity and what form it might take. This might seem to suggest that we should begin our search for a duality theory of block codes over nonabelian groups by asking questions about inner products.

This approach would lead to some difficulties. Even if we could produce candidate inner products, proving that one of them does not lead to a proper duality theory for block codes over some class of nonabelian groups would only tell us that one particular choice didn't work. Instead, we begin by asking what conditions are necessary for a duality theory in the first place.

The key condition, which we saw foreshadowed in Figure 2, is that the lattice of codes must be self-dual. In other words, the philosophy that we take is that an inner

product can produce a duality only if the subgroup lattice allows it. Our goal in the remainder of this chapter will be to show that there are no finite nonabelian groups $G$ with the property that $G^n$ has a self-dual subgroup lattice for all $n$.

## 3.2 Obstructions to Duality

When Dougherty, Kim, and Solé posed the question, they already knew that the subgroup lattices might cause a problem. They noted that the quaternion group $Q_8$, one of the most basic nonabelian groups, already has a subgroup lattice that is very non-symmetric.

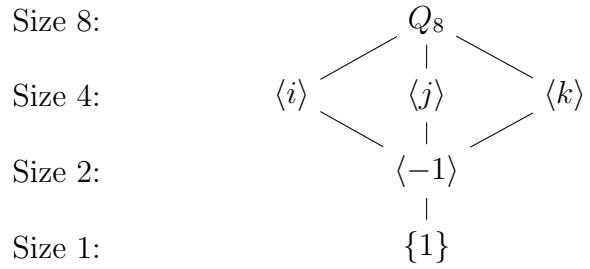


Figure 3: The subgroup lattice for the quaternion group $Q_8$

As we can see in Figure 3, $Q_8$ has three subgroups of order 4, $\langle i \rangle$, $\langle j \rangle$, and $\langle k \rangle$, but only one subgroup of order 2, $\langle -1 \rangle$. So if we expect block codes over $G$ and their duals to be subgroups of $G^n$ with the property that $|C||C^{\perp}| = |G|^n$, then we would need to restrict our attention to some subclass of nonabelian groups that does not include the quaternions. In particular, we'll need to determine which nonabelian groups have self-dual subgroup lattices.

Fortunately, subgroup lattices with duals have already been classified. In the 1950's, Suzuki [81] determined which finite solvable groups have duals, and Zacher [91] was able

to prove ten years later that all finite groups with duals are solvable. We will apply this classification to show that if $G$ is a finite nonabelian group with a self-dual subgroup lattice, then $G \times G$ will not have a self-dual subgroup lattice. As a corollary, there cannot exist any nonabelian finite group $G$ with the property that $G^n$ has a self-dual subgroup lattice for all $n$. While we can certainly define block codes over a nonabelian group $G$ to be subgroups of $G^n$, there is no class of nonabelian groups that will support a duality theory with this definition. As a result, the MacWilliams identity that is so useful for studying block codes over finite fields and abelian groups does not exist in this setting.

## 3.3 Self-dual Subgroup Lattices

Let's begin with a quick tour through the relevant terminology for discussing subgroup lattices.

**Definition 3.2.** *Let $L(G)$ denote the subgroup lattice of a group $G$. We say $G$ has a dual group $\bar{G}$ if there exists a bijective map $\delta : L(G) \to L(\bar{G})$ such that for all $H, K \in L(G)$, $H \leq K$ if and only if $\delta(K) \leq \delta(H)$.*

We are interested in groups that are self-dual, i.e. groups $G$ with a dual $\bar{G} \cong G$. Observe that if $G^n$ was self-dual, then by defining a code over $G$ to be a subgroup of $G^n$, the duality map for $G^n$ would take a code $C$ to a dual code $C^\perp$. This is the same arrangement that produces dual codes over finite fields and abelian groups, but in those cases the duality map can be produced by appealing to an inner product. Without an inner product structure to fall back on for nonabelian groups, we instead depend on studying just the subgroup lattice structure. We will make use of the classification of finite groups with duals given in Schmidt's book on subgroup lattices [75].

**Theorem 3.3.** *A finite group $G$ has a dual if and only if $G$ is a direct product of coprime groups $G_\lambda$ such that each $G_\lambda$ is a $P$-group or a non-Hamiltonian $p$-group with a modular subgroup lattice.*

This theorem gives us a handle for understanding which finite nonabelian groups have duals. In particular, if $G$ is self-dual, it must satisfy this condition, so we can proceed by asking which nonabelian groups $G$ can produce direct products $G^n$ that satisfy this condition. First, however, we'll need to understand the specialized terminology in the theorem statement.

The theorem refers to two different types of groups that can appear in a direct product to produce $G$. The first type is called a $P$-group.

**Definition 3.4.** *A finite group $G$ is called a* $P$-group *if it is:*

1. *an elementary abelian group of prime power order, or*
2. *a group which can be decomposed as $S_pS_q$, where $S_p$ is a $p$-Sylow subgroup which is an abelian $P$-group, $S_q$ is a cyclic $q$-Sylow subgroup of order $q$, $S_q = \langle B \rangle$, and for any element $A \in S_p$ we have*

$$BAB^{-1} = A^r, \;\; r \not\equiv 1, \;\; r^q \equiv 1 \pmod{p}.$$

The second type of group $G_\lambda$ has three descriptors: non-Hamiltonian, $p$-group, and a group with a modular subgroup lattice. We'll list each of these definitions separately.

**Definition 3.5.** *A* Hamiltonian group *is a nonabelian group $G$ such that every subgroup of $G$ is normal.*

The smallest example of a Hamiltonian group is the quaternion group $Q_8$ that we met earlier, and, in fact, Dedekind [15] showed that every finite Hamiltonian group is the direct product of the quaternion group with some other abelian group. Since we've already seen that the subgroup lattice of the quaternions is not self-dual, it should not be particularly surprising that the theorem we're planning to apply specifically directs our attention away from such groups.

**Definition 3.6.** *A $p$-group is a group in which the order of every element is a power of $p$.*

**Definition 3.7.** *A modular lattice is a lattice that satisfies the condition $x \leq b$ implies $x \vee (a \wedge b) = (x \vee a) \wedge b$ for any lattice element $a$, where $\vee$ and $\wedge$ are the join and meet operations on the lattice. Groups with modular subgroup lattices are called Iwasawa groups.*

Fortunately for us, $p$-groups with modular subgroup lattices have already been classified by Iwasawa [45], so we will not need to work too much with these definitions directly. We include them primarily to aid in understanding the statement of Theorem 3.3.

## 3.4 Direct Products

In order to use Theorem 3.3 to understand block codes over a nonabelian group $G$ (i.e. subgroups of $G^n$), we must understand the direct products of $P$-groups and of non-Hamiltonian $p$-groups with modular subgroup lattices with themselves. We'll examine these first before tackling the main theorem.

**Lemma 3.8.** *If $G$ is a nonabelian $P$-group, then $G \times G$ is not a $P$-group.*

*Proof.* From the definition above, a nonabelian $P$-group must be the semidirect product of an elementary abelian group of order $p^k$ and a cyclic group of prime order $q$ (along with some additional structure). In particular, $|G| = p^k q$, so $|G \times G| = p^{2k} q^2$. Then $G \times G$ cannot be a $P$-group, since the order of nonabelian $P$-groups must be of the form $p^\ell q$ for primes $p$ and $q$. □

To handle the case where $G$ is a non-Hamiltonian $p$-group with a modular subgroup lattice, we will apply a theorem of Iwasawa [45].

**Theorem 3.9.** *If $G$ is a non-Hamiltonian nonabelian $p$-group with a modular subgroup lattice, then $G$ contains an abelian normal subgroup $N$ such that $G/N = \langle q \rangle$ is cyclic and for all $n \in N$, $q^{-1}nq = n^{1+p^s}$, where $s \geq 1$ ($s \geq 2$ if $p = 2$).*

In fact, to show that this property cannot hold for both $G$ and $G \times G$, we actually only need an abelian normal subgroup with a cyclic quotient. The further structure described in Iwasawa's theorem is unnecessary for the following lemma.

**Lemma 3.10.** *If $G$ is a nonabelian group, then $G$ and $G \times G$ cannot both be non-Hamiltonian $p$-groups with modular subgroup lattices.*

*Proof.* Suppose that both $G$ and $G \times G$ are non-Hamiltonian $p$-groups with modular subgroup lattices. Let $N$ be an abelian normal subgroup of $G \times G$ such that $(G \times G)/N$ is cyclic, and let $p_1$ and $p_2$ be the standard projections $G \times G \to G$. Observe that if $p_1(N) \neq G$ and $p_2(N) \neq G$, then $(G \times G)/N$ cannot be cyclic, since $(G \times G)/N$ would have a non-cyclic quotient. In particular, since $N \subseteq p_1(N) \times p_2(N)$ and $p_1(N)$ and $p_2(N)$ are normal in $G$, we have

$$((G \times G)/N)/((p_1(N) \times p_2(N))/N) \cong (G \times G)/(p_1(N) \times p_2(N))$$

$$\cong (G/p_1(N)) \times (G/p_2(N)),$$

and since $p_1(N) \neq G$ and $p_2(N) \neq G$, we claim that $(G/p_1(N)) \times (G/p_2(N))$ is not cyclic.

To confirm this claim, we first observe that $G/p_1(N)$ and $G/p_2(N)$ are $p$-groups of orders $p^a$ and $p^b$, respectively, for some integers $a$ and $b$. Since $p_1(N) \neq G$ and $p_2(N) \neq G$, we know that $a, b > 0$. Then $(G/p_1(N)) \times (G/p_2(N))$ is a group of order $p^a p^b = p^{a+b}$. Suppose that $g_1 \in G/p_1(N)$ and $g_2 \in G/p_2(N)$ with $|g_1| = p^c$ and $|g_2| = p^d$. Then $(g_1, g_2) \in (G/p_1(N)) \times (G/p_2(N))$ and $|(g_1, g_2)| = \text{lcm}(|g_1|, |g_2|) = \text{lcm}(p^c, p^d) = p^{\max(c,d)}$. Since $c \leq a$, $d \leq b$, and $a, b > 0$, we have that $p^{\max(c,d)} < p^{a+b}$, so $(g_1, g_2)$ cannot generate $(G/p_1(N)) \times (G/p_2(N))$. Since no element of $(G/p_1(N)) \times (G/p_2(N))$ can generate the entire group, $(G/p_1(N)) \times (G/p_2(N))$ cannot be cyclic.

So, for some $i \in \{1, 2\}$, $p_i(N) = G$. But this contradicts the existence of such an abelian normal subgroup $N$, since $N$ was chosen to be abelian while $G$ is nonabelian and abelian groups cannot project onto nonabelian groups. □

There is already a result in the literature [13] that states the following: a nonabelian non-Hamiltonian $p$-group $P = P_1 \times P_2$ is an Iwasawa group if and only if $P_1$ and $P_2$ are Iwasawa and, for $i = 1$ or $i = 2$, $P_i$ is abelian such that $\text{Exp}(P_i) \leq p^s$ and $s$ is the integer that comes from the nonabelian factor $P_j$ for $j \neq i$. Here $\text{Exp}(G)$ denotes the exponent of the group $G$. Our second lemma would be a direct corollary of this result. Unfortunately, the published proof is incorrect, since after establishing elements of $P$ such that $x_1^k a_1^\ell a_2^\ell = x_1 a_1^{1+p^{s_1}} a_2$, the author claims without any further justification that this implies that either $\ell = 1$ or $\ell = 1 + p^{s_1}$. Without this step, there remains an error in the published proof, so our proof above can be considered a partial correction of that

result.

## 3.5 No Duality for Codes Over Nonabelian Groups

With these lemmas in hand, we are now ready to prove the main theorem.

**Theorem 3.11.** *If $G$ is a finite nonabelian group with a self-dual subgroup lattice, then $G \times G$ does not have a self-dual subgroup lattice.*

*Proof.* First, if $G$ is a finite nonabelian group with a self-dual subgroup lattice, then by Theorem 3.3, $G$ can be expressed as $G = \bigoplus G_\lambda$, where the $G_\lambda$ are coprime and each $G_\lambda$ is either a $P$-group or a non-Hamiltonian $p$-group with a modular subgroup lattice. Then $G \times G = \bigoplus(G_\lambda \times G_\lambda)$, and since the $G_\lambda \times G_\lambda$ are still coprime, it suffices to check whether it is possible for each $G_\lambda \times G_\lambda$ to still be a $P$-group or a non-Hamiltonian $p$-group with a modular subgroup lattice. By Theorem 3.8, we know that this will not work if $G_\lambda$ is a $P$-group, and by Theorem 3.10, we know that this will also not work if $G_\lambda$ is a non-Hamiltonian $p$-group with a modular subgroup lattice. Thus we can conclude that if $G$ is a finite nonabelian group with a self-dual subgroup lattice, then $G \times G$ cannot also have a self-dual subgroup lattice. □

We can now apply this result to our original question about block codes over nonabelian groups.

**Corollary 3.12.** *There does not exist a finite nonabelian group $G$ with the property that $G^n$ has a duality theory for all $n$.*

*Proof.* Theorem 3.11 shows that already both $G$ and $G \times G$ cannot have a duality theory, so certainly this cannot hold for $G^n$ for all $n$. □

For a more positive sounding result, we can also rephrase this theorem as follows.

**Corollary 3.13.** *If $G$ is a finite group with the property that $G^n$ has a self-dual subgroup lattice for all $n$, then $G$ is abelian.*

## 3.6 Remaining Questions

The results in this chapter show that there is no class of nonabelian groups whose block coding theory supports a duality theory. However, some variations of Dougherty, Kim, and Solé's question remain open. For example, one could change our definition so that codes over $G$ are not restricted to be subgroups of $G^n$, and ask whether some other collection of codes could produce meaningful self-dual lattices. Another possibility would be to relax our demands for the duality map. Although our work goes a long way towards describing the boundary between classes of block codes with and without duality theories, the most general form of Dougherty, Kim, and Solé's question, "Find the largest class of algebraic structures $A$ for which a duality and MacWilliams relations hold" remains a target for future research.

# Chapter 4

# Permutation Codes

*'The purpose of computation is insight, not numbers.'* (Richard Hamming)

In this chapter and chapter 5 we will leave the world of block codes behind, introducing new spaces, size functions, and distance functions in which we can ask the standard coding theory questions. In each case, we will see how new and emerging technologies lead to the development of novel coding theories for mathematicians to explore.

## 4.1 Rank Modulation Coding for Flash Memory

Flash memory was developed in the 1980's as a non-volatile electronic memory storage technology that is electrically programmable and electrically erasable. In other words, it's a form of reprogrammable computer memory that will not lose the information stored when the power supply is shut off. As the technology has matured, it's made its way into a wide variety of consumer technologies, including USB flash drives, flash memory cards for cameras and cell phones, and solid-state hard drives.

Within a flash memory chip, information is stored by controlling the charge levels on individual cells that are organized into larger blocks. Since the charge levels on an individual cell can vary across a continuous range, some encoding method is required to

decide how the information being stored is actually represented on the chip. Traditionally, this is done by partitioning each cell into charge ranges representing different values and defining thresholds between these ranges that should be avoided to limit the number of errors that arise as the charges drift and leak over time. In single-level cell (SLC) chips, there is a single threshold separating a range representing 1 from a range representing 0. Multi-level cells (MLC) have more thresholds to encode more information into a single cell; e.g. three thresholds are enough to produce four ranges representing 11, 01, 00, and 10 for two bits of information on each cell[1]. Although MLC flash chips can store more information per cell, their smaller charge ranges also lead to a higher likelihood of error.

In this storage model, a collection of cells is used to store a binary string and the errors that arise from charge leaks and mis-reads result in bit-flips. This is exactly the setting that traditional binary block codes were designed for. Most SLC chips use Hamming codes to manage errors, while MLC systems typically use Reed-Solomon or BCH codes.

But there is a problem with just directly applying these old methods to the new setting of flash memory. We do not actually have total freedom to adjust the charge levels of each cell however we'd like. Instead, there is an asymmetry between cell programming (increasing charges) and cell erasing (removing charges). Cells can be programmed individually, allowing us to increase the charges to whatever levels we'd like (at least up to our physical ability to measure and adjust the charges), but cell erasures can only be done in large blocks. So if we ever need to decrease the charge on even just one cell

[1]The order in which we listed these symbols represents how they are typically represented on a cell. Gray codes are used to ensure that each pair of adjacent symbols differs only in one bit.

to store different information or to amend an error we've detected, we would actually need to clear and reprogram an entire block, typically representing several kilobytes of memory. Worse still, every block erasure does a small amount of damage to the chip. Most flash memory chips today are only able to withstand about 10,000-100,000 block erasures (and sometimes far fewer).

Most consumers don't notice this limitation simply because the current applications for flash memory are chosen as settings where this limit won't likely be hit. That's a major reason why flash memory is most often seen in technologies like USB thumb drives and camera memory cards that are accessed and edited on human timescales. For example, a photographer might reuse a memory card dozens or even hundreds of times, but is unlikely to hit a 10,000 erasure limit. If we want to expand the use of flash memory to applications with more frequent edits like RAM that won't lose its data after losing power, we'd need a way to allow far more read/write cycles.

It was with this problem in mind that Jiang, Mateescu, Schwartz, and Bruck introduced rank modulation for flash memory in 2008 [48, 46, 47]. Their idea was to store information on flash memory chips not as binary strings, but instead as permutations in $S_n$. More precisely, after fixing $n$, a collection of $n$ cells can store a permutation as the ranking of the charge values on these cells. Since the charge ranking can be changed by only increasing charges on the cells, this coding method would reduce the number of block erasures.

Rank modulation coding offers several potential advantages for the flash memory setting. Since the charge ranking doesn't depend on the magnitude of the charges, it's no longer necessary to use charge thresholds. This allows us to more freely use the full capacity range of the cells and avoids charge overshoot issues that can arise in MLC

coding, where a programming step may unintentionally increase a charge too much and cross the threshold above the intended symbol's designated range. By taking care to only increase charges as much as is necessary to represent a given permutation, we can update the information on a flash memory chip using rank modulation several times before requiring an erasure step. Rank modulation coding is also better suited to the error patterns of flash memory. Over long term storage, most errors in traditional flash memory encodings arise from charge leaks [9], where the stored charges gradually drop, eventually crossing a threshold and entering a range representing a different symbol. Rank modulation coding is resistant to this sort of error as long as the charges in a collection of cells are leaking in a relatively uniform fashion.

Several challenges must be overcome in order to use rank modulation coding effectively. Although we will not focus on it here, there are engineering problems that must be solved: redesigning chips to use pairwise comparisons to determine permutations instead of threshold comparisons, finding efficient methods for performing the translations between binary strings and permutation codes, and even (more speculatively) searching for ways of draining charges along a block gradually instead of performing a full erasure[2].

On the coding theory side, there are two main directions. Jiang, Mateescu, Schwartz, and Bruck proposed programming rank modulation codes onto chips via 'push-to-the-top' operations that take the $i^{\text{th}}$ ranked cell and increase its charge to make it the top ranked cell [46]. This has inspired research into complete rank modulation Gray codes, which give sequences of push-to-the-top transitions that traverse the state space $S_n$, as well as cyclic variations, where the final permutation can reach the initial one via a

[2]This would potentially allow continual updates of rank modulation coded memory without any full erasure steps.

push-to-the-top [48], and balanced variations that ensure that all cells have their charges increased a similar number of times. Similarly, for any code $C \subseteq S_n$, we can ask for Gray codes that traverse the elements of the code via minimal sequences of push-to-the-top operations [90, 43, 42].

We will focus on the other direction, studying the error correcting capabilities of the codes themselves. We will see that there are several different distance measures available that model different types of errors that can arise in permutation coding. For each, we will discuss basic bounds on $A(\mathcal{C}, d)$, and develop tools for studying permutation codes in these settings.

As permutation codes under these metrics become better understood, we expect that more applications will be found. The primary driver of the renewed interest in permutation codes in the last decade has been the possibility of rank modulation coding for flash memory, but already other directions have emerged. Recently, rank modulation coding has even been proposed for DNA data storage [52, 66].

## 4.2 Metrics

As we begin to formalize the mathematical coding theory we should use for this rank modulation setting, we need to establish the space, size, and distance that we are working with. The space and size are clear. The space is the set $S_n$ of all permutations on $n$ elements, and since this space is discrete, the natural notion of size is the number of codewords; i.e. the size of a code $C \subseteq S_n$ is $\mathfrak{s}(C) = |C|$. We will allow the set of acceptable codes to just be all subsets of $S_n$. One could enforce some structure by requiring a code to be a subgroup, but the results that we will highlight here do not

require this restriction. That being said, searching through the incredibly large sets $\mathcal{P}(S_n)$ for good codes is computationally prohibitive, so in many cases, we will use subgroup codes as a more directed search method that allows us to produce codes for larger $n$.

For the distance function, there is a variety of options available [20, 19]. We need to ask ourselves which distance measures accurately model the types of errors that are likely to arise in this setting. Every distance function we will cover here arises as the minimum distance under some metric on $S_n$, so the permutation coding theories we'll discuss are all examples of standard coding theories. In the following sections we will cover results on the three permutation metrics that are most useful for the rank modulation setting: the Hamming metric, the Kendall tau metric, and the Ulam metric. From this point on, we will treat permutations formally as elements of the symmetric group $S_n$ using both the vector notation and the cycle notation. More information on these notations can be found in Appendix A.

### 4.2.1 Hamming Metric

Perhaps the most straightforward way to proceed is to simply transfer over what we already know about the Hamming metric to permutation codes. $S_n$ can be viewed as a particular subset of $\Sigma^n$, where $\Sigma = \{1, \ldots, n\}$, so it inherits the Hamming metric on $\Sigma^n$. Most early work on permutation codes took this perspective, which we formalize in the following definition.

**Definition 4.1.** *The coding theory of* permutation codes under the Hamming metric *is a standard coding theory in which $S = S_n$ for some $n \in \mathbb{N}$, $\mathfrak{s}(C) = |C|$, and $\mathfrak{d}(C) =$*

$\min_{\substack{a,b\in C \\ a\neq b}} d_H(a,b)$, *where* $d_H$ *is the Hamming distance*

$$d_H(a,b) = |\{i \in \{1,\ldots,n\} | a(i) \neq b(i)\}|.$$

*In this setting, we'll denote* $A(\mathcal{C},d)$ *by* $A_H(n,d)$.

With the added algebraic structure of the permutation space $S_n$, we can equivalently define the Hamming distance as $d_H(a,b) = n - |\text{Fix}(ba^{-1})|$, where $\text{Fix}(\sigma)$ gives the number of fixed points of the permutation $\sigma$. This form of the definition is useful for establishing that the Hamming metric is right-invariant under the action of the symmetric group, since

$$d_H(ac,bc) = n - |\text{Fix}(bc(ac)^{-1})| = n - |\text{Fix}(ba^{-1})| = d_H(a,b).$$

This tells us that the number of points in a ball of a given radius in this space does not depend on the center of the ball, so there should be a Hamming bound in this space. In fact, all three of the metrics we'll discuss in this section are right-invariant, so all three have Hamming bounds. This definition also makes it easy to see that the lowest Hamming distance between two distinct permutations is 2, since there is no permutation that moves only one element. The highest Hamming distance between two permutations is $n$, just as it was for block codes of length $n$.

### Bounds on $A_H(n,d)$

It's important to note that even though the set of permutations can be seen as a structured subset of $\Sigma^n$ for $\Sigma = \{1,\ldots,n\}$ and we are using the Hamming metric, permutation codes should not be seen as a block code under our definition, since the ambient space is different. In this setting, the only words that can be represented are those that are

actually permutations. As a result, the size of the space $S$ and the size of a Hamming ball of radius $r$ are both smaller than they were for block codes, giving us a different Hamming bound for permutation codes under the Hamming metric.

**Theorem 4.2** (Hamming Bound)**.** *For all $n, d \in \mathbb{Z}$, with $n \geq d \geq 1$:*

$$A_H(n,d) \leq \frac{n!}{B_H\left(\lfloor \frac{d-1}{2} \rfloor\right)},$$

*where $B_H(r)$ denotes the size of a ball of radius $r$ under the Hamming metric. Furthermore, $B_H(r)$ is given by*

$$B_H(r) = \sum_{k=0}^{r} \binom{n}{k} D_k,$$

*where $D_k$ is the number of derangements of order $k$.*

Just as we can obtain a Hamming bound by packing Hamming spheres in the permutation space, we can also search for a Singleton bound by trying to project permutation codes into a smaller space by deleting symbols. This gives the following bound discovered by Frankl and Deza in the 1970's [27].

**Theorem 4.3** (Singleton Bound)**.** *For all $n, d \in \mathbb{Z}$, with $n \geq d \geq 1$:*

$$A_H(n,d) \leq \frac{n!}{(d-1)!}.$$

As with the tradition Hamming and Singleton bounds for block codes, there are several trivial families of codes that meet these bounds and establish certain $A_H(n,d)$ values. The following codes are perfect or Singleton optimal permutation codes in the Hamming metric. At values where the bounds agree, they are both.

Although permutation codes under the Hamming metric were the focus of most early research into permutation codes and there is a rich combinatorial structure to

| Perfect / Singleton optimal code | $A_H(n,d)$ |
|---|---|
| $\{[1 \ldots n]\}$ | $A_H(n,\infty) = 1$ |
| Cyclic subgroup of order $n$ | $A_H(n,n) = n$ |
| Alternating group | $A_H(n,3) = \frac{n!}{2}$ |
| Symmetric group | $A_H(n,2) = n!$ |

Table 3: $A_H(n,d)$ values for the theory of permutation codes under the Hamming metric.

study, this measure of distance is actually not particularly useful for the practical rank modulation setting. Measuring the distance between permutations as the number of positions where they disagree does not reflect the types of errors that arise when the permutation represents the charge rankings in a collection of cells on a flash memory chip. For example, the permutations [12345] and [21345] have the same Hamming distance as [12345] and [52341], but it's much more likely for an error to transpose the top two ranked cells rather than dropping the charge on the first cell to the bottom of the ranking and simultaneously pushing the charge of the last cell to the top. In reality, most errors arise when the charges in the cells drift and leak causing some cells to slip past others that are nearby in the rankings. The Kendall tau and Ulam metrics are much more accurate models for this type of error pattern, so we'll spend the rest of this chapter focusing on these newer coding theories.

### 4.2.2 Kendall tau Metric

The Kendall tau metric determines the distance between two permutations by counting the number of adjacent transpositions required to reach one permutation from the other. Since small changes in the charges in a collection of flash memory cells can only change the ranking of the cells via a sequence of adjacent transpositions, this metric is a good

model for the physical situation we're in. Here's a more precise definition for this coding theory.

**Definition 4.4.** *The coding theory of* permutation codes under the Kendall tau metric *is a standard coding theory in which* $S = S_n$ *for some* $n \in \mathbb{N}$, $\mathfrak{s}(C) = |C|$, *and* $\mathfrak{d}(C) = \min_{\substack{a,b\in C \\ a\neq b}} d_K(a,b)$, *where* $d_K$ *is the Kendall tau distance*

$$d_K(a,b) = |\{(i,j) \in \{1,\ldots,n\}^2 | 1 \leq i,j \leq n,\ a(i) < a(j),\ b(i) > b(j)\}|.$$

*In this setting, we'll denote* $A(\mathcal{C},d)$ *by* $A_K(n,d)$.

Just as with the Hamming metric, there are several ways to think of what the Kendall tau metric measures. We've already described it as measuring the number of adjacent transpositions required to reach one permutation from another. For our formal definition, we chose to use a different (but equivalent) interpretation: the Kendall tau distance between permutations $a$ and $b$ is equal to the number of pairs of symbols that are ordered differently between $a$ and $b$. Computer scientists typically think of this as the 'bubble-sort' distance, since it's the same as the number of transposition steps that the bubble-sort algorithm would use to bring the permutation $a$ into the order in $b$. After defining an inversion of a permutation $\sigma$ to be a pair $(i,j)$ with $i < j$ and $\sigma(j) < \sigma(i)$, this can be rephrased one more time to say that the Kendall tau distance between $a$ and $b$ is the number of inversions in the permutation $ab^{-1}$, which we will denote $\mathrm{Inv}(ab^{-1})$. Since the Kendall tau distance counts inversions, this tells us that the Kendall tau distance between two distinct permutations can vary from 1 to $\binom{n}{2}$. This form of the definition is also useful for showing that the Kendall tau distance is right invariant, since

$$d_K(ac,bc) = \mathrm{Inv}(ac(bc)^{-1}) = \mathrm{Inv}(ab^{-1}) = d_K(a,b).$$

**Bounds on $A_K(n,d)$**

Once again, the right invariance of this metric tells us that the size of a Kendall tau ball does not depend on its center, so permutation codes under the Kendall tau metric satisfy a Hamming bound.

**Theorem 4.5** (Hamming Bound)**.** *For all $n, d \in \mathbb{Z}$, with $\binom{n}{2} \geq d \geq 1$:*

$$A_K(n,d) \leq \frac{n!}{B_K\left(\lfloor \frac{d-1}{2} \rfloor\right)},$$

*where $B_K(r)$ denotes the size of a ball of radius $r$ under the Kendall tau metric. Furthermore, $B_K(r)$ is given by*

$$B_K(r) = \sum_{k=0}^{r} M(n,k),$$

*where $M(n,k)$ are the Mahonian numbers counting the number of permutations in $S_n$ with $k$ inversions, given by the recursion*

$$M(1,0) = 1,$$

$$M(1,k) = 0 \text{ if } k \neq 0,$$

$$M(n,k) = \sum_{j=0}^{n-1} M(n-1, k-j).$$

The observant reader may have noticed that we haven't yet cited any references for this result. Certainly the existence of a sphere-packing bound is already well-known in this setting, and in fact, a form of this bound appears in a paper of Jiang, Schwartz, and Bruck from 2010 [49]. There are also several papers containing bounds for small values of $n$ and $d$ that are computed from this sphere-packing bound (presumably by explicitly computing the size of these small Kendall tau balls) [56, 8, 88]. The Mahonian numbers are similarly well-known, and the fact that they count the number of permutations in

$S_n$ with $k$ inversions was demonstrated by MacMahon as early as 1913 [57]. It seems, however, that these two facts have yet to be connected in the literature on permutation codes. It is our hope that the statement of the Hamming bound we've given above will help to clarify this situation.

As usual, the next step in developing this coding theory is to search for a Singleton bound by projecting codes in $S_n$ under the Kendall tau metric into lower dimensional symmetric groups. This was accomplished by Barg and Mazumdar in 2010 [3]. Their Singleton bound takes a somewhat more complicated looking form than most other Singleton bounds because they choose a projection that does not act injectively on the code in their proof. Instead, they compute an estimate on the distance between two codewords that project to the same point and solve the resulting inequalities to produce the following bound.

**Theorem 4.6** (Singleton Bound)**.** *Let* $\binom{n}{2} \geq d > n-1$*, then*

$$A_K(n,d) \leq \left\lfloor \frac{3}{2} + \sqrt{n(n-1) - 2d + \frac{1}{4}} \right\rfloor !.$$

Unlike the other bounds in this section, Barg and Mazumdar state that the Kendall tau Singleton bound holds when $d > n-1$. It would actually still be true if we simply took $d > 1$, but the bound is trivial until $d > n-1$.

As usual, there are a handful of trivial perfect codes and Singleton optimal codes in this setting, but no other perfect or Singleton optimal permutation codes under the Kendall tau metric are known. We summarize the trivial examples in the following table.

Although the code $\{[1 \dots n], [n \dots 1]\}$ is always Singleton optimal, it is only perfect when the balls of radius $\left\lfloor \frac{\binom{n}{2}-1}{2} \right\rfloor$ around these codewords exactly cover $S_n$. This happens whenever $\binom{n}{2}$ is odd, i.e. when $n \equiv 2$ or $3 \pmod 4$. Buzaglo and Etzion [8] have shown

| Perfect / Singleton optimal code | $A_K(n,d)$ |
|---|---|
| $\{[1\dots n]\}$ | $A_K(n,\infty)=1$ |
| $\{[1\dots n],[n\dots 1]\}$ | $A_K(n,\binom{n}{2})=2$ |
| Symmetric group | $A_K(n,1)=n!$ |

Table 4: $A_K(n,d)$ values for the theory of permutation codes under the Kendall tau metric.

that there are no perfect single-error-correcting (minimum distance 3) permutation codes under the Kendall tau metric when $n > 4$ is prime or $4 \leq n \leq 10$. Not much else is known about whether other perfect or Singleton optimal codes exist in this setting.

### Distance and Weight Enumerators

To continue digging into the Kendall tau metric, we'll now turn to studying distance and weight enumerators. In fact, in this section we'll focus on subgroup codes, so we'll only be looking at weight enumerators. For other permutation codes, the weight enumerator would not store the full information on the pairwise distances between the codewords, so the distance enumerator would be used in those cases. We'll denote the Kendall tau weight enumerator polynomial of the code $C$ as $W_C^K(x,y)$.

Because of the connections to the Mahonian numbers and counting inversions in permutations, the weight enumerator for the full symmetric group code has already been studied in other contexts. The Mahonian numbers have been investigated at least since 1871 [7], and the generating function has appeared in various contexts since the 1970's[3] [78, 64]. Here we will state this generating function as the weight enumerator polynomial for the code $C = S_n$ in the Kendall tau metric.

[3]In fact, this generating function appeared in a paper of Olinde Rodrigues in 1839 [71], but appears to have been mostly lost for the following century.

**Theorem 4.7.**

$$W_{S_n}^K(x,y) = \prod_{i=1}^{n} \frac{y^i - x^i}{y - x}.$$

Since the alternating group $A_n$ can be defined as the set of all permutations in $S_n$ with an even number of inversions, we can use this result to find the Kendall tau weight enumerator of the alternating group.

**Theorem 4.8.**

$$W_{A_n}^K(x,y) = \frac{W_{S_n}^K(x,y) + W_{S_n}^K(-x,y)}{2} = \left( \prod_{\substack{i=1 \\ i \text{ even}}}^{n} \frac{y^i - x^i}{y^2 - x^2} \right) P_n(x,y),$$

*for some polynomial* $P_n(x,y)$.

*Proof.* The left equality comes from the observation that the alternating group contains all of the permutations in the symmetric group with even Kendall tau weight. For the second equality, it is useful to also name the coset of the alternating group as a code $B_n$. Then because the coset of $A_n$ contains all of the permutations with odd Kendall tau weight, we can write

$$W_{B_n}^K(x,y) = \frac{W_{S_n}^K(x,y) - W_{S_n}^K(-x,y)}{2}.$$

Now we will study the interactions between these weight enumerators by examining recursions that generate them. First, observe that

$$W_{S_{n+1}}^K(x,y) = W_{S_n}^K(x,y) \frac{y^{n+1} - x^{n+1}}{y - x} = W_{S_n}^K(x,y) \sum_{i=0}^{n} x^i y^{n-i}.$$

To simplify the notation slightly, we'll denote $f_n(x,y) = \frac{y^{n+1} - x^{n+1}}{y-x}$. If we set $W_{S_0}^K = 1$, then this recursion generates all of the $W_{S_n}^K$ weight enumerators.

This gives us the following:

$$\begin{aligned}
2W^K_{A_{n+1}}(x,y) &= W^K_{S_{n+1}}(x,y) + W^K_{S_{n+1}}(-x,y) \\
&= W^K_{S_n}(x,y)f_n(x,y) + W^K_{S_n}(-x,y)f_n(-x,y) \\
&= 2W^K_{A_n}(x,y)\sum_{\substack{i=0 \\ i \text{ even}}}^{n} x^i y^{n-i} + 2W^K_{B_n}(x,y)\sum_{\substack{i=0 \\ i \text{ odd}}}^{n} x^i y^{n-i} \qquad (4.1) \\
&= 2W^K_{A_n}(x,y)f_{\lfloor \frac{n}{2} \rfloor}(x^2,y^2)y^{\chi_{n \text{ odd}}} + 2W^K_{B_n}(x,y)xf_{\lceil \frac{n}{2} \rceil - 1}(x^2,y^2)y^{\chi_{n \text{ even}}} \qquad (4.2)
\end{aligned}$$

The expression in Equation 4.1 follows from collecting together the terms of the sums with even degree in $x$ that appear with the same sign in $f_n(x,y)$ and $f_n(-x,y)$ and the terms of the sums with odd degree in $x$ that appear with opposite signs in $f_n(x,y)$ and $f_n(-x,y)$. Then Equation 4.2 follows from refactoring the terms of the sums in terms of powers of $x^2$ and $y^2$. Here $\chi_{n \text{ odd}}$ and $\chi_{n \text{ even}}$ refer to the indicator functions

$$\chi_{n \text{ odd}} = \begin{cases} 1 & n \text{ odd} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \chi_{n \text{ even}} = \begin{cases} 1 & n \text{ even} \\ 0 & \text{otherwise} \end{cases}$$

A similar argument shows that

$$2W^K_{B_{n+1}}(x,y) = 2W^K_{B_n}(x,y)f_{\lfloor \frac{n}{2} \rfloor}(x^2,y^2)y^{\chi_{n \text{ odd}}} + 2W^K_{A_n}(x,y)xf_{\lceil \frac{n}{2} \rceil - 1}(x^2,y^2)y^{\chi_{n \text{ even}}} \qquad (4.3)$$

Taking Equation 4.2 and Equation 4.3 along with $W^K_{A_2}(x,y) = y$ and $W^K_{B_2}(x,y) = x$ gives us a (somewhat messy) system of recurrences that can be used to generate all of the Kendall tau weight enumerators $W^K_{A_n}(x,y)$ and $W^K_{B_n}(x,y)$. The key observation for the second equality in our theorem is that when $\lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil - 1$, i.e. when $n$ is odd, both $W^K_{A_n}(x,y)$ and $W^K_{B_n}(x,y)$ gain a factor of $f_{\lfloor \frac{n}{2} \rfloor}(x^2,y^2)$. This gives us the factorization

$$W^K_{A_n}(x,y) = \left( \prod_{\substack{i=1 \\ i \text{ even}}}^{n} \frac{y^i - x^i}{y^2 - x^2} \right) P_n(x,y).$$

The exact value of the polynomial $P_n(x, y)$ can be determined using the recurrences given. □

Weight enumerators like these are useful for examining families of codes in higher dimensions than we could reasonably search by just randomly selecting permutations to put into a code, but some care has to be taken. We'll illustrate this with the example of the transitive cyclic groups $C_n = \langle (1 \cdots n) \rangle \subseteq S_n$. This is another family of codes for which we can very easily compute a Kendall tau weight enumerator, and in fact this has already been done [53].

**Theorem 4.9.**

$$W_{C_n}^K(x, y) = \sum_{j=1}^{n} x^{(i-1)(n-i+1)} y^{\binom{n}{2}-(i-1)(n-i+1)}.$$

This result, however, doesn't tell the whole story. There are, of course, many other transitive cyclic subgroups isomorphic to $C_n$ that can be reached via the conjugation action of $S_n$ on these cyclic subgroups of order $n$. Since the Kendall tau distance is only right invariant and not left invariant, there's no reason to expect that these conjugate subgroup codes have the same weight enumerator as the standard embedding of $C_n$. In fact, not only do they often have different weight enumerators, they can even have different minimum distances. In Table 5 we've listed the number of distinct weight enumerators for conjugates of the cyclic group in $S_n$. As $n$ grows larger, there can be very large numbers of conjugates to search for good codes.

| $n$ | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|
| Number of distinct weight enumerators | 1 | 3 | 3 | 25 | 15 | 208 | 167 | 6365 |

Table 5: The number of distinct Kendall tau weight enumerators for conjugates of $C_n$ in $S_n$.

This behavior does not arise for the codes $S_n$ and $A_n$, since they are both normal subgroups of $S_n$ and are thus preserved under conjugation, but the vast majority of codes are not normal subgroups. So if we really want to understand the minimum distances of families of codes in $S_n$, we need to understand how this conjugation action changes the distances between codewords. We'll refer to this conjugation action as *relabeling*. In section 4.3, we will explore how relabeling interacts with the metrics used for rank modulation coding.

### 4.2.3 Ulam Metric

While the Kendall tau metric is a good measure for the errors that arise from low magnitude charge leaks in a rank modulation setting, it is not as accurate in situations where individual cells might undergo large charge leakages. For this type of error model, we use the Ulam metric. Instead of counting the number of adjacent transpositions it takes to reach one permutation from another, the Ulam metric counts the number of *translocations* required to reach one permutation from another. A translocation is defined as a permutation in one of the following two forms: a right translocation

$$\tau = [1, \ldots, i-1, i+1, i+2, \ldots, j, i, j+1, \ldots, n],$$

or a left translocation

$$\tau = [1, \ldots, j-1, i, j, j+1, \ldots, i-1, i+1, \ldots, n].$$

Then the Ulam distance between permutations $a$ and $b$ is the smallest integer $m$ so that there is a sequence of $m$ translocations $\tau_1, \ldots, \tau_m$ such that $b = a\tau_1\tau_2\cdots\tau_m$. Intuitively, this counts the number of times a symbol must be moved from one position in the list

to another (while leaving the remaining symbols in the same order) in order to turn the permutation $a$ into $b$.

While this definition is useful for seeing how the Ulam distance models large charge drift errors on a small number of cells in the flash memory rank modulation setting, it's not the easiest method for computing the Ulam distance. So, just as we've done with both the Hamming and Kendall tau metrics, we'll use an equivalent and more easily computed definition of the metric in our formal statement of this coding theory.

**Definition 4.10.** *The coding theory of* permutation codes under the Ulam metric *is a standard coding theory in which* $S = S_n$ *for some* $n \in \mathbb{N}$, $\mathfrak{s}(C) = |C|$, *and* $\mathfrak{d}(C) = \min_{\substack{a,b \in C \\ a \neq b}} d_U(a,b)$, *where* $d_U$ *is the Ulam distance*

$$d_U(a,b) = n - |\mathrm{Inc}(ba^{-1})|,$$

*where* $\mathrm{Inc}(\sigma)$ *denotes the length of the longest increasing subsequence in* $\sigma$. *In this setting, we'll denote* $A(\mathcal{C}, d)$ *by* $A_U(n,d)$.

This form of the definition of $d_U(a,b)$ is once again useful for showing that this metric is right invariant, following exactly the same argument that we've used previously. So once again we know that the size of a ball of radius $r$ does not depend on its center, and there should be a Hamming bound in this coding theory. In practice, it's usually not necessary to actually compute $ba^{-1}$ to find the Ulam distance between $a$ and $b$. Instead, Ulam distance can be interpreted as $n$ minus the length of the longest common subsequence between $a$ and $b$, which can often be determined at a glance for small $n$.

### Bounds on $A_U(n,d)$

The usual argument gives the the Hamming bound in Theorem 4.11.

**Theorem 4.11** (Hamming Bound)**.** *For all $n, d \in \mathbb{Z}$, with $n \geq d \geq 1$:*

$$A_U(n,d) \leq \frac{n!}{B_U\left(\lfloor \frac{d-1}{2} \rfloor\right)},$$

*where $B_U(r)$ denotes the size of a ball of radius $r$ under the Ulam metric.*

The size of the Ulam balls, $B_U(r)$ in this theorem correspond to the number of permutations with increasing subsequences of lengths at most $n - r$. These numbers were determined by Schensted in 1961 using Young tableaus [74], and this strategy was recently applied by Kong and Hagiwara to the permutation code setting [54]. Kong and Hagiwara were then able to use this result to show the nonexistence of nontrivial perfect permutation codes in the Ulam metric. The trivially perfect permutation codes in the Ulam metric are exactly the same ones that we saw in the Hamming and Kendall tau metrics, the full symmetric group code $S_n$ and the code consisting of just a single codeword $\{[1 \ldots n]\}$.

The Singleton bound for permutation codes in the Ulam metric, Theorem 4.12, was found by Farnoud, Skachek, and Milenkovic [26] by following the same argument that Barg and Mazumdar had used for the Kendall tau metric just 3 years earlier. Interestingly, the Ulam version of the Singleton bound takes a significantly simpler looking form. This is due to the fact that maximum distances in the Ulam metric are linear (the Ulam weight of the reverse order permutation is $n - 1$), while the maximum distances in the Kendall tau metric are quadratic (the Kendall tau weight of the reverse order permutation is $\binom{n}{2}$).

**Theorem 4.12** (Singleton Bound)**.** *For all $n, d \in \mathbb{Z}$, with $n \geq d \geq 1$:*

$$A_U(n,d) \leq (n - d + 1)!.$$

In Table 6, we present known values of $A_U(n,d)$ in a different format than we used for previous coding theories. Instead of only listing the known perfect and Singleton optimal codes, we've given information on all pairs $(n,d)$ with $4 \leq n \leq 9$ and $1 \leq d \leq n-1$ in order to highlight the current state of knowledge and also call attention to an important open question. In cases where inequalities are given, the lower bound comes from a code found via computer searches except in two cases that we will address at the end of this chapter. These values are not necessarily expected to be close to the true value. All the upper bounds given in the table come from the Singleton bound. In some cases, these Singleton bound values have been improved. For example, the $d = n-4$ diagonal contains several strict inequalities on the upper bounds due to computations of Göloğlu, Lember, Riet, and Skachek [31], which determined that there are no Singleton optimal codes in these cases. The values $A_U(6,3) = 24$ and $A_U(7,4) = 12$ were similarly found via exhaustive computer searches by Mathon and Trung in the context of directed Steiner systems [63].

| $n$ \ $d$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 4 | **4!** | **3!** | **2** | | | | | |
| 5 | **5!** | **4!** | 4 | **2** | | | | |
| 6 | **6!** | **5!** | **24** | 4 | **2** | | | |
| 7 | **7!** | **6!** | $\geq 59$ $\leq 120$ | 12 | 4 | **2** | | |
| 8 | **8!** | **7!** | $\geq 336$ $\leq 720$ | $\geq 48$ $\leq 120$ | $\geq 8$ $\leq 12$ | 4 | **2** | |
| 9 | **9!** | **8!** | $\geq 1512$ $\leq 5040$ | $\geq 144$ $\leq 720$ | $\geq 18$ $\leq 120$ | $\geq 9$ $\leq 12$ | 2 | **2** |

Table 6: $A_U(n,d)$ values for the theory of permutation codes in the Ulam metric. Bold numbers indicate codes that meet the Singleton bound.

The bold values in Table 6 are the known Singleton optimal codes. The column $A_U(n,1) = n!$ is the full symmetric group code $S_n$, and the diagonal $A_U(n,n-1) = 2$

is the code $\{[1 \dots n], [n \dots 1]\}$. These are the trivial Singleton optimal codes. The column $A_U(n, 2) = (n-1)!$ is due to a clever construction of Levenshtein[4] [55]. In fact, Levenshtein's construction shows how to partition $S_n$ into $n$ disjoint Singleton optimal codes by grouping permutations based on their 'up-down' sequences, which are binary strings representing which adjacent pairs of symbols appear in increasing order and which appear in decreasing order.

The last Singleton optimal code in the table establishes that $A_U(6, 3) = 24$. As we've mentioned, this code was discovered through a computer search by Mathon and Trung, who also observed that this set of permutations forms a subgroup of $S_6$ isomorphic to $S_4$. Based on this observation, we propose a constructive method for finding this code.

To begin with, we observe that a permutation code, $C$, in the Ulam metric with $n = 6$, $\mathfrak{s}(C) = 24$, and $\mathfrak{d}(C) = 3$ could potentially arise from a group action of $S_4$ on a set of 6 elements. Fortunately, $S_4$ contains 6 transpositions along with a natural conjugation action on this set, giving us the sought after homomorphism $S_4 \hookrightarrow S_6$. Just as we saw with the Kendall tau metric earlier, however, labeling matters when it comes to determining weight enumerators and minimum distances. It turns out that there are exactly four possible labelings of the 6 transpositions in $S_4$ that can produce a Singleton optimal code via this construction. For example, if we label the transpositions as

$$\begin{array}{cccccc} (14) & (12) & (23) & (24) & (13) & (34) \\ 1 & 2 & 3 & 4 & 5 & 6 \end{array},$$

then this construction produces the Singleton optimal code $C = \langle (1236)(45), (14)(35) \rangle$. Two labelings of the transpositions produce this Singleton optimal code. Two of the

[4]Levenshtein mentions that this method of partitioning $S_n$ using up-down sequences was already known [51, 86], but he was the first one to apply it to error-correcting codes.

remaining labelings produce the only other Singleton optimal code in the Ulam metric with $n = 6$ and $d = 3$, $\langle (1456)(23), (13)(25) \rangle$.

Since $S_m$ always has $\binom{m}{2}$ transpositions, this construction can certainly be generalized to produce a family of codes with $n = \binom{m}{2}$ and $\mathfrak{s}(C) = n!$. The distance, $\mathfrak{d}(C)$, however depends greatly on the labeling of the transpositions. When $m = 5$, the best possible labeling of the transpositions in this construction gives a code with $n = 10$, $\mathfrak{s}(C) = 5!$, and $\mathfrak{d}(C) = 5$, where a Singleton optimal code of this size would need $\mathfrak{d}(C) = 6$. When $m = 6$, the best possible labeling of the transpositions in this construction gives a code with $n = 15$, $\mathfrak{s}(C) = 6!$, and $\mathfrak{d}(C) = 7$, where a Singleton optimal code of this size would need $\mathfrak{d}(C) = 9$. So these codes are not Singleton optimal, but this construction does give a quick way to find good codes with a nice subgroup structure for far larger values of $n$ than can be reasonably searched with the computer-aided methods used to produce the values in Table 6. We will discuss the connections between labeling and minimum distance of the Ulam metric more in depth in section 4.3. As of now, it remains an open question whether there are any other Singleton optimal permutation codes in the Ulam metric other than the ones we've mentioned in this section.

### Distance and Weight Enumerators

The weight enumerators for $S_n$ under the Ulam metric are less symmetric and somewhat harder to work with than the ones we saw for the Kendall tau metric earlier. Here are the first few:

$$W^U_{S_2}(x,y) = x + y$$

$$W^U_{S_3}(x,y) = x^2 + 4xy + y^2$$

$$W^U_{S_4}(x,y) = x^3 + 13x^2y + 9xy^2 + y^3$$

$$W^U_{S_5}(x,y) = x^4 + 41x^3y + 61x^2y^2 + 16xy^3 + y^4$$

$$W^U_{S_6}(x,y) = x^5 + 131x^4y + 381x^3y^2 + 181x^2y^3 + 25xy^4 + y^5$$

$$W^U_{S_7}(x,y) = x^6 + 428x^5y + 2332x^4y^2 + 1821x^3y^3 + 421x^2y^4 + 36xy^5 + y^6$$

Algorithms for computing these counts using Young tableaus have been known since at least the 1960's [2], but there is still no closed form formula known. However, if we restrict ourselves to particular sequences of coefficients, it turns out much more can be said.

Let $p(n,m)$ be the number or permutations in $S_n$ with longest increasing subsequence of length $m$. In other words, $p(n,m)$ is the coefficient of the $x^{n-m}y^{m-1}$ term in $W^U_{S_n}(x,y)$. It is clear that $p(n,n) = 1$ and $p(n,1) = 1$, since there is only one identity permutation and one reverse-order permutation. The reader may have also observed that $p(n,n-1) = (n-1)^2$. This is also true in general and follows from a short counting argument. Every permutation of Ulam weight $n-2$ can be produced from the reverse-order permutation of Ulam weight $n-1$ via a single translocation. There are $n$ values we can translocate, each of which can be moved into $n-1$ new positions. This would give a count of $n(n-1)$, but $n-1$ of these are the adjacent transpositions which were counted twice. Thus there are $n(n-1) - (n-1) = (n-1)^2$ permutations in $S_n$ of Ulam weight $n-2$, i.e. $p(n,n-1) = (n-1)^2$.

In fact, it was shown by the 1970's that $p(n,n-2)$ and $p(n,2)$ are also given by

polynomials with

$$p(n, n-2) = \frac{1}{2}n(n-1)(n-2)(n-3) + 1$$

and

$$p(n, 2) = \frac{(2n)!}{n!(n+1)!} - 1.$$

Note that this means $p(n, 2)$ is always one fewer than a Catalan number, giving even more motivation for the importance of these sequences. Surprisingly, even though $p(n, n-2)$ seems to have a very clean formula, according to Hammersley [39], it is easier to prove this formula by writing this function in the form

$$p(n, n-2) = \binom{n-1}{2} + \left(\binom{n-1}{2} - 1\right)\left(2\binom{n-1}{2} - 1\right).$$

This is similar to what we saw in the counting argument for $p(n, n-1)$ above, since it turned out that $(n-1)^2$ should be written as $n(n-1)-(n-1)$ for the sake of the proof.

In Hammersley's paper on this topic, he also provides a seventh degree polynomial computed by a Mr. Izenman for $p(n, n-3)$. This polynomial turns out to be incorrect, since it was shown by Rogers that the function $p(n, n-3)$ must be a polynomial of degree 6 [72]. In Figure 4, we've computed the polynomial $p(n, n-3)$, which is quickly proved to be correct as a result of Rogers' result and the fact that it matches the data when $n \geq 5$. We've also computed formulas for the next several functions $p(n, n-4)$, $p(n, n-5)$, and $p(n, n-6)$, which we conjecture to be correct for large enough $n$ (the exact necessary size of $n$ is listed below). For each successive formula, we've found that more initial terms must be dropped from the sequence before we eventually find a degree $2m$ polynomial that interpolates $p(n, n-m)$. It's possible that the number of terms dropped gives some hint as to what kind of counting argument is necessary to prove these formulas, but for now they remain conjectural. Although many of these

polynomials can be written in several more simplified forms, we've left them in their expanded form because it is unclear in which form they should be written in order to elucidate a strategy for proving them.

$$
\begin{aligned}
p(n,n) &= 1 && (n \geq 1)\\
p(n,n-1) &= n^2 - 2n + 1 && (n \geq 2)\\
p(n,n-2) &= \frac{1}{2}n^4 - 3n^3 + \frac{11}{2}n^2 - 3n + 1 && (n \geq 3)\\
p(n,n-3) &= \frac{1}{6}n^6 - 2n^5 + \frac{26}{3}n^4 - \frac{49}{3}n^3 + \frac{79}{6}n^2 - \frac{11}{3}n + 1 && (n \geq 5)\\
p(n,n-4) &= \frac{1}{24}n^8 - \frac{5}{6}n^7 + \frac{79}{12}n^6 - \frac{157}{6}n^5 + \frac{443}{8}n^4 - \frac{181}{3}n^3 && (n \geq 7)\\
&\quad + \frac{59}{2}n^2 - \frac{25}{6}n + 1\\
p(n,n-5) &= \frac{1}{120}n^{10} - \frac{1}{4}n^9 + \frac{25}{8}n^8 - \frac{127}{6}n^7 + \frac{10163}{120}n^6 && (n \geq 9)\\
&\quad - \frac{12271}{60}n^5 + \frac{6989}{24}n^4 - \frac{451}{2}n^3 + \frac{2309}{30}n^2 - \frac{137}{30}n + 1\\
p(n,n-6) &= \frac{1}{720}n^{12} - \frac{7}{120}n^{11} + \frac{761}{720}n^{10} - \frac{781}{72}n^9 + \frac{16711}{240}n^8 && (n \geq 11)\\
&\quad - \frac{11651}{40}n^7 + \frac{192449}{240}n^6 - \frac{171667}{120}n^5 + \frac{568663}{360}n^4\\
&\quad - \frac{35131}{36}n^3 + \frac{23519}{90}n^2 - \frac{49}{10}n + 1
\end{aligned}
$$

Figure 4: Polynomials $p(n, n-m)$ giving the number of permutations in $S_n$ with longest increasing subsequence of length $n - m$.

In the previous section, we referenced a formula for the Kendall tau weight enumerators of $C_n$, but we saw that, in fact, different conjugates of $C_n$ in $S_n$ can have very different weight distributions, since $C_n$ is not a normal subgroup. The same is true of $C_n$ under the Ulam metric, so in this section we'll only give the counts of how many distinct Ulam weight enumerator polynomials can arise from the different conjugates of $C_n$. This data is in Table 7.

| $n$ | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|
| Number of distinct weight enumerators | 1 | 3 | 2 | 11 | 12 | 49 | 48 | 267 |

Table 7: The number of distinct Ulam weight enumerators for conjugates of $C_n$ in $S_n$.

Once again, we can see that conjugation can drastically change the weights of permutations. We'll now turn to exploring this in more detail in section 4.3.

## 4.3 Relabeling

We've seen in the previous sections of this chapter that relabeling can drastically change the weight distributions of permutations codes. In this section we will explore the relabeling problem in greater depth and establish some bounds on how much relabeling can affect the weight of an individual codeword and how much it can affect the minimum distance of a code. To be precise, when we refer to relabeling a permutation or a code, we mean conjugating by some element of $S_n$.

**Definition 4.13.** *The conjugate of a permutation $\sigma \in S_n$ by a permutation $\tau \in S_n$ given by $\tau\sigma\tau^{-1}$ is called the* relabeling *of $\sigma$ by $\tau$.*

*Similarly, the* relabeling *of a permutation code $C$ by $\tau$ is the set $\tau C\tau^{-1}$.*

This is called relabeling because conjugation in the symmetric group has the effect of relabeling the terms in the cycle decomposition of the permutation being conjugated. That is, if

$$\sigma = (a_1, \ldots, a_{k_1})(b_1, \ldots, b_{k_2}) \cdots ,$$

then the cycle decomposition of the relabeling of $\sigma$ by $\tau$ is

$$\tau\sigma\tau^{-1} = (\tau(a_1), \ldots, \tau(a_{k_1}))(\tau(b_1), \ldots, \tau(b_{k_2})) \cdots .$$

Note that this operation preserves the cycle structures of the permutations being relabeled, so in particular the number of 1-cycles or fixed points remains the same. This means that relabeling will have no effect on the Hamming weight of a permutation. In this section, we will study the effect of relabeling on the Kendall tau and Ulam metrics. This work was inspired by the work of Tamo and Schwartz studying the effect of relabeling on the infinity metric, which is used in rank modulation settings where limited magnitude errors are expected [82].

Throughout this section, we will express the cycle structure of a permutation $\sigma$ as a list $(k_1, k_2, \ldots, k_r)$, where $k_1, k_2, \ldots, k_r$ are the lengths of the nontrivial[5] disjoint cycles in the cycle decomposition of $\sigma$. The cycle structure is sufficient to determine the maximum and minimum Kendall tau or Ulam weights of the the relabelings of a permutation. This follows from the fact that any permutation $\sigma$ can be relabeled to any other permutation with the same cycle structure, including those of maximum or minimum weight.

### 4.3.1 Relabeling in the Kendall tau Metric

For an individual permutation our main objects of study are the minimum and maximum Kendall tau weights that can be reached by relabeling.

**Definition 4.14.** *$K^n_{\min}(k_1, k_2, \ldots, k_r)$ is the minimum Kendall tau weight that can be reached by a permutation in $S_n$ with cycle type $(k_1, k_2, \ldots, k_r)$.*

*$K^n_{\max}(k_1, k_2, \ldots, k_r)$ is the maximum Kendall tau weight that can be reached by a permutation in $S_n$ with cycle type $(k_1, k_2, \ldots, k_r)$.*

Recall that the Kendall tau weight of a permutation $\sigma$ can be interpreted as the

[5]We refer to the 1-cycles as trivial cycles.

minimum number of adjacent transpositions required to generate $\sigma$ or as the number of inversions in $\sigma$. We will use both of these interpretations in this section.

The relationship between inversions and cycle structure has already been touched upon by Edelman [24], but he was interested in the number of cycles in a permutation rather than the full cycle structure, leading to significantly different results. Conjugation and relabeling do not appear in Edelman's paper at all. Instead, Edelman defines $l(\sigma)$ to be the number of inversions in $\sigma$ and $c(\sigma)$ to be the number of cycles in a permutation $\sigma$. His primary goal is understanding the bivariate generating function $F(x,y) = \sum_{\sigma \in S_n} x^{l(\sigma)} y^{c(\sigma)}$. After defining

$$b(n,k) = \min\{l(\sigma) | \sigma \in S_n, c(\sigma) = k\}$$

and

$$B(n,k) = \max\{l(\sigma) | \sigma \in S_n, c(\sigma) = k\},$$

his main theorem states that for $1 \leq k \leq n-1$,

$$b(n,k) = n - k$$

and

$$B(n,k) = \begin{cases} \binom{n}{2} - \left\lceil \frac{n}{2} \right\rceil + k, & k \leq \left\lceil \frac{n}{2} \right\rceil \\ \binom{n}{2} - \binom{2k-n}{2}, & k > \left\lceil \frac{n}{2} \right\rceil . \end{cases}$$

Since the only permutation with $n$ cycles is the identity permutation with $n$ 1-cycles and all the permutations with only one cycle are cyclic permutations with cycle type $(n)$, these extremal cases of Edelman's theorem are also special cases of our Theorem 4.15 and Theorem 4.16. Our results give more refined information on the individual cycle types rather than just counting the number of cycles, so our theorems can also be used to give alternate proofs of Edelman's theorems.

Our main results in this section are formulas for the minimum and maximum Kendall tau weights of a permutation given its cycle structure. We'll begin with the more straightforward formula for the minimum Kendall tau weights.

**Theorem 4.15.** *For all cycle structures $(k_1, k_2, \ldots, k_r)$, where $\sum k_i \leq n$,*

$$K^n_{\min}(k_1, k_2, \ldots, k_r) = \sum_{i=1}^{r} (k_i - 1).$$

*Proof.* First, suppose $g$ consists of a single $k$-cycle. As at least $k-1$ adjacent transpositions are required to generate a $k$-cycle, the Kendall tau weight of $g$ must be at least $k-1$, i.e. there are at least $k-1$ inversions amongst the $k$ elements permuted by $g$. Now, let $g$ be a permutation with cycle type $(k_1, k_2, \ldots, k_r)$. Since these cycles are disjoint, it follows that $g$ has at least $\sum_{i=1}^{r}(k_i - 1)$ inversions, and thus has Kendall tau weight at least $\sum_{i=1}^{r}(k_i - 1)$. This bound is attained, for example, by the permutation $(1, \ldots, k_1)(k_1 + 1, \ldots, k_1 + k_2) \cdots (k_1 + \cdots + k_{r-1} + 1, \ldots, k_r)$. □

Our formula for the maximum Kendall tau weight of a permutation given its cycle structure is significantly more complicated. It is easiest to state it when the cycle structure $(k_1, k_2, \ldots, k_r)$ is already given in a particular order, with all of the even cycle types in the leftmost terms and all of the odd cycle types listed to the right in decreasing order.

**Theorem 4.16.** *For all cycle structures $(k_1, k_2, \ldots, k_r)$, where $\sum_{i=1}^{r} k_i \leq n$ and the $k_i$'s are ordered such that, for some integer $m$ with $0 \leq m \leq r$,*

- *$k_1, k_2, \ldots, k_m$ are even,*
- *$k_{m+1}, k_{m+2}, \ldots, k_r$ are odd,*

- *and* $k_{m+1} \geq k_{m+2} \geq \cdots \geq k_r$,

*we have*

$$K^n_{\max}(k_1, k_2, \ldots, k_r) = \sum_{i=1}^{r} W^{k_i}_{n-\sum_{j=1}^{i-1} k_j},$$

*where* $W^k_n = 2\left\lfloor \frac{k}{2} \right\rfloor (n-k) + \left\lceil \frac{(k-1)^2}{2} \right\rceil$. *Note that* $W^k_n$ *gives the maximum Kendall tau weight of a permutation in* $S_n$ *with a single* $k$*-cycle and* $n-k$ *fixed points.*

*Proof.* First, suppose that $\sigma \in S_n$ consists of a single $k$-cycle and $(n-k)$ 1-cycles. We claim that the weight of $\sigma$ is at most $W^k_n$. Note that every inversion of $\sigma$ must be of one of two types: inversions between an element of the cycle and a fixed point, or inversions between two elements of the cycle.

We'll start with the first type. For each of the $(n-k)$ fixed points in $\sigma$, there can be at most $2\left\lfloor \frac{k}{2} \right\rfloor$ inversions. This follows from the fact that in order for a point $p$ to remain fixed by the permutation $\sigma$, the number of elements that move from the left of $p$ to the right of $p$ must be equal to the number of elements that move from the right of $p$ to the left of $p$. Since only $k$ elements are being moved by the cycle in $\sigma$, at most $\left\lfloor \frac{k}{2} \right\rfloor$ can move in each direction past $p$, for a total of $2\left\lfloor \frac{k}{2} \right\rfloor$ inversions involving $p$. This accounts for the maximum of $2\left\lfloor \frac{k}{2} \right\rfloor (n-k)$ inversions of the first type.

To count the number of inversions of the second type, we must compute $W^k_k$, the maximum number of inversions in a $k$-cycle on $k$ elements. We claim that this number is $\left\lceil \frac{(k-1)^2}{2} \right\rceil$. First observe that the maximum number of inversions in a permutation on $k$ symbols is $\binom{k}{2}$, which only appears in the reverse order permutation with $\left\lceil \frac{k}{2} \right\rceil$ cycles in its cycle decomposition. In order to connect these $\left\lceil \frac{k}{2} \right\rceil$ cycles into a single cycle by composing with adjacent transpositions, it would take at least $\left\lceil \frac{k}{2} \right\rceil - 1$ adjacent transpositions, each of which reduces the number of inversions by 1, since we've started

from a state where every pair is an inversion. Thus the total number of inversions in a $k$-cycle on $k$ elements is at most $\binom{k}{2} - \lceil \frac{k}{2} \rceil + 1 = \left\lceil \frac{(k-1)^2}{2} \right\rceil$. This number is obtained by a $k$-cycle with vector form

$$\left[ \left\lfloor \frac{k}{2} \right\rfloor + 1, k, k-1, \ldots, \left\lfloor \frac{k}{2} \right\rfloor + 2, \left\lfloor \frac{k}{2} \right\rfloor, \ldots, 2, 1 \right]. \tag{4.4}$$

Alternatively, the fact that $W_k^k = \left\lceil \frac{(k-1)^2}{2} \right\rceil$ follows immediately from the special case of Edelman's theorem where a permutation in $S_k$ has only one cycle, since $B(k,1) = \binom{k}{2} - \lceil \frac{k}{2} \rceil + 1$.

To produce a permutation with a $k$ cycle and $(n-k)$ fixed points with this maximum Kendall tau weight $W_n^k = 2 \left\lfloor \frac{k}{2} \right\rfloor (n-k) + \left\lceil \frac{(k-1)^2}{2} \right\rceil$, we can simply insert the fixed points within the permutation in Equation 4.4 at the position we've marked as $*$ in Equation 4.5 and adjust the indexing.

$$\left[ \left\lfloor \frac{k}{2} \right\rfloor + 1, k, k-1, \ldots, \left\lfloor \frac{k}{2} \right\rfloor + 2, *, \left\lfloor \frac{k}{2} \right\rfloor, \ldots, 2, 1 \right]. \tag{4.5}$$

Now, given a more general cycle type $(k_1, k_2, \ldots, k_r)$ with the $k_i$'s in the prescribed order, we can produce a permutation of this cycle type of maximal Kendall tau weight by nesting them in a similar fashion. In other words, begin with the $k_1$-cycle of the form in Equation 4.5, and then insert the $k_2$-cycle, again in the arrangement given in Equation 4.5, into the position marked $*$ in the $k_1$-cycle and adjust the indexing. Repeat this procedure by placing the $k_3$ cycle, in the form given in Equation 4.5, into the $*$ position of the $k_2$-cycle adjusting the indexing as we go, and continue until all nontrivial cycles have been nested together, finally inserting the remaining fixed points

into the $*$ position of the innermost $k_r$-cycle. This arrangement of cycles has Kendall tau weight $\sum_{i=1}^{r} W^{k_i}_{n-\sum_{j=1}^{i-1} k_j}$, and this is maximized since every cycle has the maximum number of inversions with the other members of the same cycle, the elements in the even-length cycles are inverted relative to every element in every other cycle, and placing the longer odd-length cycles around the shorter odd-length cycles maximizes the number of inversions between the elements of each odd-length cycle and the members of the other odd-length cycles, since $k_1, k_2$ odd and $k_1 > k_2$ implies that

$$2\left\lfloor \frac{k_1}{2} \right\rfloor (n-k_1) + 2\left\lfloor \frac{k_2}{2} \right\rfloor (n-k_1-k_2) > 2\left\lfloor \frac{k_2}{2} \right\rfloor (n-k_2) + 2\left\lfloor \frac{k_1}{2} \right\rfloor (n-k_1-k_2).$$

□

These $W_n^k$ numbers are likely of somewhat general interest, since they measure the maximum number of inversions possible in a permutation on $n$ symbols with a single $k$-cycle and $n-k$ fixed points. Below, we give a table of the first few rows of the $W_n^k$ triangle. Note that the diagonal of the table contains the numbers $W_k^k = \left\lceil \frac{(k-1)^2}{2} \right\rceil$, which measure the maximum number of inversions in a $k$-cycle on $k$ elements. Each column is an arithmetic sequence with common difference $2\left\lfloor \frac{k}{2} \right\rfloor$.

With these results on the effect of relabeling on the weights of individual codewords, we can now look to apply these theorems to larger codes and subgroup codes in particular. Our main result here is that the minimum minimum[6] Kendall tau distance among the relabelings of a permutation code $C$ is given by the minimum weight attainable by codewords with cycle structures that appear in $C$.

[6]There are two minimums here because for each code we are computing the minimum distance between codewords, and we are minimizing this value over the relabelings of the code.

| $n$ \ $k$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 1 | | | | | | | | |
| 3 | 3 | 2 | | | | | | | |
| 4 | 5 | 4 | 5 | | | | | | |
| 5 | 7 | 6 | 9 | 8 | | | | | |
| 6 | 9 | 8 | 13 | 12 | 13 | | | | |
| 7 | 11 | 10 | 17 | 16 | 19 | 18 | | | |
| 8 | 13 | 12 | 21 | 20 | 25 | 24 | 25 | | |
| 9 | 15 | 14 | 25 | 24 | 31 | 30 | 33 | 32 | |
| 10 | 17 | 16 | 29 | 28 | 37 | 36 | 41 | 40 | 41 |

Table 8: The triangle of $W_n^k$ numbers giving the maximum number of inversions in a permutation on $n$ symbols with a single $k$-cycle and $n-k$ fixed points.

**Theorem 4.17.** *Given a subgroup code $C \subseteq S_n$ in the Kendall tau metric, let $K$ denote the set of all cycle structures that appear in the non-identity permutations of $C$. Then*

$$\min_{\tau \in S_n} \mathfrak{d}(\tau C \tau^{-1}) = \min_{(k_1,k_2,\ldots,k_r)\in K} K^n_{\min}(k_1, k_2, \ldots, k_r).$$

*Proof.* Since $C$ is a subgroup and the Kendall tau distance is right invariant, the minimum Kendall tau distance between codewords in $C$ is equal to the minimum weight of the non-identity codewords in $C$. The minimum possible weight amongst permutations that are conjugates to non-identity codewords in $C$, $\min_{(k_1,k_2,\ldots,k_r)\in K} K^n_{\min}(k_1, k_2, \ldots, k_r)$, appears in some relabeling of $C$, $\tau C \tau^{-1}$, for some $\tau \in S_n$, so the claimed result follows. □

This theorem can now be applied to various permutation groups to determine their minimum Kendall tau minimum distance under relabeling. As an example, we'll apply it to the cyclic groups $C_n$.

**Corollary 4.18.** *In the Kendall tau metric,*

$$\min_{\tau \in S_n} \mathfrak{d}(\tau C_n \tau^{-1}) = n - \frac{n}{p},$$

*where $p$ is the smallest prime factor of $n$.*

*Proof.* By Theorem 4.17, we should begin by determining the cycle types that appear in $C_n$. $C_n$ is generated by a permutation $\sigma$ that contains a single $n$-cycle and has order $n$. For any $k$ dividing $n$, $\sigma^k$ has order $\frac{n}{k}$ and has cycle decomposition $\left(\underbrace{\frac{n}{k}, \cdots, \frac{n}{k}}_{k \text{ times}}\right)$. By Theorem 4.15,

$$K^n_{\min}\left(\underbrace{\frac{n}{k}, \cdots, \frac{n}{k}}_{k \text{ times}}\right) = \sum_{i=1}^{k}\left(\frac{n}{k} - 1\right) = n - k.$$

The smallest non-zero value occurs when $k$ is the largest nontrivial factor of $n$, i.e. $k = \frac{n}{p}$ where $p$ is the smallest prime factor of $n$. This gives the minimum Kendall tau minimum distance of relabelings of $C_n$ as $n - \frac{n}{p}$. □

Of course, the more important question in the search for good permutation codes in the Kendall tau metric is "What is the *maximum* Kendall tau minimum distance of the relabelings of a permutation code $C$?" This is a much harder question and remains open in general. For the equivalent question in the infinity metric, Tamo and Schwartz were able to show that this question is hard to approximate and even showed that determining whether a code can be relabeled to distance 2 or more in the infinity metric is NP-complete [82].

### 4.3.2 Relabeling in the Ulam Metric

We'll now turn to answering these same questions in the Ulam metric, once again starting off by defining the minimum and maximum Ulam weights that can be obtained by relabeling a permutation with a given cycle structure.

**Definition 4.19.** *$U^n_{\min}(k_1, k_2, \ldots, k_r)$ is the minimum Ulam weight that can be reached by a permutation in $S_n$ with cycle type $(k_1, k_2, \ldots, k_r)$.*

*$U^n_{\max}(k_1, k_2, \ldots, k_r)$ is the maximum Ulam weight that can be reached by a permutation in $S_n$ with cycle type $(k_1, k_2, \ldots, k_r)$.*

Since the adjacent transpositions generating the Kendall tau metric are a small proper subset of the translocations generating the Ulam metric, we should expect both $U^n_{\min}(k_1, k_2, \ldots, k_r)$ and $U^n_{\max}(k_1, k_2, \ldots, k_r)$ to be significantly smaller than the corresponding values for the Kendall tau metric. It turns out that they are also more straightforward to compute when $n$ is sufficiently large relative to the cycle type. We will see that $U^n_{\min}(k_1, k_2, \ldots, k_r)$ does not even depend on the lengths of the cycles and $U^n_{\max}(k_1, k_2, \ldots, k_r)$ stabilizes to the sum of the cycle lengths when $n$ is large enough.

**Theorem 4.20.** *For all cycle structures $(k_1, k_2, \ldots, k_r)$, where $\sum k_i \leq n$,*

$$U^n_{\min}(k_1, k_2, \ldots, k_r) = r.$$

*Proof.* First, suppose $\sigma \in S_n$ consists of a single $k$-cycle. Since $\sigma$ is not the identity, at least 1 translocation is required to generate $\sigma$. Now, let $\sigma$ be a permutation with cycle type $(k_1, k_2, \ldots, k_r)$. Since these $r$ cycles are disjoint, at least one translocation is necessary to generate each of the nontrivial cycles in $\sigma$. This implies that $U^n_{min}(k_1, k_2, \ldots, k_r) \geq r$. This bound is attained, for example, by the permutation $(1, \ldots, k_1)(k_1 + 1, \ldots, k_1 + k_2) \cdots (k_1 + \cdots + k_{r-1} + 1, \ldots, k_r)$. □

For the maximum Ulam weight of a permutation with a given cycle type $(k_1, k_2, \ldots, k_r)$, we will restrict ourselves to large values of $n$, where $2r + \sum k_i \leq n$. When $n$ is at least $2r + \sum k_i$, the maximum Ulam weight is $\sum_{i=1}^{r} k_i$, but in cases where $n$ is only slightly

larger than $\sum k_i$, the maximum Ulam weight may be smaller. We'll discuss this in a bit more detail after our proof of Theorem 4.21.

**Theorem 4.21.** *For all cycle structures $(k_1, k_2, \ldots, k_r)$, where $2r + \sum k_i \leq n$,*

$$U^n_{\max}(k_1, k_2, \ldots, k_r) = \sum_{i=1}^{r} k_i.$$

*Proof.* Let $\sigma \in S_n$ be a permutation with cycle type $(k_1, k_2, \ldots, k_r)$. Observe that for any $k$-cycle in $\sigma$, all $k$ elements can be returned to their sorted positions with at most $k$ translocations. Summing this over all the cycles in $\sigma$, we see that the Ulam weight of $\sigma$ must be less than or equal to $\sum_{i=1}^{r} k_i$.

Now, note that $\sigma$ has Ulam weight $\sum_{i=1}^{r} k_i$ if and only if the fixed points of $\sigma$ form a longest increasing subsequence. We will consider the cases $r = 1$ and $r > 1$ separately.

If $r = 1$ and $k_1 = n - 2$, then we must show that there is a permutation in $S_n$ with a single $(n-2)$-cycle and two fixed points that form a longest increasing subsequence. To construct this, we'll begin with the cycle in Equation 4.4. We will then insert two fixed points at 2 and $\lceil \frac{n}{2} \rceil + 1$, increasing the value of every element of the original cycle greater than or equal to 2 by 1 and then increasing the value of every element greater than or equal to $\lceil \frac{n}{2} \rceil + 1$ by 1. If $n$ is odd, this gives us the permutation in Equation 4.6, and if $n$ is even, this gives us the permutation in Equation 4.7.

$$\left[\left\lceil \frac{n}{2} \right\rceil, 2, n, n-1, \ldots, \left\lceil \frac{n}{2} \right\rceil + 1, \left\lceil \frac{n}{2} \right\rceil - 1, \ldots, 4, 3, 1\right] \tag{4.6}$$

$$\left[\left\lceil \frac{n}{2} \right\rceil + 2, 2, n, n-1, \ldots, \left\lceil \frac{n}{2} \right\rceil + 3, \left\lceil \frac{n}{2} \right\rceil + 1, \ldots, 4, 3, 1\right] \tag{4.7}$$

Since the first element of each permutation has a value between the two fixed points, every element between the fixed points in the permutation is larger than both and in decreasing order, and every element to the right of the fixed points is smaller than both

fixed points and in decreasing order, we have that the longest increasing subsequence is of length 2 and is given by the two fixed points. Thus if $r = 1$ and $k_1 = n - 2$, the maximum Ulam weight is $k_1$.

If $r = 1$ and $k_1 < n - 2$, then we can use the same construction to form a $k_1$ cycle in $S_{k_1+2}$ and then append further fixed points to the right hand side to lift this permutation into $S_n$. Since the sequence of fixed points is still a longest increasing subsequence, we still have that the maximum Ulam weight is $k_1$.

Finally, suppose that $r > 1$ and $2r + \sum k_i \leq n$. We will begin with each $k_i$-cycle in $S_{k_i+2}$ in the form given in Equation 4.6 or Equation 4.7, depending on whether $k_i$ is odd or even. Now, within each of these length $k_i + 2$ permutations, the longest increasing subsequence is of length two and is given by the fixed points. So if we append these together and adjust the indexing (i.e. increase the values in the $k_j^{\text{th}}$ permutation by $\sum_{i=1}^{j-1} k_i$) we'll get a permutation on $2r + \sum k_i$ symbols with the longest increasing subsequence given by the fixed points. Since $2r + \sum k_i \leq n$, we can then lift this permutation into $S_n$ by appending fixed points to the right-hand side to produce a permutation in $S_n$ with cycle type $(k_1, \ldots, k_r)$ and Ulam weight $\sum_{i=1}^{r} k_i$. □

In fact, there is likely a stronger statement that can be made here. For small $n$, the maximum Ulam weight of a permutation with cycle type $(k_1, \ldots, k_r)$ is almost always either $\sum k_i$ if $\sum k_i \leq n-2$ or $n-2$ if $\sum k_i > n-2$ unless the permutation has the cycle type of the reverse-order permutation in which case the maximum Ulam weight is $n-1$. There are only a few known counterexamples to this, typically involving at least three 3-cycles. For example, the cycle type $(3, 3, 3)$ in $S_9$ only has maximum Ulam weight 6, the cycle type $(3, 3, 3, 3)$ in $S_{12}$ only has maximum Ulam weight 9, and the cycle type

$(3,3,3,3,3)$ in $S_{15}$ only has maximum Ulam weight 12. Each of these examples also has one less than the 'expected' maximum Ulam weight in the next few larger symmetric groups until it eventually stabilizes at the value $\sum k_i$ that we proved above. Random sampling suggests similar behavior for other cycle types in larger symmetric groups. The exact general formula for the maximum Ulam weight of a permutation of a given cycle type remains an open question.

Just as with the Kendall tau metric, we can now take these results on the weights of individual codewords under relabeling and apply them to subgroup codes in $S_n$.

**Theorem 4.22.** *Given a subgroup code $C \subseteq S_n$ in the Ulam metric, let $K$ denote the set of all cycle structures that appear in the permutations in $C$. Then*

$$\min_{\tau \in S_n} \mathfrak{d}(\tau C \tau^{-1}) = \min_{(k_1,k_2,\ldots,k_r)\in K} U^n_{\min}(k_1, k_2, \ldots, k_r).$$

*Proof.* This follows from the same argument used for Theorem 4.17. □

Once again, we'll demonstrate the use of this theorem by applying it to the transitive cyclic groups $C_n$, but the result will be significantly more straightforward this time.

**Corollary 4.23.** *In the Ulam metric,*

$$\min_{\tau \in S_n} \mathfrak{d}(\tau C_n \tau^{-1}) = 1.$$

*Proof.* By Theorem 4.22, we should begin by determining the cycle types that appear in $C_n$. $C_n$ is generated by a permutation $\sigma$ that contains a single $n$-cycle. By Theorem 4.20, any permutation with a single nontrivial cycle can be relabeled to one with Ulam weight 1, so we have that the minimum Ulam minimum distance of a relabeling[7] of $C_n$ is 1. □

[7] In fact, the relabeling that achieves this is $C_n$ itself, so it wasn't really necessary to use Theorem 4.22.

The more important question of "What is the *maximum* Ulam minimum distance of the relabelings of a permutation code $C$?" remains hard in this setting. Since increasing the weight of an individual codeword via relabeling will often push other codewords closer together, it's unclear how to determine which relabelings lead to larger minimum distances across the whole code. Even for a fairly basic permutation group like $C_n$ this becomes a complicated question. To illustrate this, Table 9 gives the number of conjugates of $C_n$ with minimum distance $d$. Note that the two transitive cyclic group codes in $S_9$ with minimum distance 6, $\langle(143957286)\rangle$ and $\langle(176942835)\rangle$, are the reason we knew that $A_U(9,6) \geq 9$ in Table 6, and the forty transitive cyclic groups codes in $S_8$ with minimum distance 5 are the reason we knew that $A_U(8,5) \geq 8$. To understand how good transitive cyclic groups can be as permutation codes in the Ulam metric, it is very important to know where the rightmost non-zero entry falls for each value of $n$ in Table 9. This remains an open question.

| $n$ \ $d$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 2 | 1 | | | | | | |
| 3 | 1 | | | | | | |
| 4 | 1 | 2 | | | | | |
| 5 | 1 | 5 | | | | | |
| 6 | 1 | 22 | 37 | | | | |
| 7 | 1 | 16 | 77 | 26 | | | |
| 8 | 1 | 33 | 353 | 833 | 40 | | |
| 9 | 1 | 31 | 572 | 3639 | 2475 | 2 | |
| 10 | 1 | 70 | 1650 | 17197 | 57712 | 14090 | |
| 11 | 1 | 50 | 1344 | 19584 | 132866 | 200783 | 8252 |

Table 9: The number of conjugates of $C_n$ with minimum Ulam distance $d$.

# Chapter 5

# Coding Curves

*'No plan survives first contact with the enemy.'*

(Helmuth von Moltke the Elder)

In this chapter, our goal is to develop an entirely new type of coding theory. While one could certainly just pick a mathematical structure at will and attempt to find a reasonable size and distance function for that setting, we take the view that a good coding theory should assist in answering a natural question. Towards that end, we take our inspiration here from recent research involving time-of-flight (ToF) cameras as well as a more intuitive question about packing ropes into boxes. It is our hope that this chapter will serve as an example that will provide the reader with a framework for developing future coding theories.

Typically, the first few steps are as follows:

1. Find a natural setting in which you would like to pack a lot of data in a self-avoiding way.

2. Carefully define your space $S$, the set of acceptable codes $\mathcal{C}$, the size function $\mathfrak{s}$, and the distance function $\mathfrak{d}$ in a way that matches the demands of your setting.

3. Check whether your new coding theory satisfies a Hamming-type or Singleton-type bound.

4. Search for good codes that approach these bounds.

Of course, this is far from the whole story. There are many bounds beyond the basic Hamming and Singleton bounds, which are often unique to the specifics of this new coding theory. Beyond that are a great many secondary coding theory questions to be answered: Is encoding and decoding necessary in your setting, and can it be done efficiently? Are decision functions necessary for correcting errors in this setting, and do the good codes you've found also have good decision functions? What adjustments are necessary to ensure that the abstract coding theory you've developed can be applied effectively to the problem that inspired it? Any sufficiently interesting coding theory will open up a wealth of questions to inspire future research.

## 5.1 Time-of-Flight Cameras

Time-of-flight cameras are a type of range detecting camera, which can determine the distance between the camera and the scene at each point of the image. Within the last couple decades, ToF technology has improved to the point that it now appears in a variety of consumer technologies including collision detection systems in automobiles and motion-sensing input devices like the Microsoft Kinect. These cameras are also becoming important tools for robotics and computer vision applications.

In order to detect the distance to an object, ToF cameras operate by sending their own light signal to the scene from a light source adjacent to the lens. This signal then bounces off of the surface (losing some intensity depending on the reflectivity of the surface) and back to the lens, where it is received along with some additional ambient light. The distance must then be resolved from whatever light is received. Most current

applications use a version of this technique called continuous-wave ToF, in which the intensity of the light sent out is modulated over time according to some *modulation function* $M(t)$ and the sensitivity of the sensor is modulated over time according to some *demodulation function* $D(t)$. In this setup, the data received is the integral of the product of the received light and the demodulation function over some period of time.

This data collected by the sensor depends on three unknowns: the distance to the object, the albedo or reflectivity of the object, and the ambient light intensity. To handle this, continuous-wave ToF systems use at least three measurements to solve for the distance, e.g. if 30ms are allotted for a measurement and three sensor readings are to be used, 10ms will be used for each measurement, potentially using different modulation and/or demodulation functions for each. The details of this procedure can be found in [36].

So far, this does not necessarily look like coding theory. There is certainly data, but it's not yet clear that we are packing it into any sort of space, and it's not clear in what sense it should be self-avoiding. These notions come out of an abstraction due to Gupta and Velten [36]. They view this imaging method as taking a point in the unknown space of distances, albedos, and ambient light intensities, and encoding it as some point in a $K$-dimensional normalized measurement space in the the shape of a $K$-dimensional unit cube, where $K$ is the number of measurements taken. The data collected gives some point in this measurement space, and it must be decoded back into the unknown space to determine the distance to the object in the scene. Gupta and Velten observed that as the distance to a point varies, the image of that point in the measurement space traces out some curve.

Crucially, the shape of this curve is determined by the modulation and demodulation

functions, which gives us the ability to choose which curve works best. Gupta and Velten were able to show that the length of this coding curve helps to determine the depth resolution of the camera. Longer curves are better, but if a curve isn't self-avoiding enough, then a small amount of noise may cause a very large error in the depth measurement. It is at this point that we can begin identifying this ToF camera setting as a coding theory. Essentially, good coding curves for ToF cameras are those that pack long curves into a $K$-dimensional unit box in a self-avoiding way.

Most continuous-wave ToF cameras available today use either sinusoidal or square-wave functions for both modulation and demodulation. The resulting curves in the measurement space are a circle and a polygon, respectively. With these curves, modern ToF cameras are able to reach a depth resolution of ~1cm. Gupta and Velten were able to produce a longer 'Hamiltonian' curve that follows a Hamiltonian path along the edges of the edge graph of the cube (after removing the vertices $(0, 0, 0)$ and $(1, 1, 1)$ for reasons we'll discuss later in this chapter). Their longer curve is able to improve on the depth-resolution by a factor of 10, giving a resolution of ~1mm with exactly the same time and power budget.

In this chapter, we propose a formal coding theory to study coding curves for ToF cameras. To do this, we will abstract away all of the details of the modulation and demodulation factorization, and focus on the coding theory issue of packing long self-avoiding curves into boxes. Of course, once a good coding curve is found, there are separate questions of finding practical pairs of modulation and demodulation functions that produce this curve. We will not cover that problem here, but it is one that has already begun to be explored in the case of the Hamiltonian curve [38].

## 5.2 Definitions and Background

In order to formalize a coding theory for this setting, we'll need to begin by establishing what we mean by a curve, and in what sense it should be self-avoiding.

**Definition 5.1** (Curve)**.** *A curve is the image of a continuous map* $q : [0, 1] \to S$ *for some topological space* $S$*.*

Since the normalized measurement space for ToF cameras takes the shape of a unit box, the topological space $S$ will typically be $[0, 1]^n \subset \mathbb{R}^n$ in this chapter. A code will then be a curve or a collection of curves in this $n$-dimensional unit box.

Our measurement of self-avoidingness will have to be somewhat different from what we've used in standard coding theories earlier in this dissertation. There is a Euclidean distance between points in the cube that we can work from, but it would be meaningless to define the distance function for a curve to be the minimum of the pairwise distances between points on the curve. Since curves are continuous maps, this minimum distance would just be 0 in any case where the curves consist of more than just single points.

Instead, we'll focus on adapting the notion of packing radius to this setting. Recall that in standard coding theories, the minimum distance of a code measures the same properties as the packing radius. That is, if a standard code has minimum distance $d$, then every error of size up to $\lfloor \frac{d-1}{2} \rfloor$ can be uniquely corrected to the nearest codeword. In the case of standard codes, if the error is at most $\lfloor \frac{d-1}{2} \rfloor$, then minimum distance decoding will always give the codeword that was sent. In our new setting, it is too much to ask that minimum distance decoding will always give original transmitted codeword, but we can still ask that minimum distance decoding gives a unique answer up to a certain error radius. Since the curves are continuous, we can also expect that minimum

distance decoding will give a point that is close to the codeword that was sent when the error is less than this radius.

Now, to define precisely what it means for a curve or a collection of curves to be self-avoiding, we need to define the maximum radius for which minimum distance decoding can give a unique answer for each point in that neighborhood. Fortunately, knot theorists have already done this.

**Definition 5.2** (Reach)**.** *The reach of a curve, Reach$(q)$, is the largest $\epsilon$ so that any point in an $\epsilon$-neighborhood of the curve has a unique nearest neighbor on the curve.*

*More generally, the reach of a set $S$, Reach$(S)$ is the largest $\epsilon$ so that any point in an $\epsilon$-neighborhood of $S$ has a unique nearest neighbor in $S$.*

This measurement is also sometimes referred to as the 'normal injectivity radius' [34]. Just as with many of our earlier distance measures, there are alternative definitions of reach available as well. Here is another intuitive way to think about the reach of a curve as a sort of global curvature measure [33]:

**Theorem 5.3.**

$$Reach(q) = \inf_{s,t,\sigma} ppp(s,t,\sigma),$$

*where the infimum is taken over all distinct points on the curve, and the function $ppp(s,t,\sigma)$ is the radius of the circle through the points $q(s)$, $q(t)$, and $q(\sigma)$. The reciprocal of $ppp(s,t,\sigma)$ is called the Menger curvature of the points.*

Reach is certainly not the only notion of self-avoidingness we can use to define a theory of coding curves. We've chosen it because of the fact that it provides the best analogy for packing radius and measures the largest error magnitude that is still uniquely

decodable via minimum distance decoding. That being said, there are good coding curves that have bad reach. In fact, the Hamiltonian coding curve of Gupta and Velten has reach 0, as does any other polygonal path that isn't just a single line segment, since any point along the bisector of an angle at a vertex of the path would not have a unique closest point on the curve. It's possible that 'expected reach' or some other measurement would be a better fit for the ToF camera application, but that remains a target for future research.

When reach is referred to in knot theory settings, it is typically used to define the following notion of ropelength.

**Definition 5.4** (Ropelength)**.** *The ropelength of a curve $q$ is given by*

$$Rop(q) = Len(q)/Reach(q),$$

*where $Len(q)$ denotes the length of $q$.*

Most knot theory and topology work on reach and ropelength focuses on minimizing the ropelength for particular knots and proving that ropelength minimizers exist and are sufficiently nice [12, 32]. For example, a standard question in this area would ask "If we have rope that is 1" thick, what is the minimum length of rope required to tie a trefoil knot?[1]" While some work in this area attempts to approximate the minimum ropelength of various knots, another common direction searches for bounds on the minimum ropelength given other parameters like the crossing number [21, 18, 11].

Our goal is to push ropelength in the opposite direction. More precisely, instead of trying to find the shortest curves of a given reach satisfying certain topological properties,

[1]It turns out that about 16.372" of 1" thick rope is necessary to tie a trefoil [65, 80, 67].

we're looking for the longest curves of a given reach that satisfy certain compactness properties (e.g. they have to fit inside of an $n$-dimensional unit box). Despite the fact that this problem has a very natural interpretation as asking "How much rope of a given thickness can be packed into a box?", this direction has seen very little investigation so far. The 2- and 3-dimensional rope packing problems were proposed as open problems in 2010 [17], but no researchers have yet picked up this thread.

Surprisingly, the versions of this problem that have been addressed only appear in a handful of papers. Some research in the biophysics and materials physics communities has used Monte Carlo simulations to study the shapes that DNA, protein, and other polymers might form to pack long strings into compact spaces [62, 79], but much of the interest in those communities seems to be in the qualitative properties of the longest curves instead of the length itself. There has also been some recent work by Gerlach and von der Mosel on wrapping ropes around a sphere [29], but they seem to have not addressed packing ropes into any other spaces. We hope that the coding theory that we develop in this chapter will help to supplement this area of research and renew some of the focus on these broadly pertinent questions.

## 5.3 The Theory of Coding Curves

Before jumping into defining the full coding theory we're studying in this chapter, we'll first define what we mean by a coding curve.

**Definition 5.5** (Coding Curves with $m$ components)**.** *We'll say that a collection of $m$ disjoint and non-self-intersecting curves $q_i(s) : [0,1] \to [0,1]^n$, $i = 1, \ldots, m$, is an $n$-dimensional coding curve with $m$ components. When we refer to a coding curve with*

*m components, C, we will use C to represent both the collection of curves and the set of points in these curves. The usage should be clear from context.*

We will soon see that the properties and bounds of coding curves can depend on whether the curves form a complete loop or the ends are left loose. We'll distinguish between these two situations by defining closed and open coding curves.

**Definition 5.6** (closed/open coding curves)**.** *A curve is closed if* $q(0) = q(1)$*, and a curve is open if it is not closed.*

*A coding curve with m components is closed if each of its components is closed, and it is open if each of its components is open.*

Note that if a coding curve has more than one component, then it is not necessarily closed or open. It may have some components that are closed curves and others that are open.

With these preliminaries in hand, we can now define the theory of coding curves using a distance and size function that match the setting we described earlier in this chapter.

**Definition 5.7** (The Theory of Coding Curves)**.** *The* theory of coding curves *consists of the triple* $(\mathcal{C}_n, \mathfrak{s}, \mathfrak{d})$*, where*

- $\mathcal{C}_n$ *is the set of all n-dimensional coding curves with m components for any* $m \in \mathbb{N}$*.*
- $\mathfrak{s}(C) = \sum_{q_i \in C} \mathrm{Len}(q_i)$*, where* $\mathrm{Len}(q_i)$ *denotes the length of the component* $q_i$*.*
- $\mathfrak{d}(C) = \mathrm{Reach}(C)$*, where the reach of the set* $C \subset [0,1]^n$ *is the largest* $\epsilon$ *so that any point in an* $\epsilon$*-neighborhood of C has a unique nearest neighbor in the set C.*

*The* theory of (open/closed) coding curves *is defined similarly, but only includes the (open/closed) coding curves in* $\mathcal{C}$. *The* theory of coding curves with $m$ components *only includes coding curves with exactly* $m$ *components in* $\mathcal{C}$.

We will be particularly interested in the case where a coding curve has only one component, so we'll give some additional notation for that case.

**Definition 5.8** ($(n, l, r)$-coding curves)**.** *We'll say that a curve* $q(s)$ *in a unit box in* $\mathbb{R}^n$ *is an* $(n, l, r)$*-coding curve if* $Len(q) = l$ *and* $Reach(q) = r$.

Somewhat counterintuitively, it seems that even though the family of closed coding curves is more restricted than the family of open coding curves, the closed coding curves may actually have longer length bounds in general. For example, the circle coding curve on the left in Figure 5 is longer than the horseshoe shaped coding curve on the right, even though they both have the same reach of $\frac{1}{2}$. Later on we will prove that the circle is longer than every open $(2, l, \frac{1}{2})$-coding curve.

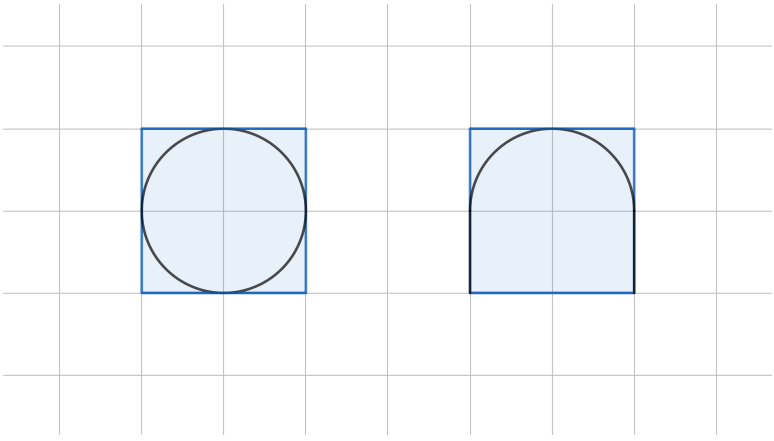

Figure 5: Closed vs. open coding curves. The circle is a closed $(2, \pi, \frac{1}{2})$ coding curve, and the horseshoe is an open $(2, \frac{\pi}{2} + 1, \frac{1}{2})$ coding curve.

It is difficult to optimize curves in general, so to assist in finding examples of good coding curves, we'll define a family of *lattice coding curves* to turn this into a more discrete and combinatorial problem.

**Definition 5.9** (lattice $(n, l, r)$-coding curves)**.** *A lattice $(n, l, r)$-coding curve is an $(n, l, r)$ coding curve comprised of straight edges and quarter-circle tile pieces arranged in a grid. These can be formed from $k^n$ tile pieces, where $k = \frac{1}{2r} + 1$.*

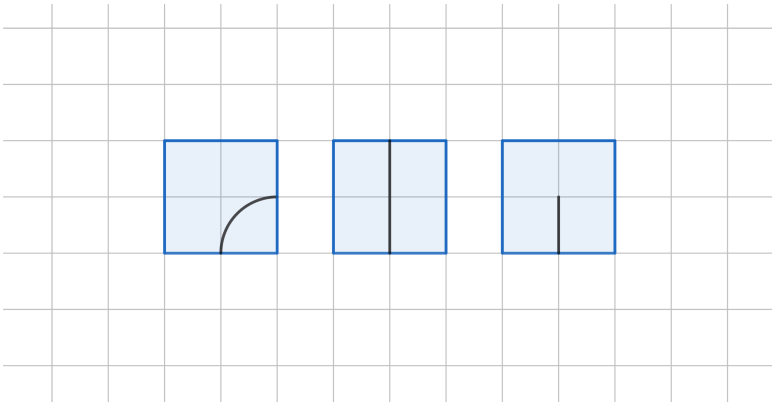

Figure 6: The tile shapes available for building 2-dimensional lattice coding curves.

The tile pieces available for building 2-dimensional lattice coding curves are shown in Figure 6. Closed coding curves will only use the two tiles on the left. Open coding curves will use exactly two copies of the tile on the right.

Since only the curve itself has to lie within the unit box, the tile pieces along the boundary will actually extend out of the box. This restricts which tiles can appear at the boundary. We've illustrated in Figure 7 how the tile pieces could be connected together to form a curve. The red outlines indicate the individual tiles, and the blue region indicates the unit box that the curve must lie within. We'll discuss this particular example in more detail later on.

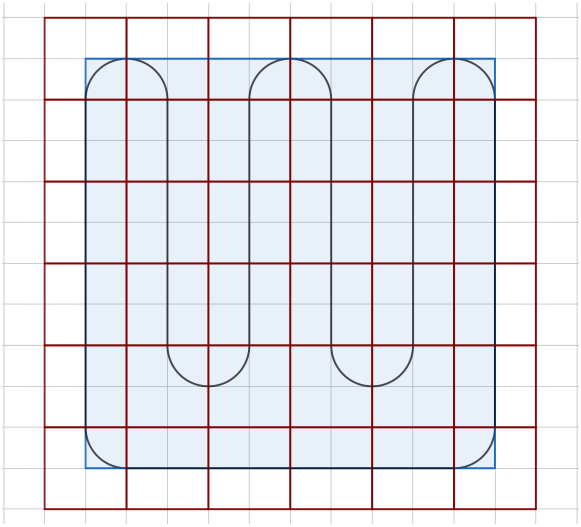

Figure 7: A closed lattice coding curve formed from 36 tiles.

Any length $l$ achieved by a lattice coding curve of reach $r$ will be a lower bound for the maximum length of a (not necessarily lattice) coding curve of reach $r$. However, upper bounds for the lengths of lattice coding curves are not necessarily upper bounds for the lengths of general coding curves. In fact, for high dimensions we should expect the longest coding curves to be non-lattice coding curves, and we will see later that the upper bound for the length of a lattice coding curve is much much smaller than the known upper bounds for general coding curves.

## 5.4 Examples

Before rushing into the fundamental questions and developing bounds, let's familiarize ourselves with a handful of basic examples.

**Example 5.10.**

- *The circle of radius $\frac{1}{2}$ in Figure 5 is a $(2, \pi, \frac{1}{2})$-coding curve.*
- *The horseshoe of radius $\frac{1}{2}$ in Figure 5 is a $(2, \frac{\pi}{2} + 1, \frac{1}{2})$-coding curve.*

- *The saddle curve in Figure 8 is a* $(3, 2\pi, \frac{1}{2})$*-coding curve.*

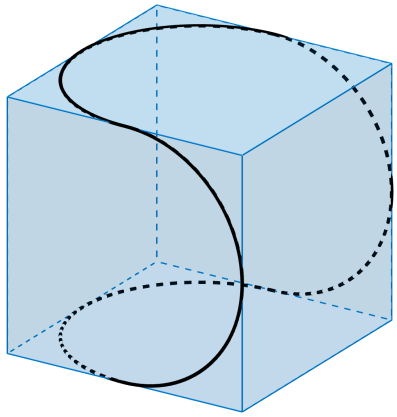

Figure 8: The saddle curve is a $(3, 2\pi, \frac{1}{2})$ coding curve.

- *The curve in Figure 9 is a* $(3, \frac{3\pi}{2}, \frac{1}{2})$*-coding curve. This example can be seen as a rounded off version of Gupta and Velten's Hamiltonian curve.*

In higher dimensions, lattice coding curves are typically not very good. Here's an example to build our intuition as to why that's the case:

**Example 5.11.** *The saddle curve above can be constructed as follows: First, take a Hamiltonian path along the edges of a cube. Then replace each corner with a quarter circle of radius* $\frac{1}{2}$*.*

*If we apply this same procedure to a unit hypercube in* $\mathbb{R}^n$*, we would get a* $(n, 2^{n-2}\pi, \frac{1}{2})$*-coding curve. And since the tile pieces placed into the corners of a lattice coding curve must always be the quarter circle tiles, it is immediately clear that this family gives the longest lattice* $(n, l, \frac{1}{2})$*-coding curves in* $\mathbb{R}^n$*.*

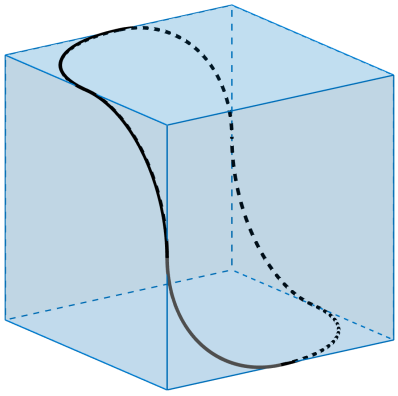

Figure 9: A $(3, \frac{3\pi}{2}, \frac{1}{2})$ coding curve.

*But now consider the distance from the center of the box to its faces. The distance from the center of the cube to a corner is given by* $\frac{\sqrt{n}}{2}$*, the distance from the center of the cube to an edge is given by* $\frac{\sqrt{n-1}}{2}$*, and the distance from the center of the cube to a* 2*-dimensional face is given by* $\frac{\sqrt{n-2}}{2}$*. Since the lattice coding curves we've just constructed lie entirely within the set of* 2*-dimensional faces, the center of the cube is at least* $\frac{\sqrt{n-2}}{2}$ *units from the curve. This means that as long as* $\frac{\sqrt{n-2}}{2} \geq 1$*, there will be room at the center of the box for a second coding curve component. This inequality holds as long as* $n \geq 6$*, so once we reach* $n \geq 6$*, we know that there are coding curves with multiple components that are longer than any lattice coding curve of the same dimension and reach.*

## 5.5 Core Questions

Now we are ready to ask the fundamental questions of the theory of coding curves.

**Definition 5.12** ($A_m(n,r)$ and $A(n,r)$)**.** *The following maximum lengths will be our primary objects of study:*

- $A_m(n,r)$ *is the largest* $l$ *for which there exists an* $n$*-dimensional coding curve with* $m$ *components with length* $l$ *and reach* $r$*.*
- $A_m^o(n,r)$ *is the largest* $l$ *for which there exists an open* $n$*-dimensional coding curve with* $m$ *components with length* $l$ *and reach* $r$*.*
- *We will also define* $A(n,r) = \max_m A_m(n,r)$ *and* $A^o(n,r) = \max_m A_m^o(n,r)$*.*

*Note that* $A_1(n,r)$ *is the largest* $l$ *for which there exists an* $(n,l,r)$*-coding curve, and* $A_1^o(n,r)$ *is the largest* $l$ *for which there exists an open* $(n,l,r)$*-coding curve.*

*If there do not exist any* $n$*-dimensional coding curves with* $m$ *components and reach* $r$*, then we will say that* $A_m(n,r) = -\infty$*, and if there are no* $n$*-dimensional open coding curves with* $m$ *components and reach* $r$*, then we will say that* $A_m^o(n,r) = -\infty$*.*

Using this language, the classic question of the largest number of spheres of radius $r$ that can be packed into an $n$-dimensional unit box (or equivalently the maximum separation problem) is given by $\max\{m \in \mathbb{N} | A_m^o(n,r) \geq 0\}$. So our theory of coding curves can be seen as a generalization of this classic problem.

The sphere packing problem and the maximum separation problem (MSP) are typically distinguished by whether the entire sphere has to fit within the box or if it's

sufficient for just the center of the spheres to be in the box. This is a very minor translation between the two. Our setup is most directly equivalent to the maximum separation problem, where only the center of the spheres must lie within the box.

Our notation is also capable of representing whether or not there are so called 'rattlers' in the sphere packing. If $A^o_m(n,r) = 0$, then there exists a sphere packing with no rattlers, whereas if $A^o_m(n,r) > 0$, then there exists a sphere packing with rattlers. The size of $A^o_m(n,r)$ could be interpreted as giving a measure of how much free motion the rattlers have along paths.

For $n$-dimensional coding curves with length $l$ and reach $r$, our primary questions are as follows:

**Question 5.13.**

- *What bounds govern the relationships between $n$, $l$, and $r$?*
- *Given $n$ and $r$, what is $A(n,r)$? In particular, what is $A_1(n,r)$?*
- *Is it true that $m_1 < m_2 \implies A_{m_1}(n,r) \geq A_{m_2}(n,r)$? And in particular, is $A(n,r) = A_1(n,r)$?*
- *For which $(n,r)$ is $A_1(n,r) < A^o_1(n,r)$? More generally, when is $A_m(n,r) < A^o_m(n,r)$?*

## 5.6 Bounds

We'll explore these questions and build our understanding of the functions $A(n,r)$ and $A^o(n,r)$ by establishing some fundamental bounds, and then later searching for examples that approach these bounds.

### 5.6.1 Lower Bounds

Recall that any length $l$ attained by a lattice coding curve is a lower bound for the maximum length of a coding curve with the same $n$ and $r$ parameters. In other words every $(n, l, r)$-coding curve gives a lower bound on $A_1(n, r)$ and thus a lower bound on $A(n, r)$. We'll study lattice coding curves in greater depth in section 5.8. Here's another lower bound:

**Theorem 5.14** (Diagonal Lower Bound)**.**

$$\sqrt{n} \leq A_1(n, r)$$

*Proof.* This is the observation that the diagonal line segment across the cube is an $(n, \sqrt{n}, r)$-coding curve for any value of $r$ (although this becomes somewhat meaningless for values of $r$ above $\frac{\sqrt{n}}{2}$). □

Although this lower bound is somewhat trivial, it's very useful for seeing how different this coding theory is from others we've seen earlier. No matter what values we choose for $n$ and $r$, we can always find curves of a reasonable length. Of course, for small values of $r$, we can do much better.

### 5.6.2 Hamming Bounds

As usual the first upper bound we should look for is the Hamming-type bound. Since the codes in this coding theory are formed from continuous curves instead of discrete points, the Hamming bound should no longer be thought of as a 'sphere-packing' bound[2], but

[2]In fact, even in earlier coding theories it should've been thought of as a 'ball-packing' bound instead of a 'sphere-packing' bound, but the original phrasing has stuck around.

we can still establish bounds by comparing the volume of the $r$-neighborhood around the curve to the space it must be contained in.

**Theorem 5.15** (Basic Hamming Bound)**.**

$$A_m(n,r) \leq \frac{(1+2r)^n}{V_{n-1}(r)},$$

*where $V_i(r)$ is the volume of an $i$-ball of radius $r$, $V_i(r) = \frac{\pi^{\frac{i}{2}}}{\Gamma(\frac{i}{2}+1)} r^i$. Similarly,*

$$A_m^o(n,r) \leq \frac{-mV_n(r) + (1+2r)^n}{V_{n-1}(r)}.$$

*Proof.* Let $C$ be an $n$-dimensional coding curve with $m$ components of length $l$ and reach $r$. Since every point along the curve must lie within the unit $n$-cube, the $r$-neighborhood of $C$ must lie within an $n$-cube with a side length of $1+2r$. This $n$-cube has a volume of $(1+2r)^n$.

The $r$-neighborhood of $C$ will have a volume of at least $V_{n-1}(r)A_m(n,r)$. This value represents the exact volume if $C$ is a closed coding curve, but if $C$ is not closed, then there will be a pair of hemispherical caps of size $\frac{V_n(r)}{2}$ for each open component. Comparing this volume to the size of the $n$-cube containing it gives us the bound

$$A_m(n,r) \leq \frac{(1+2r)^n}{V_{n-1}(r)}.$$

If $C$ is an open coding curve, then there will be exactly $m$ pairs of hemispherical caps, so the $r$-neighborhood of $C$ would have a total volume of $V_{n-1}A_m(n,r) + mV_n(r)$. Comparing this volume to the volume of the $n$-cube gives us the bound

$$A_m^o(n,r) \leq \frac{-mV_n(r) + (1+2r)^n}{V_{n-1}(r)}.$$

□

For $r = \frac{1}{2}$, this gives bounds of $A(2, \frac{1}{2}) \leq 4$ and $A(3, \frac{1}{2}) \lesssim 10.19$. The best known curves in these cases have $l \approx 3.14$ and $l \approx 6.18$, respectively.

The traditional sphere packing volume bound for unit boxes in $\mathbb{R}^n$ can be recovered from this volume bound by taking the largest $m$ such that the basic volume bound given above for $A_m^o(n, r)$ is greater than 0.

We obtained this basic Hamming bound by comparing the volume of the $r$-neighborhood of a coding curve to the volume of an $n$-cube of side length $1 + 2r$. In fact, we can do a bit better. Many of the points in the $n$-cube of side length $1 + 2r$ are actually not within $r$ units of the unit $n$-cube, so they could never be in the neighborhood of radius $r$ from the coding curve $C$. If we replace this $n$-cube with an $r$-neighborhood of the unit $n$-cube, then we get a better Hamming bound.

**Theorem 5.16** (Refined Hamming Bound)**.**

$$A_m(n, r) \leq \frac{1}{V_{n-1}(r)} \sum_{i=0}^{n} \binom{n}{n-i} V_i(r),$$

*where $V_i(r)$ is the volume of an $i$-ball of radius $r$, $V_i(r) = \frac{\pi^{\frac{i}{2}}}{\Gamma(\frac{i}{2}+1)} r^i$. Similarly,*

$$A_m^o(n, r) \leq \frac{1}{V_{n-1}(r)} \left( -mV_n(r) + \sum_{i=0}^{n} \binom{n}{n-i} V_i(r) \right).$$

*Proof.* Let $C$ be an $n$-dimensional coding curve with $m$ components of length $l$ and reach $r$. The $r$-neighborhood of $C$ must lie entirely within the $r$-neighborhood of the unit $n$-cube containing $C$. We claim that the $r$-neighborhood of the unit $n$-cube has volume $\sum_{i=0}^{n} \binom{n}{n-i} V_i(r)$. This volume follows from the fact that the unit $n$-cube contains $2^i \binom{n}{n-i}$ total $(n-i)$-cube faces, and each of these corresponds to a region of the $r$-neighborhood of the unit $n$-cube. For each of these $(n-i)$-cubes, there are $i$ edge directions that point out of the cube, forming at each point a $\frac{1}{2^i}$ fraction of an $i$-ball in the $r$-neighborhood

of the unit $n$-cube. Integrating the volumes of these regions across the entire $(n-i)$ dimensional face, which is a unit $(n-i)$-cube of volume 1, gives us a total volume of $\frac{1}{2^i}V_i(r)$ corresponding to that $(n-i)$ dimensional face in the $r$-neighborhood of the unit $n$-cube. Adding all of these contributions from each face together gives the total volume of the $r$-neighborhood of the unit $n$-cube as $\sum_{i=0}^{n}\binom{n}{n-i}V_i(r)$.

Once again, the $r$-neighborhood of $C$ will have a volume of at least $V_{n-1}(r)A_m(n,r)$. This value represents the exact volume if $C$ is a closed coding curve, but if $C$ is not closed, then there will be a pair of hemispherical caps of size $\frac{V_n(r)}{2}$ for each open component. Comparing this volume to the size of the $r$-neighborhood of the $n$-cube containing it gives us the bound

$$A_m(n,r) \leq \frac{1}{V_{n-1}(r)}\sum_{i=0}^{n}\binom{n}{n-i}V_i(r).$$

If $C$ is an open coding curve, then there will be exactly $m$ pairs of hemispherical caps, so the $r$-neighborhood of $C$ would have a total volume of $V_{n-1}A_m(n,r)+mV_n(r)$. Comparing this volume to the volume of the $r$-neighborhood of the $n$-cube containing it gives us the bound

$$A_m^o(n,r) \leq \frac{1}{V_{n-1}(r)}\left(-mV_n(r)+\sum_{i=0}^{n}\binom{n}{n-i}V_i(r)\right).$$

□

For $r=\frac{1}{2}$, this gives bounds of $A(2,\frac{1}{2}) \lesssim 3.79$ and $A(3,\frac{1}{2}) \lesssim 8.76$, and open coding curve bounds of $A_1^o(2,\frac{1}{2}) \lesssim 3$ and $A_1^o(3,\frac{1}{2}) \lesssim 8.09$. For $n=2$, this proves that the closed $(2,\pi,\frac{1}{2})$-coding curve given by the circle of radius $\frac{1}{2}$ is longer than any possible open $(2,l,\frac{1}{2})$-coding curve. Later, we will show that this circle is in fact the longest possible $(2,l,\frac{1}{2})$-coding curve, establishing that $A_1(2,\frac{1}{2})=\pi$.

**Theorem 5.17.** *The closed* $(2, \pi, \frac{1}{2})$*-coding curve given by the circle of radius* $\frac{1}{2}$ *is longer than any open* $(2, l, \frac{1}{2})$*-coding curve.*

*Proof.* The length of the circle exceeds the bound for open coding curves given by the previous theorem. □

The refined Hamming bound is only significantly better than the basic Hamming bound for large values of $r$. In fact, the refined bound converges to the basic bound from below as $r \to 0$.

### 5.6.3 A Singleton Bound?

In other coding theory settings, Singleton Bounds arise from projecting a code in $n$ dimensions with minimum distance $d$ into a lower dimensional space, and counting the number of points in the new space. Often the dimension reduction is chosen based on the size of the minimum distance. Projecting into an $n-d+1$ dimensional space is quite common. In our setting, we don't have minimum distance. We've replaced minimum distance with reach, and it doesn't make sense to reduce the dimension of the space by the reach because the reach of a coding curve is typically less than 1. Instead, let's consider what happens when we reduce the dimension of the code by 1.

Suppose we project an $n$ dimensional coding curve into an $n-1$ dimensional space by puncturing just one coordinate. If we also suppose that this projection is injective at all but finitely many points and the length of the projected curve is of the form $2rm$ for some integer $m$, then it seems that the maximum length of the original curve would be $m(\pi r + 1 - 2r)$. This heuristic follows from assuming that the original coding curve that lies above the projection oscillates as quickly as possible under the curvature constraint

between the top and bottom faces of the $n$-cube.

As some evidence that an argument of this sort might actually be useful, we can examine the $(3, 2\pi, \frac{1}{2})$ saddle curve. It has exactly one coordinate that can be punctured to give a projection that is injective at all but finitely many points. The 2-dimensional image of the curve under this projection is a square of side length 1. This curve has length 4, which would give $m = 4$ in the above heuristic. This gives a maximum length of $4(\frac{\pi}{2} + 1 - 2 \cdot \frac{1}{2}) = 2\pi$, exactly the length of the saddle curve itself. In other words, the saddle curve meets the upper bound given by this argument.

## 5.7 Feasible Regions

In this section, we'll examine some consequences of Theorem 5.3. In particular, we will assume that we are give some points or some points and tangent directions on a curve of reach $r$, and we will determine the restrictions that this places on the feasible region that the curve can occupy. We will see that this further reduces the regions that a coding curve of a given reach can occupy in the box and this will allow us to further improve upon our earlier volume bounds.

Our first theorem in this direction will make use of the following definition:

**Definition 5.18** (Lemon)**.** *Given two points $A$ and $B$ with $d(A, B) \leq 2r$, the lemon of radius $r$ between $A$ and $B$ is the intersection of all closed balls of radius $r$ containing $A$ and $B$.*

In two dimensions, there are only two disks to consider, so the lemon of radius $r$ between two points $A$ and $B$ is just the lens of two circles of radius $r$ passing through

$A$ and $B$. In three dimensions, the lemon of radius $r$ between $A$ and $B$ can be given as the region in the interior of a spindle torus.

**Theorem 5.19.** *Let $A$ and $B$ be two points in $\mathbb{R}^n$ with $d(A, B) \leq 2r$. If a third point, $C$, is on a curve of reach $r$ that contains $A$ and $B$, then $C \in \mathcal{F}$, where the feasible region, $\mathcal{F}$, is the union of the lemon of radius $r$ between $A$ and $B$ and the complement of the set of open balls of radius $r$ containing $A$ and $B$ on their boundary.*

Figure 10 illustrates the two dimensional version of this feasible set. As we'll see in the proof, the higher dimensional cases can be reduced to this case as well.

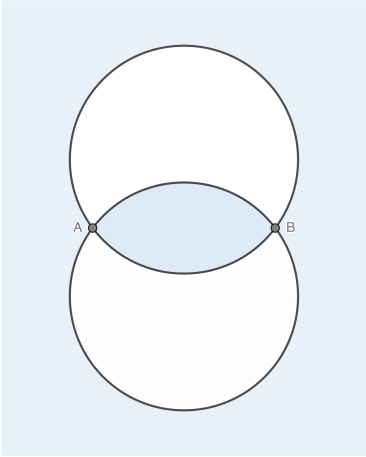


Figure 10: The feasible region given two points on a coding curve.

*Proof.* First, we reduce this question to the two dimensional version by intersecting the space $\mathbb{R}^n$ with the plane containing $A$, $B$, and $C$. This gives us the setup in Figure 10.

Now, we will prove that $C$ must lie in the blue shaded region. We apply Theorem 5.3, which implies that the radius of the circle containing $A$, $B$, and $C$ is at least $r$. We'll refer to Figure 11 for this argument.

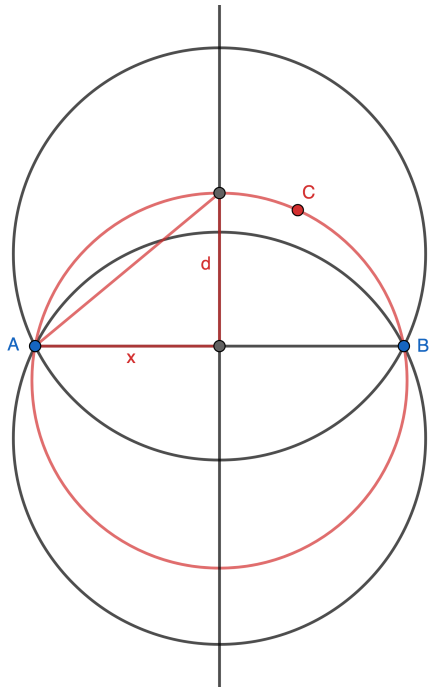


Figure 11: Diagram for proving the 2 point feasible region.

In Figure 11, $x$ represents half the distance from $A$ to $B$. For a given point $C$, we can construct the circle through $A$, $B$, and $C$, and construct the intersection points between this circle and the line through the centers of the circles of radius $r$ containing $A$ and $B$. We will then parametrize these circles by $d \in (0, \infty)$, the distance from the segment $\overline{AB}$ to the the intersection point above the segment. Note that we've skipped the case when $d = 0$, since in that case $A$, $B$, and $C$ are collinear and certainly $C$ is in the feasible region. The radius of the red circle through $A$, $B$, and $C$ is then given by

$$R = \frac{|AB||AC||BC|}{4|\Delta ABC|} = \frac{(2x)(\sqrt{x^2+d^2})(\sqrt{x^2+d^2})}{4xd} = \frac{x^2+d^2}{2d}.$$

The feasible region is then contained in the set of points where $r \leq R$, which is given as follows:

$$r \leq \frac{x^2+d^2}{2d} \implies 0 \leq x^2 - 2rd + d^2 \implies d \notin \left(r - \sqrt{r^2-x^2}, r + \sqrt{r^2-x^2}\right).$$

Since these two boundary points are exactly where the black circles through $A$ and $B$ with radius $r$ intersect the vertical line in the diagram, we can see that the infeasible region consists of exactly those points that are on the interior of exactly one of the two circles in the diagram.

To complete the proof, it now suffices to show that every point in this blue shaded region is on some curve of reach $r$ containing $A$ and $B$. But this is immediate, since for any point $C$ in the blue region the circle through $A$, $B$, and $C$ is a curve of reach at least $r$.

Therefore the feasible region $\mathcal{F}$ consists of the union of the lemon of radius $r$ between $A$ and $B$ and the complement of the set of open balls of radius $r$ containing $A$ and $B$ on their boundary. □

Our theorem determines the feasible region for an individual point $C$ on the curve connecting $A$ and $B$. In fact, any curve of reach $r$ containing these two points must lie entirely within this blue feasible region. As we continue to add points, the feasible region becomes more and more restricted.

If we now take the point $B$ arbitrarily close to $A$, then we also obtain a result on the feasible region for a curve that passes through $A$ with a particular tangent direction at $A$. This is illustrated in Figure 12.

**Theorem 5.20.** *Let $A$ be a point on a curve of reach $r$ that has tangent vector $\vec{v}$ at $A$. If a second point $C$ lies in the curve, then $C \in \mathcal{F}$, where the feasible region $\mathcal{F}$ is the complement of the union of all balls of radius $r$ that are tangent to the vector $\vec{v}$ at $A$.*

This now gives us enough machinery to prove that the following theorem:

**Theorem 5.21.** $A_1(2, \frac{1}{2}) = \pi$.

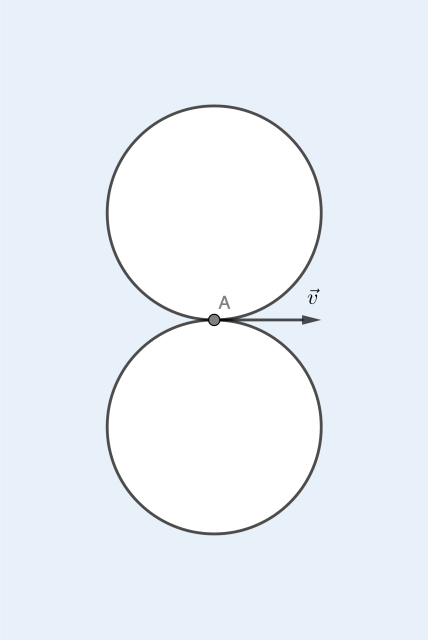


Figure 12: The feasible region given a point and a tangent direction along a coding curve.

*Proof.* We have already shown that the circle of radius $\frac{1}{2}$ is longer than any open $(2, l, \frac{1}{2})$-coding curve, so it suffices to show that this circle is the longest closed $(2, l, \frac{1}{2})$-coding curve. In fact, we will show that it is the only closed $(2, l, \frac{1}{2})$-coding curve.

Let $C$ be a closed $(2, l, \frac{1}{2})$-coding curve, and let $A$ be a point on $C$ with tangent direction $\vec{v}$. By the previous theorem, we know that the curve $C$ must pass through the feasible region defined in that theorem for the point $A$ and tangent direction $\vec{v}$. If $A$ is not on the circle of radius $\frac{1}{2}$ inscribed in the unit box, then there is no way to arrange the feasible region so that there is a closed loop passing through $A$ within the unit box. And similarly if $A$ is on the unit circle, but $\vec{v}$ doesn't point tangent to the circle at $A$, then there is no way to arrange the feasible region so that there is a closed loop passing through $A$ within the unit box. Thus $C$ must actually be the circle of radius $\frac{1}{2}$. □

## 5.8 Lattice Coding Curves

We'll now turn our attention to the family of lattice coding curves. This family provides us with a method for studying the problem of coding curves in a more combinatorial way. In small dimensions, lattice curves perform very well compared to the known bounds on $A(n, r)$, but we will also see that in higher dimensions their performance quickly falls away from these bounds.

We'll begin with the observation that the length of a lattice coding curve depends only on the types of tiles in the curve.

**Theorem 5.22.** *The length of a closed lattice $(n, l, r)$-coding curve is of the form $l = 2ra + \frac{r\pi}{2}b$, where $a$ is the number of straight line segment tiles and $b$ is the number of quarter-circle tiles, and $a + b = k^n = (\frac{1}{2r} + 1)^n$.*

*The length of an open lattice $(n, l, r)$-coding curve is of the form $l = 2r + 2ra + \frac{r\pi}{2}b$, where $a$ is the number of (full) straight line segment tiles and $b$ is the number of quarter-circle tiles, and $a + b + 2 = k^n = (\frac{1}{2r} + 1)^n$.*

Since $2 > \frac{\pi}{2}$, this means that the length of a lattice coding curve is bounded above by $2rk^n = 2r(\frac{1}{2r} + 1)^n$. The observation that $2 > \frac{\pi}{2} > 1$ also implies that the best lattice coding curves are those that have the largest possible number of quarter-circle tiles and the fewest number of loose ends. Taking this into account, we can now construct a family of long lattice coding curves that we will refer to as the '$m$-fingers' curves.

**Theorem 5.23.** *For any positive integer $m$, there exist closed lattice $(2, 2m - 1 + \frac{\pi}{2} + \frac{-1+\frac{\pi}{2}}{2m-1}, \frac{1}{4m-2})$-coding curves.*

*Proof.* This family of $m$-fingers curves are formed from $4m$ quarter-circle tiles and $4m^2 - 4m$ straight line segment tiles arranged as in *Figure* 13 with an edge along

the bottom and $m$ vertical 'fingers' pointing up from there. Using the length formula from Theorem 5.22, we see that the total length is

$$\frac{4m^2 - 4m + \pi m}{2m - 1} = 2m - 1 + \frac{\pi}{2} + \frac{-1 + \frac{\pi}{2}}{2m - 1}.$$

□

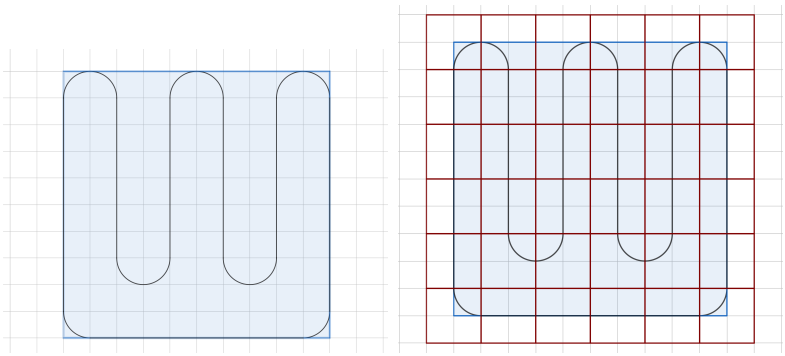

Figure 13: The 3-fingers curve is a $(2, \frac{24+3\pi}{5}, \frac{1}{10})$-coding curve.

We can now use this family of 2-dimensional lattice coding curves to construct higher dimensional coding curves.

**Theorem 5.24.** *Closed lattice* $\left(n, (2m)^{n-2}(2m - 1 + \frac{\pi}{2} + \frac{-1+\frac{\pi}{2}}{2m-1}), \frac{1}{4m-2}\right)$*-coding curves exist for* $n \geq 2$ *and any* $m \in \mathbb{N}$.

*Proof.* We proceed by induction. The base case is the $n = 2$ case proved in Theorem 5.23.

Given an $\left(n-1, (2m)^{n-3}(2m - 1 + \frac{\pi}{2} + \frac{-1+\frac{\pi}{2}}{2m-1}), \frac{1}{4m-2}\right)$-coding curve, we can construct an $\left(n, (2m)^{n-2}(2m - 1 + \frac{\pi}{2} + \frac{-1+\frac{\pi}{2}}{2m-1}), \frac{1}{4m-2}\right)$-coding curve by stacking $2m$ copies of the $(n-1)$-dimensional coding curve on top of each other, thickening each $(n-1)$-dimensional tile into an $n$-dimensional tile, and then rotating pairs of adjacent quarter-circle tiles towards each other to connect the curves together into one long lattice coding

curve of the desired length. Since every $m$-fingers curve has at least two distinct pairs of adjacent quarter-circle tiles, we are able to connect all of the layers together into one continuous curve. □

Although this theorem gives the longest known lattice coding curves, the length still falls away from the Hamming bounds given in subsection 5.6.2, so in higher dimensions there are likely non-lattice coding curves that perform much better.

## 5.9 Analyzing $n = 2$

When $n = 2$, we have a collection of seemingly good coding curves in the $m$-fingers curves, and we have upper bounds coming from the basic and refined volume bounds, so before moving on, it would be useful to compare these lengths and bounds directly. Since the $m$-fingers curves are lattice coding curves, it would also make sense to compare their length to the lattice coding curve upper bound (given by just assuming that all the tiles are straight line segments), but it turns out that for $n = 2$ the lattice coding curve upper bound and the basic volume bound coincide. For larger $n$, the lattice bound is much smaller than the volume bounds (especially for small $r$), so in general lattice coding curves do not do particularly well even though they seem to perform well for $n = 2$.

In Table 10, we can see that, as expected, the two bounds converge for large $m$ (i.e. for small $r$), but the length of the $m$-fingers curves are actually just short of these bounds by $2 - \frac{\pi}{2}$. This difference comes from the fact that the $r$-neighborhoods of the $m$-fingers curves leave small gaps at the bases of the fingers and between the tips of the fingers. These gaps probably cannot be avoided, so we conjecture that the $m$-fingers curves are

| | |
|---|---|
| $m$-fingers Length | $2m - 1 + \frac{\pi}{2} + \frac{-1+\frac{\pi}{2}}{2m-1}$ |
| Basic Hamming Bound | $2m + 1 + \frac{1}{2m-1}$ |
| Refined Hamming Bound | $2m + 1 + \frac{\frac{\pi}{4}}{2m-1}$ |

Table 10: A comparison of the length of an $m$-fingers curve and the two Hamming bounds.

the longest lattice coding curves in 2-dimensions.

## 5.10 Connections to Sphere Packing

As we discussed earlier, our coding curves are a generalization of the traditional sphere packing and maximum separation problems. Since each of these problems has already been studied extensively, it makes sense to look for ways to leverage those results to prove statements about coding curves. This section contains theorems in that direction.

To begin with, we observe that a sphere packing in $n$ dimensions can be extended to a cylinder packing in $n + 1$ dimensions.

**Theorem 5.25.** *Suppose that there exists an arrangement of $k$ points in a unit box in $\mathbb{R}^n$ with minimum distance $2r$. Then $A_k(n + 1, r) \geq k$.*

*Proof.* This follows from constructing cylinders of height 1 over each of the $k$ balls in $\mathbb{R}^n$. □

This idea of just stretching a sphere packing into a cylinder packing one dimension higher gives a very quick way to find a lower bound on the optimal coding curve packings. There are also more complicated constructions that can do slightly better. We'll

highlight just one more here.

**Theorem 5.26.** *Suppose that there exists an arrangement of $k$ points in a unit box in $\mathbb{R}^n$ with minimum distance $2r$, where $\left\lfloor \frac{1}{2r} + 1 \right\rfloor$ is even. Then*

$$A_{\left(\frac{k}{2}\right)\left\lfloor \frac{1}{2r}+1 \right\rfloor}(n+2, r) \geq \frac{k}{2} \left\lfloor \frac{1}{2r} + 1 \right\rfloor (2 - 4r + 2\pi r).$$

*Proof.* First, given the packing of $k$ points into a unit $n$-cube, we can form a packing of $k \left\lfloor \frac{1}{2r} + 1 \right\rfloor$ points in an $(n+1)$-cube by stacking $\left\lfloor \frac{1}{2r} + 1 \right\rfloor$ copies of the $n$-dimensional packing on top of each other. Now, just as in Theorem 5.25, we could stretch this packing into a packing of cylinders in an $(n+2)$-cube, but we can actually do slightly better by pairing up adjacent pairs of points in the $(n+1)$-dimensional packing. Instead of simply stretching these points into straight edges one dimension higher, we can cap the ends with semicircles of radius $r$, using each pair of points to form a single loop in an $(n+2)$-dimensional curve packing. The length of this loop comes from taking two edges of length 1, removing four end segments of length $r$, and then adding two semicircular caps of length $\pi r$ to connect the curves together again. This gives a total length of $2 - 4r + 2\pi r$ for each pair, for a total length of $\frac{k}{2} \left\lfloor \frac{1}{2r} + 1 \right\rfloor (2 - 4r + 2\pi r)$ from a total of $\frac{k}{2} \left\lfloor \frac{1}{2r} + 1 \right\rfloor$ curves of reach $r$ in an $(n+2)$-dimensional unit cube. □

There are other similar results that would come from different strategies for bending the ends together.

We can also work in the opposite direction. Given a coding curve with reach $r$, we can also place balls of radius $r$ along the curve to obtain a (probably non-optimal) packing of spheres. The exact number of spheres that would fit along the curve depends on the curvature pattern of the curve, since you can fit more spheres per curve length for a straight curve than a highly curved one.

## 5.11 Manin Curves

In this section, we'll briefly explore an analogy to the Manin curves in section 2.8. For each $n$, we consider the graph of reach vs. length, plotting a point in the first quadrant for every attainable $(n, l, r)$-coding curve. Our goal is to understand the region of attainable points.

We can make the following observations about the region of attainable points:

- Every point $(r, l)$ with $l < \sqrt{n}$ is attainable, since we have the diagonal segment as a coding curve.
- Given an attainable point $(r, l)$, any point $(r', l)$ with $r' < r$ is also attainable, since a coding curve of reach $r$ is also a coding curve of every smaller reach.
- Given an attainable point $(r, l)$, any point $(sr, sl)$ with $s < 1$ is also attainable, since we can scale a curve by a factor of $s$ to simultaneously shorten it and reduce its reach.

Together, these observations show that the space of attainable points is connected and path connected. Just as it is for block codes, the shape of the curve bounding this region remains unknown.

**Question 5.27.** *For each dimension $n$, what curve bounds the upper edge of this attainable region?*

## 5.12 Simulations

Now that we've developed this theory of coding curves to the point that we have reasonable bounds and a handful of good coding curves, we should ask whether these curves can now be applied to the setting of ToF cameras that originally inspired this work. Fortunately, Felipe Gutierrez-Baragan and Mohit Gupta were able to assist us with using their ToF camera simulation software to test some of these curves. Figure 14 shows the results of one of these simulations comparing the performance of three curves: the industry standard CosCosK4 curve, the Hamiltonian curve, and the saddle curve developed in this chapter.

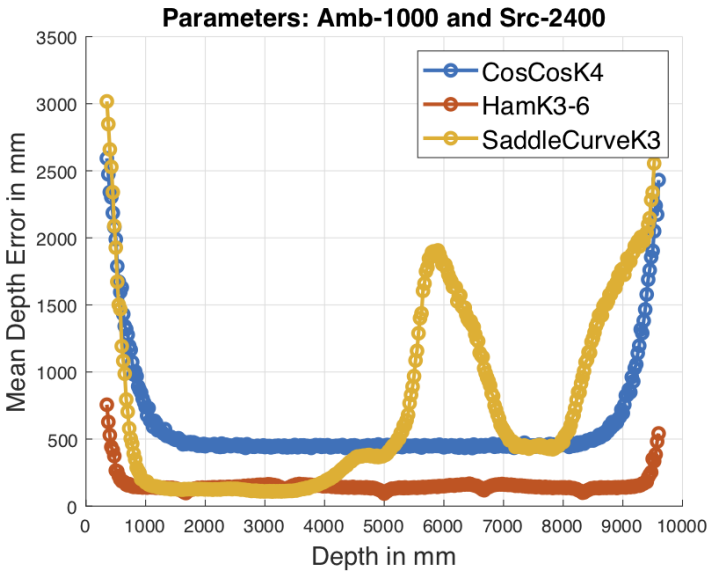


Figure 14: A comparison of the depth errors in a ToF camera simulation for the industry standard curve CosCosK4, the Hamiltonian curve, and the saddle curve.

A first glance at this diagram will quickly reveal two things: that all of these codes perform poorly at the ends of the depth range, and that something strange is happening in the saddle curve once the depth gets beyond about 4000mm. The poor performance

at the ends of the depth range is a result of the fact that all of these curves are closed, so the beginning of the parametrization and the end both converge upon the same point. This would be improved by using open coding curves.

Before looking into the saddle curve's performance on the right hand side of the plot, let's first observe that in the range from about 1000mm to 4000mm the Hamiltonian and saddle curves both perform significantly better than the industry standard, showing a factor of 10 improvement on the depth resolution. In fact, for much of this region, the saddle curve performs slightly better than the Hamiltonian curve and without the variations in resolution that arise at the corners of the Hamiltonian curve. However, once the distance to the scene exceeds about 4000mm, the saddle curve seems to go a bit haywire.

This behavior is due to the fact that the abstraction we used to develop our theory of coding curves left off a couple important features from the real-world setting of ToF cameras. In particular, at no point did we discuss the effects of ambient light or albedo of the surface. One could think of our coding curve model as assuming that both the ambient light and albedo are already known. In the case that they are, decoding from a noisy point in the measurement space onto the saddle curve can be done with just a straightforward and efficient geometric projection.

Unfortunately, in real applications (and even the simulation we used for the above plot), ambient light and albedo have to be taken into account. In this case where we're using three measurement functions, we can understand the effect of ambient light and albedo as follows: first the albedo will multiply the coordinates of the original point in the measurement space by a factor $\alpha \in [0, 1]$, and then ambient light will add a vector $\beta \cdot (1, 1, 1)$ for some $\beta \in [0, 1]$ to the coordinates of the original point. The decoding

algorithm must take this into account and attempt to find the most likely starting point in the case that the values of $\alpha$ and $\beta$ are unknown. As a result, curves perform much better when they do not contain multiple points in the same direction from the origin and they don't contain multiple points whose difference is of the form $\beta \cdot (1, 1, 1)$. To visualize this problem, Figure 15 shows the projections of the saddle curve and the Hamiltonian curve onto the sphere of radius 1.

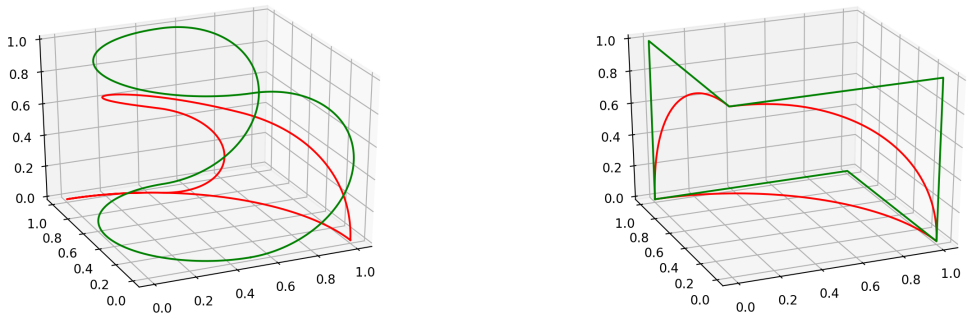


Figure 15: The saddle curve and Hamiltonian curve in green, along with their projections onto the unit sphere in red.

As you can see, the Hamiltonian curve's projection traces out the boundary of the intersection of the unit sphere and the unit box[3], but the saddle curve instead traces out a curve with a large cusp. This cusp corresponds to the distances where we saw poor performance in Figure 14. There are a couple ways to deal with this issue. One option is to restrict the family of curves considered in the theory of coding curves we discussed above to only include those curves that have good projections onto this unit sphere, i.e. those that don't produce ambiguous distances when ambient light and albedo are

[3]In fact, the vertices $(0, 0, 0)$ and $(1, 1, 1)$ were left off of this Hamiltonian path because of these issues with the unknown albedo and ambient light.

considered. Another option would be to perform some channel estimation as part of the data collection process. For example, if 30ms are allocated for the three measurements, instead of using 10ms for each measurement, we could use 1.5ms to estimate the ambient light, 1.5ms to estimate the albedo, and then use 9ms for each measurement. This would also have the benefit of allowing us to use the more geometric and efficient decoding methods that were available before we added in the albedo and ambient light.

In fact, this suggests an entirely new strategy for taking ToF camera measurements. While current methods all allocate the same amount of time for each measurement, this is not a necessary assumption in the theory. We could then ask, "what partitioning of the time given to the different measurements would lead to the best depth resolution?"

## 5.13 Extensions

There are many open directions of research suggested by the work in this chapter. Here we will highlight just a few that we think are most worthy of future research.

Our decision to use the reach of a curve to measure self-avoidingness is not the only choice available. We chose it in order to give a clear guarantee on the decodability of a received point. It's clear, however, that there are curves with poor reach that still perform well in practice. For example, the Hamiltonian curve has reach 0, but has very good properties for ToF cameras. So are there other measures of self-avoidingness that would better match the ToF camera setting? Some form of 'expected reach' might take into account that it's not much of an issue if reach problems only come up for a handful of points along the curve, but there may be better options.

We've also assumed in this section that the coding curves were being packed into a

unit box. This is certainly not a necessary assumption either, and the idea of allocating time non-uniformly for the measurements of ToF cameras suggests that there may be a practical reason to drop this assumption. So another line of research would be to investigate the optimal lengths of coding curves in rectangular prisms or other spaces.

Finally, the restriction that we placed on the reach of a curve enforced a restriction on the local curvature as a result of Theorem 5.3. That being said, there's not any reason why we would need the reach restriction and the local curvature restriction to match in general. We could certainly produce curves where the reach has to be below $r_1$ and the curvature has to be below $r_2$ for some $r_2 < r_1$. What do optimal packings for these thinner but inflexible ropes look like?

# Chapter 6

# Epilogue

*'We are just an advanced breed of monkeys on a minor planet of a very average star. But we can understand the universe. That makes us something very special.'*

(Stephen Hawking)

For the first several decades of coding theory research, the primary goal was meeting Shannon's challenge in a practical way. As of the last few years, this goal has essentially been met. Modern wireless technologies use a variety of LDPC codes, turbo codes, and polar codes to approach the channel capacity with error-correction so quick and accurate that the vast majority of users are hardly aware that these algorithms have been run at all. These codes have been tremendously successful, and as they operate on the binary strings that form the basis for our current computation and communication architectures, they are also broadly applicable.

However, there are still many settings presenting complications that are not addressed by these codes. We've already seen that physical properties of flash memory have inspired a new generation of research into permutation codes under new metrics that are better suited to the expected error patterns. Through our work developing a theory of coding curves for time-of-flight cameras, we've also seen that entirely new coding theories can arise from settings that don't immediately resemble Shannon's communication model. We expect that many more unique settings will arise in the future, demanding new and

artisanal coding theories, which will then inspire a wide variety of new mathematical problems. It is our hope that by formalizing a broader perspective on what makes a coding theory and providing the examples in this work, we will play a small role in accelerating the development of this new generation of coding theory.

# Appendix A

# Permutations

In this section, we'll briefly summarize the notations we'll be using to discuss permutations and provide a few examples to build familiarity.

**Definition A.1.** *A* permutation *on $n$ symbols is a bijective mapping $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$. The set of all permutations on $n$ symbols forms the symmetric group, $S_n$, under the composition operation.*

We'll use the permutation $\alpha \in S_5$ defined in the following table as a running example:

| $i$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\alpha(i)$ | 3 | 4 | 1 | 5 | 2 |

The definition above emphasizes a functional perspective in which a permutation acts on the set of integers between 1 and $n$. We'll frequently use the vector notation for a permutation to connect this perspective to the more traditional perspective of permutations as arrangements of the numbers 1 though $n$.

**Definition A.2.** *The* vector notation *for the permutation $\sigma$ is given by the vector $[\sigma(1), \sigma(2), \ldots, \sigma(n)]$.*

Since we've already defined our example permutation, $\alpha$, using a table, it is straightforward to compute the vector notation for $\alpha$. We simply read off the values from the

second row of the table[1]: $\alpha = [3, 4, 1, 5, 2]$.

In the context of rank modulation codes that we study in chapter 4, this vector notation can be interpreted as the charge ranking of the cells. In other words, the permutation $\alpha$ represents the charge ranking on a collection of 5 cells in which cell 3 has the highest charge, cell 4 has the next highest charge, followed by cell 1, then cell 5, and finally cell 2 has the lowest charge. More concisely, the $i^{\text{th}}$ highest charged cell is the cell $\alpha(i)$.

While the vector notation is very useful for quickly reading off information about the charge rankings of the cells in a rank modulation code, it is probably not the most useful notation for understanding the algebraic structure of the symmetric group. It is more straightforward to compute compositions using the cycle notation.

**Definition A.3.** *A* cycle *of a permutation, $\sigma$, is the sequence of elements in a single orbit of the action of the cyclic group generated by $\sigma$ on $\{1, \ldots, n\}$ listed in the order in which they are permuted by $\sigma$. The* cycle decomposition *of a permutation, $\sigma$, is the list of all non-trivial disjoint cycles of $\sigma$ in the form $\sigma = (a_1, \ldots, a_{k_1})(b_1, \ldots, b_{k_2}) \cdots$. A permutation with $r$ non-trivial cycles of lengths $k_1, k_2, \ldots, k_r$ is said to have cycle type $(k_1, k_2, \ldots, k_r)$.*

Note that this cycle decomposition is not unique, since each cycle can be cyclically permuted and the disjoint cycles commute with each other. We'll typically take the convention of listing each cycle with it's lowest element in the left-most position and list the cycles so that their left-most elements are in increasing order. For example, the canonical cycle notation for our example permutation $\alpha$ is $\alpha = (1, 3)(2, 4, 5)$, and we can

[1] In cases where $n < 10$, we'll often drop the commas from this and the upcoming cycle notation.

see that $\alpha$ has cycle type $(2, 3)$.

Now, given a second permutation $\beta = (2, 3)$, we can compute the product $\alpha\beta$ by concatenating their cycle notations, and then following the action of each cycle from right to left on the elements of $\{1, \ldots, n\}$. This gives us $\alpha\beta = (1, 3)(2, 4, 5)(2, 3) = (1, 3, 4, 5, 2)$. Of course, the symmetric group is not abelian, we could also multiply $\alpha$ and $\beta$ in the other order to get a different permutation $\beta\alpha = (2, 3)(1, 3)(2, 4, 5) = (1, 2, 4, 5, 3)$.

The cycle notation is also very convenient for computing inverses, since we can simply read each cycle in the opposite order. So if $\alpha = (1, 3)(2, 4, 5)$, then $\alpha^{-1} = (1, 3)(2, 5, 4)$.

In order to understand the Kendall tau metric in chapter 4, we'll also need to understand the inversions of a permutation.

**Definition A.4.** *An* inversion *of a permutation $\sigma$ is a pair $(i, j)$ with $i < j$ and $\sigma(j) < \sigma(i)$.*

Our example permutation $\alpha = [3, 4, 1, 5, 2]$ has the five inversions $(1, 3)$, $(1, 4)$, $(2, 3)$, $(2, 4)$, and $(2, 5)$. The number of inversion in a permutation can be counted by hand from the vector notation as we've done here, or by counting the number of adjacent transpositions that would be required to sort the vector notation into the vector notation of the identity permutation using the bubble-sort algorithm. Another intuitive way to understand the number of inversions is to use the arrow notation of a permutation.

**Definition A.5.** *The* arrow notation *of a permutation $\sigma$ lists a top and bottom row of $n$ nodes numbered $1, \ldots, n$, and then attaches an arrow from each symbol in the top row to its image under $\sigma$ in the bottom row.*

The arrow notation of $\alpha = (1,3)(2,4,5)$ is given in Figure 16, and we can very quickly see that there are five crossings corresponding to the five inversions.

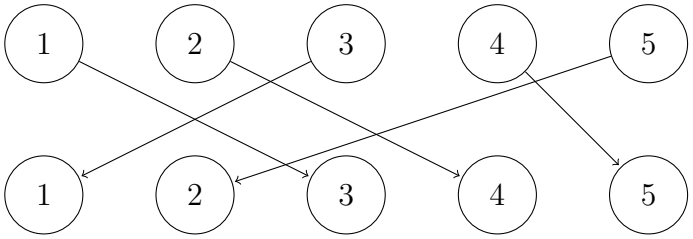


Figure 16: The arrow diagram notation for the permutation $\alpha = [3,4,1,5,2]$.

These arrow notations can also be used to illustrate compositions of permutations by stacking the arrow diagrams on top of each other and following the composite paths. As we've already demonstrated how to compose permutations using either the functional interpretation or using the cycle notation, we won't illustrate the arrow notation composition method here.

Our main purpose for discussing the arrow notation is to clarify a particular operation we'll make use of in chapter 4. In that chapter, we will refer to inserting a permutation into another at some location and then reindexing. For example, we might identify a particular location $*$ in $\alpha = [3,4,1,5,2]$ by writing $\alpha$ as $\alpha = [3,4,1,*,5,2]$. We could then suggest inserting a permutation $\beta = [2,1]$ into this position $*$ and reindexing. We've illustrated this operation in Figure 17. The steps are as follows: Write down the arrow notations for both permutations, write a version of the arrow diagram for $\alpha$ with extra space for a permutation of size $\beta$ at location $*$, then insert the permutation $\beta$ into this space, and finally renumber the top and bottom rows to give the arrow notation for a permutation in $S_{n+k}$, where $\alpha \in S_n$ and $\beta \in S_k$.

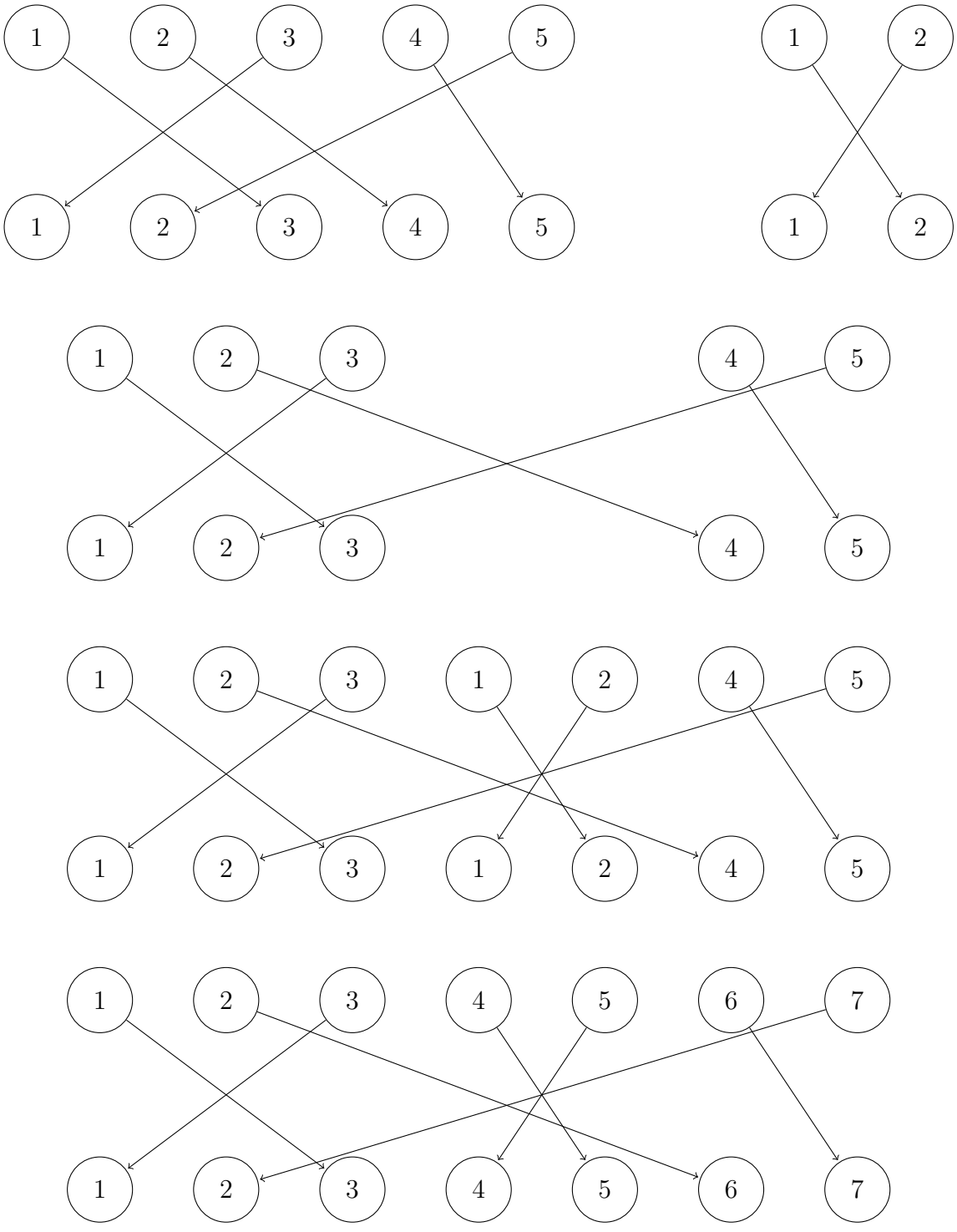


Figure 17: Inserting the permutation $[2, 1]$ into the permutation $[3, 4, 1, *, 5, 2]$ at the position marked $*$ produces the permutation $[3, 6, 1, 5, 4, 7, 2]$.

# Bibliography

[1] Erdal Arikan, *Channel polarization: A method for constructing capacity-achieving codes*, 2008 IEEE International Symposium on Information Theory, IEEE, 2008, pp. 1173–1177.

[2] RM Baer and P Brock, *Natural sorting over permutation spaces*, Mathematics of Computation **22** (1968), no. 102, 385–410.

[3] Alexander Barg and Arya Mazumdar, *Codes in permutations and error correction for rank modulation*, 2010 IEEE International Symposium on Information Theory, IEEE, 2010, pp. 854–858.

[4] Claude Berrou, Alain Glavieux, and Punya Thitamajshima, *Near Shannon limit error-correcting coding and decoding: Turbo-codes*, Proceedings of ICC'93-IEEE International Conference on Communications, vol. 2, IEEE, 1993, pp. 1064–1070.

[5] Ian F Blake, Gérard Cohen, and Mikhail Deza, *Coding with permutations*, Information and Control **43** (1979), no. 1, 1–19.

[6] Raj Chandra Bose and Dwijendra K Ray-Chaudhuri, *On a class of error correcting binary group codes*, Information and Control **3** (1960), no. 1, 68–79.

[7] J Bourget, *Des permutations*, Nouvelles annales de mathématiques: journal des candidats aux écoles polytechnique et normale **10** (1871), 254–268.

[8] Sarit Buzaglo and Tuvi Etzion, *Bounds on the size of permutation codes with the*

*Kendall tau-metric*, IEEE Transactions on Information Theory **61** (2015), no. 6, 3241–3250.

[9] Yu Cai, Erich F Haratsch, Onur Mutlu, and Ken Mai, *Error patterns in MLC NAND flash memory: Measurement, characterization, and analysis*, Proceedings of the Conference on Design, Automation, and Test in Europe, EDA Consortium, 2012, pp. 521–526.

[10] Peter J Cameron, *Permutation codes*, European Journal of Combinatorics **31** (2010), no. 2, 482–490.

[11] Jason Cantarella, Joseph HG Fu, Robert B Kusner, and John M Sullivan, *Rope-length criticality*, Geometry & Topology **18** (2014), no. 4, 2595–2665.

[12] Jason Cantarella, Robert B Kusner, and John M Sullivan, *On the minimum rope-length of knots and links*, Inventiones Mathematicae **150** (2002), no. 2, 257–286.

[13] Julia Chifman, *Note on direct products of certain classes of finite groups*, Communications in Algebra **37** (2009), no. 5, 1831–1842.

[14] Wensong Chu, Charles J Colbourn, and Peter Dukes, *Constructions for permutation codes in powerline communications*, Designs, Codes and Cryptography **32** (2004), no. 1-3, 51–64.

[15] Richard Dedekind, *Über Gruppen, deren sämmliche Theiler Normaltheiler sind.*, Mathematische Annalen **48** (1897), no. 4, 548–561.

[16] Philippe Delsarte, *Bounds for unrestricted codes, by linear programming*, Philips Res. Rep **27** (1972), 272–289.

[17] Erik D Demaine and Joseph O'Rourke, *Open problems from CCCG 2010*, Proceedings of the 23rd Canadian Conference on Computational Geometry (CCCG 2011), August 10–12 2011.

[18] Elizabeth Denne, Yuanan Diao, and John M Sullivan, *Quadrisecants give new lower bounds for the ropelength of a knot*, Geometry & Topology **10** (2006), no. 1, 1–26.

[19] Michael Deza and Tayuan Huang, *Metrics on permutations, a survey*, Journal of Combinatorics, Information and System Sciences, Citeseer, 1998.

[20] Persi Diaconis, *Group representations in probability and statistics*, Lecture notes-monograph series **11** (1988), i–192.

[21] Yuanan Diao, Claus Ernst, and Xingxing Yu, *Hamiltonian knot projections and lengths of thick knots*, Topology and its Applications **136** (2004), no. 1-3, 7–36.

[22] ST Dougherty, Jon-Lark Kim, and Patrick Solé, *Open problems in coding theory*, Contemp. Math **634** (2015), 79–99.

[23] Jarek Duda, *Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding*, arXiv preprint arXiv:1311.2540 (2013).

[24] Paul H Edelman, *On inversions and cycles in permutations*, European Journal of Combinatorics **8** (1987), no. 3, 269–279.

[25] Robert M Fano, *The transmission of information*, Massachusetts Institute of Technology, Research Laboratory of Electronics (1949).

[26] Farzad Farnoud, Vitaly Skachek, and Olgica Milenkovic, *Error-correction in flash memories via codes in the Ulam metric*, IEEE Transactions on Information Theory **59** (2013), no. 5, 3003–3020.

[27] Peter Frankl and Mikhail Deza, *On the maximum number of permutations with a given maximal or minimal distance*, Journal of Combinatorial Theory, Series A **22** (1977), no. 3, 352–360.

[28] Robert Gallager, *Low-density parity-check codes*, IRE Transactions on Information Theory **8** (1962), no. 1, 21–28.

[29] Henryk Gerlach and Heiko von der Mosel, *On sphere-filling ropes*, The American Mathematical Monthly **118** (2011), no. 10, 863–876.

[30] Marcel JE Golay, *Notes on digital coding*, Proc. IEEE **37** (1949), 657.

[31] Faruk Göloğlu, Jüri Lember, Ago-Erik Riet, and Vitaly Skachek, *New bounds for permutation codes in Ulam metric*, 2015 IEEE International Symposium on Information Theory (ISIT), IEEE, 2015, pp. 1726–1730.

[32] Oscar Gonzalez and Raphael de la Llave, *Existence of ideal knots*, Journal of Knot Theory and its Ramifications **12** (2003), no. 01, 123–133.

[33] Oscar Gonzalez and John H Maddocks, *Global curvature, thickness, and the ideal shapes of knots*, Proceedings of the National Academy of Sciences **96** (1999), no. 9, 4769–4773.

[34] Oscar Gonzalez, John H Maddocks, and J Smutny, *Curves, circles, and spheres*, Contemporary Mathematics **304** (2002), 195–216.

[35] VD Goppa, *Bounds for codes*, Dokl. Acad. Nauk SSSR **333** (1993).

[36] Mohit Gupta, Andreas Velten, Shree K Nayar, and Eric Breitbach, *What are optimal coding functions for time-of-flight imaging?*, ACM Transactions on Graphics (TOG) **37** (2018), no. 2, 13.

[37] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan, *Essential coding theory*, Draft available at `https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/`, 2019.

[38] Felipe Gutierrez-Barragan, Syed Azer Reza, Andreas Velten, and Mohit Gupta, *Practical coding function design for time-of-flight imaging*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1566–1574.

[39] John M Hammersley et al., *A few seedlings of research*, Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics, The Regents of the University of California, 1972.

[40] Richard W Hamming, *Error detecting and error correcting codes*, Bell System Technical Journal **29** (1950), no. 2, 147–160.

[41] Alexis Hocquenghem, *Codes correcteurs d'erreurs*, Chiffres **2** (1959), no. 2, 147–156.

[42] Alexander E Holroyd, *Perfect snake-in-the-box codes for rank modulation*, IEEE Transactions on Information Theory **63** (2016), no. 1, 104–110.

[43] Michal Horovitz and Tuvi Etzion, *Constructions of snake-in-the-box codes for rank modulation*, IEEE Transactions on Information Theory **60** (2014), no. 11, 7016–7025.

[44] David A Huffman, *A method for the construction of minimum-redundancy codes*, Proceedings of the IRE **40** (1952), no. 9, 1098–1101.

[45] Kenkichi Iwasawa, *Über die endlichen Gruppen und die Verbände ihrer Untergruppen*, J. Fac. Sci. Imp. Univ. Tokyo. Sect. I **4** (1941), 171–199.

[46] Anxiao Jiang, Robert Mateescu, Moshe Schwartz, and Jehoshua Bruck, *Rank modulation for flash memories*, IEEE Transactions on Information Theory **55** (2009), no. 6, 2659–2673.

[47] ———, *Rank modulation for flash memories*, Aug 2012, US Patent 8,245,094.

[48] Anxiao Jiang, Moshe Schwartz, and Jehoshua Bruck, *Error-correcting codes for rank modulation*, 2008 IEEE International Symposium on Information Theory, IEEE, 2008, pp. 1736–1740.

[49] ———, *Correcting charge-constrained errors in the rank-modulation scheme*, IEEE Transactions on Information Theory **56** (2010), no. 5, 2112–2120.

[50] M Ryan Julian Jr, *No MacWilliams duality for codes over nonabelian groups*, Journal of Algebra Combinatorics Discrete Structures and Applications **5** (2017), no. 1, 45–49.

[51] L. Karlitz, *Permutations with prescribed pattern*, Mathematische Nachrichten **58** (1973), 31–53.

[52] Han Mao Kiah, Gregory J Puleo, and Olgica Milenkovic, *Codes for DNA sequence profiles*, IEEE Transactions on Information Theory **62** (2016), no. 6, 3125–3146.

[53] Justin Kong, *Extending lp–decoding for permutation codes from Euclidean to Kendall tau metric*, Plan B paper, M.A., Mathematics, University of Hawaii at Manoa (2012), 1–28.

[54] Justin Kong and Manabu Hagiwara, *Nonexistence of perfect permutation codes in the Ulam metric*, 2016 International Symposium on Information Theory and Its Applications (ISITA), IEEE, 2016, pp. 691–695.

[55] Vladimir I Levenshtein, *On perfect codes in deletion and insertion metric*, Discrete Mathematics and Applications **2** (1992), no. 3, 241–258.

[56] Fabian Lim and Manabu Hagiwara, *Linear programming upper bounds on permutation code sizes from coherent configurations related to the Kendall tau distance metric*, 2012 IEEE International Symposium on Information Theory Proceedings, IEEE, 2012, pp. 2998–3002.

[57] Percy Alexander MacMahon, *The indices of permutations and the derivation therefrom of functions of a single variable associated with the permutations of any assemblage of objects*, American Journal of Mathematics **35** (1913), no. 3, 281–322.

[58] FJ MacWilliams, NJA Sloane, and J-M Goethals, *The MacWilliams identities for nonlinear codes*, Bell System Technical Journal **51** (1972), no. 4, 803–819.

[59] Florence Jessie MacWilliams, *A theorem on the distribution of weights in a systematic code*, Bell System Technical Journal **42** (1963), no. 1, 79–94.

[60] Florence Jessie MacWilliams and Niel James Alexander Sloane, *The theory of error-correcting codes*, vol. 16, Elsevier, 1977.

[61] Yuri Ivanovitch Manin, *What is the maximum number of points on a curve over F2*, J. Fac. Sci. Univ. Tokyo **28** (1981), 715–720.

[62] Amos Maritan, Cristian Micheletti, Antonio Trovato, and Jayanth R Banavar, *Optimal shapes of compact strings*, Nature **406** (2000), no. 6793, 287.

[63] Rudolf Mathon and Tran Van Trung, *Directed t-packings and directed t-Steiner systems*, Designs, Codes and Cryptography **18** (1999), no. 1-3, 187–198.

[64] Roger H Moritz and Robert C Williams, *A coin-tossing problem and some related combinatorics*, Mathematics Magazine **61** (1988), no. 1, 24–29.

[65] Piotr Pierański, *In search of ideal knots*, Computational Methods in Science and Technology **4** (1998), 9–23.

[66] Netanel Raviv, Moshe Schwartz, and Eitan Yaakobi, *Rank modulation codes for DNA storage*, 2017 IEEE International Symposium on Information Theory (ISIT), IEEE, 2017, pp. 3125–3129.

[67] Eric J Rawdon, *Can computers discover ideal knots?*, Experimental Mathematics **12** (2003), no. 3, 287–302.

[68] Irving S Reed and Gustave Solomon, *Polynomial codes over certain finite fields*, Journal of the Society for Industrial and Applied Mathematics **8** (1960), no. 2, 300–304.

[69] Tom Richardson and Ruediger Urbanke, *Modern coding theory*, Cambridge university press, 2008.

[70] Jorma J Rissanen, *Generalized Kraft inequality and arithmetic coding*, IBM Journal of Research and Development **20** (1976), no. 3, 198–203.

[71] Olinde Rodrigues, *Note sur les inversions, ou dérangements produits dans les permutations*, J. de Math **4** (1839), 236–240.

[72] DG Rogers, *Ascending sequences in permutations*, Discrete Mathematics **22** (1978), no. 1, 35–40.

[73] Steven Roman, *Coding and information theory*, vol. 134, Springer Science & Business Media, 1992.

[74] Craige Schensted, *Longest increasing and decreasing subsequences*, Canadian Journal of Mathematics **13** (1961), 179–191.

[75] Roland Schmidt, *Subgroup lattices of groups*, vol. 14, Walter de Gruyter, 1994.

[76] Claude Elwood Shannon, *A mathematical theory of communication*, Bell System Technical Journal **27** (1948), no. 3, 379–423.

[77] Richard Singleton, *Maximum distance q-nary codes*, IEEE Transactions on Information Theory **10** (1964), no. 2, 116–118.

[78] Richard P Stanley, *Binomial posets, Möbius inversion, and permutation enumeration*, Journal of Combinatorial Theory, Series A **20** (1976), no. 3, 336–356.

[79] Andrzey Stasiak and John H Maddocks, *Mathematics: best packing in proteins and DNA*, Nature **406** (2000), no. 6793, 251.

[80] John M Sullivan, *Approximating ropelength by energy functions*, Contemporary Mathematics **304** (2002), 181–186.

[81] Michio Suzuki, *On the lattice of subgroups of finite groups*, Transactions of the American Mathematical Society **70** (1951), no. 2, 345–371.

[82] Itzhak Tamo and Moshe Schwartz, *On the labeling problem of permutation group codes under the infinity metric*, IEEE Transactions on Information Theory **58** (2012), no. 10, 6595–6604.

[83] Aimo Tietäväinen, *On the nonexistence of perfect codes over finite fields*, SIAM Journal on Applied Mathematics **24** (1973), no. 1, 88–96.

[84] JH Van Lint, A Dold, and B Eckmann, *Coding theory*, Springer, 1971.

[85] Lekh R Vermani, *Elements of algebraic coding theory*, vol. 12, CRC Press, 1996.

[86] Gérard Viennot, *Maximal chains of subwords and up-down sequences of permutations*, Journal of Combinatorial Theory, Series A **34** (1983), no. 1, 1–14.

[87] SG Vleduts and Yu I Manin, *Linear codes and modular curves*, Journal of Soviet Mathematics **30** (1985), no. 6, 2611–2643.

[88] Xin Wang, Yiwei Zhang, Yiting Yang, and Gennian Ge, *New bounds of permutation codes under Hamming metric and Kendall's $\tau$-metric*, Designs, Codes and Cryptography **85** (2017), no. 3, 533–545.

[89] J Wood, *Lecture notes on the MacWilliams identities and extension theorem*, 2008.

[90] Yonatan Yehezkeally and Moshe Schwartz, *Snake-in-the-box codes for rank modulation*, IEEE Transactions on Information Theory **58** (2012), no. 8, 5471–5483.

[91] Giovanni Zacher, *Caratterizzazione dei gruppi immagini omomorfe duali di un gruppo finito*, Rendiconti del Seminario Matematico della Università di Padova **31** (1961), 412–422.

[92] VA Zinoviev and T Ericson, *On Fourier-invariant partitions of finite abelian groups and the MacWilliams identity for group codes*, Problems of Information Transmission **32** (1996), no. 1, 117–122.