Beyond unique decoding: topics in error-correcting codes

Carol Wang

CMU-CS-15-136

September 2015

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Venkatesan Guruswami, Chair
Ryan O'Donnell
Bernhard Haeupler
Po-Shen Loh
Madhu Sudan, Microsoft Research

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under grant numbers CCF-0953155, CFF-0963975, CCF-1016799, CFF-1115525, DGE-0750271, DGE-1252522; United States-Israel Binational Science Foundation under grant number 2008293; and Microsoft.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Error-correcting codes, algebraic coding, list decoding

For my parents.

Abstract

Error-correcting codes give efficient ways to store and recover information, even when the information has been corrupted. They have seen wide applicability in areas like software and communication, where they allow for improvements in both redundancy and resilience.

This thesis covers various areas in the field of error-correcting codes, addressing problems which arise in different applications and error models. One such area is that of list-decoding, a model in which the decoder may output multiple possible decodings, allowing for correction from a larger number of errors. We give an explicit construction of good codes which are efficiently list-decodable up to an information theoretically optimal fraction of errors. The framework we have developed for decoding, the linear-algebraic method, has proven to be a powerful tool for designing and decoding codes.

We extend our techniques to construct the first nontrivially list-decodable codes with high rate for the rank metric, a model which has applications in wireless network communication. We also construct the first explicit deletion codes correcting a constant fraction of deletions with rate approaching one, and correcting a fraction of deletions approaching one with constant rate. The central theme of this thesis is that effective communication is possible, even for these very different models.

Acknowledgements

Many thanks to my advisor, Venkat, who has always been helpful and patient, who has tolerated all of my research whims with good humor. In these years of imposing on his time, I have learned a lot about how to do research, and maybe (in direct opposition to his sensible guidance) how not to do research.

Thanks are due as well to my thesis committee, and to all of the researchers who have given me their time and expertise, most notably: Anupam Gupta, who advised me in my first year; Suguru Tamaki, my mentor during a research fellowship at Kyoto University; and Sergey Yekhanin, my mentor during an internship at Microsoft Research. I also owe a lot to my co-authors: Venkat, Srivatsan, Madhu, Ameya, and Chaoping.

In life beyond research, I am grateful to my enabler friends (you know who you are). Without these friends, I would have gotten a lot more work done, but these past years would not have been nearly as fun. Of course my biggest enablers are actually my parents, who have provided unconditional support even as my work becomes less and less comprehensible. And a close contender for biggest enabler is Luke, an always reliable source of moral support and sugar.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In which we meet Alice and Bob • Mallory is malicious • Coding theory comes to the rescue

This thesis is concerned with the following scenario: Alice has a message which she fervently wishes to communicate to Bob. Between them stands Mallory, who aims to foil Alice by tampering with whatever she sends. Alice's goal is somehow to protect her message so that Bob can recover useful information, even after Mallory has introduced errors into the transmission. One way she can do this is by adding redundancy; that is, by sending some extra helpful information. On the opposing side, in the interests of economy, she would like to use as little redundancy as possible.

The theory of error-correcting codes seeks to understand what Alice and Bob (or, in another life, the *sender and receiver* or *encoder and decoder*) can hope to achieve. We are interested not only in the kinds of errors from which we can recover, but also in how each step of the process can be done efficiently, in polynomial time. Aside from the natural applications in communication and data storage, advances in coding theory have played an important role in developments in areas such as pseudorandomness and complexity theory.

One of the most common settings studied in coding theory is one in which the adversary, Mallory, can corrupt a fixed number of letters or symbols in what is being transmitted, and the receiver must recover the original message, no matter which symbols have been corrupted. For example, one easy way for Alice to protect, or *encode*, her message, is to send three copies of the message. Although she has to send three times as many symbols, Bob can determine the original message even if one transmitted symbol has been changed. He can do this simply by checking every symbol against its two redundant copies. On the other hand, if Mallory is allowed to corrupt two symbols, this strategy will not always

work, and the original message may not be recoverable.

Thus far, we have two possibilities for Alice. If she has a message of n symbols, she can send those symbols to Bob, but if any errors occur, he may not be able to decode the message. We've seen that she can send more symbols ($3n$ symbols) in exchange for being able to handle a single error. This naturally leads us to wonder: Can Alice do better? In other words, if she needs to handle a single error, can she send fewer than $3n$ symbols? What if she expects a growing number of errors, say $n/10$? Does the answer change if Alice's encoding and Bob's decoding both have to be efficient?

As it turns out, the affairs of Alice and Bob have been of interest to coding theorists for a long time, and we know quite a bit about the answers to these questions. In this thesis, we will consider how these answers change in new settings where the definitions of "error" and "decoding" can be quite different. These definitions arise in different applications, but the fundamental goal of efficient communication remains the same. In each of these settings, we study new ways of protecting information from errors, and show how this can be done efficiently.

1.1 Contributions of this thesis

We outline the models we study and our results on coding for these models; more detailed exposition appears in Chapter 2.

1.1.1 List-decodable codes

One drawback of insisting that Bob decodes Alice's original message is that there is no way to handle extremely high error rates. If the adversary is allowed to introduce errors in over half of the symbols, then they can change half of the symbols in whatever Alice sends to match some completely different string, making it impossible for Bob to be certain which was originally sent. It may seem that all is lost, but it turns out even as the error rate approaches 1 (that is, nearly every symbol corrupted), it is still possible for Bob to extract meaningful information about Alice's message, with only a constant-factor increase in transmission length. Our notion of "meaningful information" is captured by the model of *list-decoding*.

Informally, list-decoding allows Bob to output a list of possible decodings. We consider his decoding to be successful if the list is short (polynomial in the message length), and contains Alice's original message. In other words, even if too much information has

been lost to pinpoint the exact message Alice meant to send, Bob should be able to narrow down the possibilities to a manageable number.

As we will see in Chapter 2, not only does this relaxation allow decoding from up to twice as many errors, but the encoding and decoding can be done *efficiently*. The first efficient construction to achieve the optimal trade-off between redundancy and correctable error rate was the folded Reed-Solomon code, a variant of the classical Reed-Solomon code. These codes, with the relaxed model of list-decoding, allow us to handle arbitrarily high error rates.

In this thesis, we give a new construction of list-decodable codes, also based on Reed-Solomon codes. These codes, known as *derivative codes*, also achieve the optimal trade-off between redundancy and error rate while admitting efficient algorithms. In fact, we are able to adapt our algorithms to list-decode folded Reed-Solomon codes as well. An advantage of our approach is that it gives a nice structure to the decoded list, leading to explicit constructions of codes which can guarantee a constant (rather than merely polynomial) list size.

1.1.2 Deletion codes

In the deletion model, rather than having individual symbols be changed, we allow Mallory to remove (delete) some fraction of symbols from the transmission, so that Bob receives a *substring* of what was originally sent. One could think of Alice having typed up a text file, and Mallory being given access to the backspace key — the resulting text shows no sign of which symbols are missing. Under this model, Mallory can easily turn "I don't trust Mallory" into "I trust Mallory" without anyone being the wiser. Even if Bob detects from the brevity of the message that something is amiss, the absent symbols could just as easily belong at the end, perhaps "I trust Mallory a lot."

As this example shows, a major difficulty in handling deletions is that the receiver not only loses the information in the deleted symbols, but information about where they came from; that is, which position in the message they occupied. As it stands, Bob has no (coding-theoretic) reason for disbelieving the second possibility. This turns out to be quite challenging when we are using a small alphabet of symbols, and there are still many basic questions to which we do not know the answers.

In this thesis, we initiate a systematic study of codes against worst-case deletions, showing bounds on what can be achieved combinatorially, and then giving *efficient* codes which are not far from these bounds. Our codes focus on the cases when we want to handle arbitrarily high deletion fractions, and when we want very low redundancy.

1.1.3 Rank-metric and subspace codes

The worst-case model gives the adversary, Mallory, the ability to corrupt what is being sent in the most confusing way possible. In order to allow us to be able to perform meaningful communication, we must constrain the errors in some other fashion. One way to think of these constraints is as imposing *structure* on the errors Mallory may introduce, making our task tractable.

Thus far, we have touched on the case when we restrict the number of symbols which can be affected. In many applications, the kind of errors which occur might be quite different, and the codes we use must change accordingly. For example, one simple class of error patterns is one where Mallory shifts every symbol by a fixed offset (say, $a \to c$, $b \to d$, $c \to e$, etc). These patterns can corrupt *every* symbol, ruling out algorithms which rely on receiving some number of correct symbols, but their rigid structure means that they are still easy to correct.

It turns out that this kind of error pattern arises in communication over linear networks. In this model, we think of the network as a directed graph, and messages as being passed along outgoing edges. The key difficulty is that if a single message is corrupted early on, it may propagate through the network, corrupting everything which the receiver sees. However, we will see that this kind of error (defined more formally in Chapter 5) can be corrected using subspace codes and the related rank-metric codes. Although the two models are different in many ways, we will see that we can achieve many of the same guarantees in the rank-metric case as in the Hamming metric.

In this thesis, we give the *first* explicit construction of rank-metric codes and subspace which are efficiently list-decodable with constant redundancy. In fact, as with derivative codes, we obtain the optimal trade-off between redundancy and correctable error rate. Previously constructed codes for this regime had an exponential list size or required a polynomial blow-up in message length.

1.2 Organization

In Chapter 2, we introduce the basics of error-correcting codes and survey some results in this area, in addition to introducing deletion codes. Our results on list-decodable codes appear in Chapter 3, and results on deletion codes appear in Chapter 4. Chapter 5 introduces rank-metric and subspace codes, and Chapter 6 gives our construction of good list-decodable codes for these models. We conclude in Chapter 7 with some open questions.

Chapter 2

Preliminaries

An abbreviated survey of coding theory • Polynomial codes come to stay

In this chapter, we will develop some of the necessary background on error-correcting codes and the various settings which we consider. For clarity, we will defer discussion of rank-metric codes until Chapter 5. The reader is assumed to have some familiarity with some basic algebra (polynomials over finite fields, etc.) and linear algebra.

Some standard notation before we begin. Let $q = p^n$ for some prime p and integer n. We denote by $\mathbb{F}_q$ the finite field of q elements. The prime p is the *characteristic* of $\mathbb{F}_q$. The notation $\mathbb{F}_q[X]$ refers to the ring of univariate polynomials in X with coefficients in the field $\mathbb{F}_q$.

For functions f, g, the notation $g = O(f)$ means that there is some constant C such that $g(n) \leq Cf(n)$ for sufficiently large n. In this case we may also write $f = \Omega(g)$. All logs will be base 2 unless specified.

For a set Σ, we write Σ^n to denote all n-tuples over Σ; that is, $\Sigma^n = \{(x_1, \ldots, x_n) \mid x_i \in \Sigma\}$. We will often refer to elements of Σ^n as *strings* over Σ.

2.1 Coding basics

At a high level, a good code is simply a set of strings which are difficult to confuse. This section will inject some rigor into this idea, and allow us to begin investigating the possibilities of error correction.

Definition 2.1. A **code** C of block length n over an alphabet Σ is a subset of Σ^n, together with a one-to-one **encoding map** which maps a **message** set M (say, $\{1, 2, \ldots, |C|\}$) to C.

In other words, a code takes messages and encodes them as strings of n symbols, or **codewords**. Our goal for this encoding is to increase the resilience of the messages to errors.

For the rest of this thesis, we will implicitly assume that the alphabet Σ is finite. In fact, most commonly, rather than being an arbitrary set, Σ will be some finite field $\mathbb{F}_q$, or a subspace over $\mathbb{F}_q$. We will make good use of the algebraic structure of the field. For example, we can define the notion of linearity for a code.

Definition 2.2. If Σ is a field, and $C \subseteq \Sigma^n$ is a *subspace* of Σ^n over Σ, then C is a **linear code**.

As alluded to in the previous chapter, the fundamental challenge which drives coding theory is understanding the relationship between the correctable error rate and the redundancy required. We now define these notions more formally, as the *rate* and *distance* of a code.

Definition 2.3. Let Σ have size q. The **rate** of a code $C \subseteq \Sigma^n$ is

$$R(C) := \frac{\log_q |C|}{n}.$$

For example, when the message set is Σ^k, or all strings of k symbols, the rate is simply k/n. Thus the rate measures the efficiency of the encoding, with higher rate being more desirable. Note that the rate is always between 0 and 1, with rate 1 indicating no redundant symbols.

Definition 2.4. The **(Hamming) distance** between two strings $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ in Σ^n is

$$d_H(x, y) := |\{i \mid x_i \neq y_i\}|.$$

That is, the Hamming distance counts how many coordinates differ between the two strings.

Definition 2.5. The **(relative) distance** of a code $C \subseteq \Sigma^n$ is

$$\delta_H(C) := \min_{x \neq y \in C} \frac{d_H(x, y)}{n}.$$

The Hamming distance between two strings tells us how many errors must be introduced to transform one string into the other. Thus the distance of a code is a good measure of its error resilience. As with rate, the distance of a code is between 0 and 1, and higher distance is better.

Because we are interested in being able to encode messages of any length, when we refer to codes, we will actually be referring to **families** of codes of growing block length. More specifically, a family of codes is an infinite sequence $\mathcal{C} = \{C_i\}$ of codes C_i over increasing block lengths n_i. We can define rate and distance for families of codes: the rate is $R(\mathcal{C}) = \liminf_i R(C_i)$, and the relative distance is $\delta_H(\mathcal{C}) = \liminf_i \delta_H(C_i)$.

Throughout this thesis, we will often abuse terminology by referring to a family of codes simply as a code. If the rate and distance of a family of codes are both bounded away from zero, we will call the family (asymptotically) *good*, and we will use the term *positive* or *constant rate* to refer to rates which are bounded away from zero.

Example. One basic class of codes is the repetition codes mentioned in Chapter 1. For any positive integer r, let us define the r-repetition code, which maps any vector in Σ^k to r copies of itself in Σ^{rk}. This gives a family of codes $\{C_i\}$ which is defined for any block length n_i which is a multiple of r.

The rate of the r-repetition code of block length n is $1/r$, and its relative distance is r/n. In particular, for constant r, this family of codes is not asymptotically good; it has positive rate, but the distance goes to zero as block length increases.

As this example suggests, there is a trade-off between the rate and distance of a code. For repetition codes, as we increase r, the rate decreases, but the distance goes up. Thus, it makes sense not only to ask whether asymptotically good codes exist, but also to ask *how* good they can be. We explore this in the next section.

2.1.1 Basic bounds on codes

We ease into investigating the rate-distance trade-off with the classical Singleton bound.

Theorem 2.6 (Singleton bound)**.** Let C be a code of block length n and relative distance δ over an alphabet of size q. Then $|C| \leq q^{(1-\delta)n+1}$. In particular, the rate R of C satisfies

$$R \leq 1 - \delta + 1/n.$$

Proof. We argue that any two codewords c_1, c_2 in C cannot agree in the first $(1-\delta)n + 1$ coordinates. If they do, the Hamming distance between c_1 and c_2 is at most $\delta n - 1$, which

contradicts the minimum distance property of C. Thus, as each codeword is identified by its first $(1-\delta)n+1$ coordinates, $|C| \leq q^{(1-\delta)n+1}$. $\square$

Despite its simplicity, it turns out that the Singleton bound is tight; we will see in Section 2.3 that the Reed-Solomon code meets this bound.

On the other hand, if we are only looking for asymptotically good codes, it turns out that we need not look very hard. One way we can construct fairly large codes is via a greedy algorithm: For a target distance of δ, we begin with an arbitrary codeword in Σ^n. Choosing this codeword excludes all strings within Hamming distance $\delta n - 1$ from being in our code, but we can add any other string to our code. We then continue adding strings which are at distance at least δn from all previously chosen strings.

It is easy to check that the number of strings at distance d from any fixed string is

$$V_q(n,d) := \sum_{i=0}^{d} \binom{n}{i} (q-1)^i.$$

(This is the volume of a *Hamming ball* of radius d.)

As each codeword we add excludes at most $V_q(n, \delta n - 1)$ additional codewords, our greedy procedure terminates only when we run out of possible strings to add to C, or when the final code C satisfies

$$|C| \cdot V_q(n, \delta n - 1) \geq q^n.$$

This gives us the following.

Theorem 2.7 (Gilbert-Varshamov bound)**.** There exist codes of block length n, relative distance δ, and alphabet size q satisfying:

$$|C| \geq \frac{q^n}{V_q(n, \delta n - 1)}.$$

We can also state an asymptotic version of the Gilbert-Varshamov bound using the following:

Definition 2.8. Let $q \geq 2$ be a positive integer. The ***q*-ary entropy function** $h_q \colon [0,1] \to \mathbb{R}$ is

$$h_q(x) = x \log_q(q-1) - x \log_q x - (1-x) \log_q(1-x).$$

As it turns out, the entropy gives us a fairly good estimate for the volume $V_q(n, \delta n)$.

Lemma 2.9. For $q \geq 2$ and $\delta \in (0, 1 - 1/q)$,

$$q^{(h_q(\delta)-o(1))n} \leq V_q(n, \delta n) \leq q^{(h_q(\delta))n}.$$

Combined with Theorem 2.7, we have the following:

Theorem 2.10 (Asymptotic Gilbert-Varshamov bound)**.** For $q \geq 2$ and $\delta \in (0, 1 - 1/q)$, there exists a family of codes C with alphabet size q, distance δ, and rate

$$R(C) \geq 1 - h_q(\delta) - o(1).$$

This gives us asymptotically good families of positive rate $\approx 1 - h_q(\delta)$ and distance δ.

In fact, it turns out that a *random* linear code will meet the Gilbert-Varshamov bound. The greedy construction does, however, have the advantage that it is guaranteed to work, and we will see a version of this construction in Chapter 4 when we investigate bounds on deletion codes.

2.1.2 Unique decoding

Now that we have some idea of what to expect from the rate and distance of our code families, we will formalize our assertion that the distance of a code measures its error resilience.

We will say that a code $C \subseteq \Sigma^n$ is **uniquely decodable** from a τ fraction of errors if it is possible to correct all codewords in C from any pattern of τn symbol errors.

Notice that this is a *worst-case* guarantee; we make no assumptions on how the errors are distributed. There are many lines of work for when errors occur at random, but all of the codes constructed in this thesis will be against the stronger model of adversarial errors. In this setting, it is easy to see that the following characterizes which errors can be corrected.

Fact. A code $C \subseteq \Sigma^n$ can be uniquely decoded from a τ fraction of errors if and only if C has distance greater than 2τ.

A pictorial representation of this fact can be seen in Figure 2.1.2.

One corollary of this fact is that *no* code with more than one codeword can be uniquely decoded from a $1/2$ fraction of errors. However, in practice we may expect to see higher error rates, and it turns out that even over a $1/2$ error rate, we can still transmit meaningful information. What we mean by "meaningful" is captured by the model of list-decoding, defined in the next section.

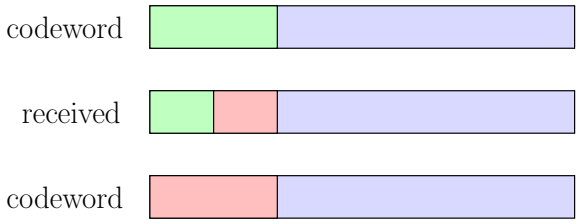

Figure 2.1: The unique decoding radius: For two strings at distance d, the adversary can use $d/2$ errors to turn them into the same received word, so the receiver will not be able to determine which was originally sent.

2.2 List-decoding

The model of **list-decoding** is a relaxation of unique decoding in which the decoder is allowed to output a *list* of candidate messages, one of which must be the correct message. By weakening the decoding requirement, we are able to correct a larger fraction of errors. In fact, although unique decoding cannot correct more than a $1/2$ fraction of errors, with list-decoding we will be able to correct an error fraction approaching 1.

More formally, we have the following definition.

Definition 2.11. A code $C \subseteq \Sigma^n$ is (p, L) **list-decodable** if for all $y \in \mathbb{F}_q^n$, $|\{c \in C \mid d_H(c, y) \leq pn\}| \leq L$.

If this holds, we call L the **list size** of the code for error fraction p.

This says that there are only L codewords within distance pn of any possible received word. In other words, although the original message might not be uniquely identifiable, we can still narrow the possibilities down to a list of size L. We will often refer to a code as being list-decodable from some fraction p of errors; in this case, the list size will be understood to be bounded by some polynomial in the block length.

It turns out that under the Hamming metric, *all* codes are list-decodable from an error fraction which is larger than half their minimum distance, a fact which is captured by the Johnson bound.

Theorem 2.12 (Johnson bound)**.** Let Σ have size q, and let $C \subseteq \Sigma^n$ have relative distance δ. Then C is $\big(J_q(\delta), O(n)\big)$ list-decodable, where

$$J_q(\delta) = \left(1 - \sqrt{1 - \frac{q}{q-1}\delta}\right)\left(1 - \frac{1}{q}\right).$$

In other words, up to the "Johnson radius" $J_q(\delta)$, although unique decoding will not always be possible, there will only be a linear number of possible messages, compared to the exponential number of messages in any positive-rate code.

The list-decodable codes in this thesis will be over alphabets which grow with block length, so we also record an "alphabet-oblivious" version of the Johnson bound below.

Theorem 2.13 (Johnson bound again)**.** Let $C \subseteq \Sigma^n$ have relative distance δ. Then C is $\big(J(n,\delta), O(n|\Sigma|)\big)$ list-decodable, where

$$J(n,\delta) = 1 - \sqrt{1-\delta}.$$

Notice that although this result shows that the list of nearby codewords will be short, it does not give an efficient way to actually compute the list, and the fraction of errors for which it guarantees decoding is not necessarily tight for a given code. It turns out that there are certain families of codes for which we can decode from a much higher fraction of errors. Indeed, this holds for *most* codes. As before, $h_q(\cdot)$ is the q-ary entropy function.

Theorem 2.14. Let $0 < p < 1 - 1/q$, where $|\Sigma| = q$. Then with high probability, a random code $C \subseteq \Sigma^n$ of rate $R = 1 - h_q(p) - 1/L$ is list-decodable from a p error fraction with list size L.

On the other hand, for $\gamma > 0$, any code $C \subseteq \Sigma^n$ of rate $R = 1 - h_q(p) + \gamma$ is *not* list-decodable from a p fraction of errors with polynomial list size.

For large alphabets $q \approx 2^{1/\varepsilon}$, the case we are interested in, the achievable rate promised by this theorem for error rate p is at least $1 - p - \varepsilon$, the so-called *list-decoding capacity*. In the next section, and in Chapter 3, we will see two efficient constructions of codes which achieve list-decoding capacity.

2.3 A few code constructions

We have seen already that choosing a code at random is often enough to achieve the trade-offs we are looking for. However, the primary drawback of these codes is that they are unwieldy objects; if we choose a random code of constant rate, it will have exponential size and we may not be able to perform encoding and decoding efficiently. We now give a few constructions of codes which can be handled efficiently.

2.3.1 Reed-Solomon codes

Reed-Solomon codes are the first of a series of codes we will see which are based on polynomial evaluation. These codes are defined over some finite field $\mathbb{F}_q$, and encode messages by identifying vectors $(f_0, f_1, \ldots, f_{k-1}) \in (\mathbb{F}_q)^k$ with the associated polynomial $f(X) = \sum_{i=0}^{k-1} f_i X^i$ over $\mathbb{F}_q$.

Definition 2.15. Let $a_1, a_2, \ldots, a_n \in \mathbb{F}_q$ be distinct, and let $k \leq n$ be the *degree parameter*.

The *Reed-Solomon code* $\mathrm{RS}[n, k]$ is a code over $\mathbb{F}_q$ which encodes a polynomial $f \in \mathbb{F}_q[X]$ of degree at most $k-1$ by[1]

$$f(X) \mapsto \big(f(a_1), f(a_2), \ldots, f(a_n)\big).$$

This is a code of block length n and rate k/n.

The key fact that makes Reed-Solomon codes useful for error correction is that two polynomials of degree $k-1$ over a field $\mathbb{F}_q$ can have the same evaluations on at most $k-1$ points. This means that any k evaluations can be used to determine the polynomial (for example, two points determine a line). Not only does this give $\mathrm{RS}[n, k]$ a distance of at least $n - (k-1)$, meeting the Singleton bound, but as each coordinate is (in some sense) equally helpful, these codes are well suited for handling worst-case errors.

The following algorithm, due to Welch and Berlekamp, gives an efficient way to correct Reed-Solomon codes from any number of errors up to half the distance. As this will form the starting point of our later algorithms, we present it here.

Proposition 2.16. The Reed-Solomon code $\mathrm{RS}[n, k]$ can be uniquely decoded in polynomial time from up to $\lfloor \frac{n-k}{2} \rfloor$ errors.

Proof. Suppose that we have encoded a polynomial $f(X) = \sum_{i=0}^{k-1} f_i X^i$, and received a vector $(y_1, \ldots, y_n)$ such that $y_i \neq f(a_i)$ for at most $D := \lfloor \frac{n-k}{2} \rfloor$ values of i.

The algorithm proceeds in two steps: first, we use the erroneous received word to find a condition that any nearby codeword must satisfy. Then we solve the condition to find the original codeword.

Step one (Interpolation): We interpolate a nonzero, bivariate polynomial $Q(X, Y)$ of the form $Q(X, Y) = A_0(X) + A_1(X)Y$. We will require the following:

[1] To be more precise, the code depends on the choice of the evaluation points. As this does not affect our results, here and in later code definitions, we will suppress the dependence on the a_i's.

- A_0 and A_1 are univariate polynomials with $\deg(A_0) \leq D+k-1$, and $\deg(A_1) \leq D$.
- $Q(a_i, y_i) = 0$ for all $1 \leq i \leq n$.

The polynomial Q can be found in polynomial time by solving a homogeneous linear system in $2D+k+1$ variables with n constraints. We know that a nonzero solution exists; it is easy to check that $A_1(X) = \prod_{y_i \neq f(a_i)}(X - a_i)$ and $A_0(X) = A_1(X)f(X)$ satisfies the constraints.

The key observation is the following.

Lemma 2.17. If f is a polynomial of degree at most $k-1$ satisfying $y_i \neq f(a_i)$ for at most D values of i, then $Q\big(X, f(X)\big) = 0$.

Proof. The degree of $Q\big(X, f(X)\big)$ is at most $D+k-1$. By our interpolation conditions, for each correctly received coordinate i with $y_i = f(a_i)$, we have $Q\big(a_i, f(a_i)\big) = 0$. As this holds for at least $n - D \geq D + k$ values of i, $Q(\big(X, f(X)\big)$ must be the zero polynomial. □

Step two (Root finding): By the above claim, we have $A_0(X) + A_1(X)f(X) = 0$. Thus, as long as fewer than D errors have occurred, we can uniquely decode the original message as $f(X) = -A_0(X)/A_1(X)$. (This is well-defined as $Q(X, Y)$ was nonzero.) □

This theorem shows that we can uniquely decode Reed-Solomon codes from up to the optimal error rate of half the code distance. In fact, it turns out that we can *efficiently* list-decode Reed-Solomon codes up to the Johnson bound (see [GS99]). List-decoding beyond the Johnson bound, however, has been a challenging open problem. It was shown in [BSKR10] that for Reed-Solomon codes where the evaluation points are the whole field $\mathbb{F}_q$, then one cannot hope to do much better than the Johnson bound. On the other hand, the authors of [RW14] show that this is (in some sense) a pathological example, and *most* random choices of the set of evaluation points yields a code which is combinatorially list-decodable beyond the Johnson bound; indeed, nearly achieving list-decoding capacity.

Unfortunately, the results of [RW14] do not give an efficient way to find a good choice of evaluation points, nor to perform the actual list-decoding. For efficient list-decoding up to capacity, we turn to folded Reed-Solomon codes.

2.3.2 Folded Reed-Solomon codes

Folded Reed-Solomon codes, introduced in [GR08a], provided the first explicit family of rate-R codes which could be list-decoded from a $1-R-\varepsilon$ error fraction for any R, ε. (We

will see an alternative construction in Chapter 3.)

We saw in the Reed-Solomon unique decoding algorithm that every correctly received coordinate $f(a_i)$ gave us information about the function $Q\big(X, f(X)\big)$; namely, that it has a root at a_i. The idea behind folding is to make each coordinate not a single evaluation, but a "bundle" (tuple) of related evaluations. This way, instead of receiving an arbitrary set of correct evaluations, we get evaluations with some algebraic structure. This will allow us to find *multiple* roots for every correctly received coordinate.

Definition 2.18 (m-folded Reed-Solomon code)**.** Let $\gamma \in \mathbb{F}_q$ be a primitive element of $\mathbb{F}_q$. Let $n \leq q-1$ be a multiple of m, and let $1 \leq k < n$ be the *degree parameter*.

The *folded Reed-Solomon (FRS) code* $\mathrm{FRS}_q^{(m)}[n,k]$ is a code over alphabet $\mathbb{F}_q^m$ which encodes a polynomial $f \in \mathbb{F}_q[X]$ of degree at most $k-1$ as

$$\left(\begin{bmatrix} f(1) \\ f(\gamma) \\ \vdots \\ f(\gamma^{m-1}) \end{bmatrix}, \begin{bmatrix} f(\gamma^m) \\ f(\gamma^{m+1}) \\ \vdots \\ f(\gamma^{2m-1}) \end{bmatrix}, \ldots, \begin{bmatrix} f(\gamma^{n-m}) \\ f(\gamma^{n-m+1}) \\ \vdots \\ f(\gamma^{n-1}) \end{bmatrix} \right) . \tag{2.1}$$

The block length of $\mathrm{FRS}_q^{(m)}[n,k]$ is $N = n/m$, and its rate is $R = k/n$.

As this definition shows, a correctly received coordinate now does not merely contain some evaluation $f(a_i)$, but $f(a_i)$ together with $f(\gamma a_i)$, $f(\gamma^2 a_i)$, and so on. This allows for a more involved interpolation step in the decoding, allowing us to reach list-decoding capacity. This result is summarized in the following theorem from [GR08a].

Theorem 2.19 ([GR08a])**.** For every $\varepsilon > 0$ and $0 < R < 1$, there is a family of folded Reed-Solomon codes which have rate at least R and which can be list-decoded up to a fraction $1 - R - \varepsilon$ of errors in time $(N/\varepsilon^2)^{O(\log(1/R)/\varepsilon)}$, where N is the block length of the code.

More specifically, for rate R and an integer parameter $s \leq m$, they show list-decodability from an error fraction of

$$1 - (1+\delta)\left(\frac{mR}{m-s+1}\right)^{s/(s+1)} \tag{2.2}$$

in time $\big(O_\delta(q)\big)^{O(s)}$. By picking $\delta \approx \varepsilon$, $s \approx 1/\varepsilon$ and $m \approx 1/\varepsilon^2$, the above quantity is at least $1 - R - \varepsilon$, and the decoding complexity and list size are $\approx q^{O(1/\varepsilon)}$.

We now sketch a variant of their decoding algorithm due to Vadhan, which can be found in his monograph [Vad11, Chap. 5]. This algorithm will correct a smaller fraction of errors, but it is simpler to describe, and is still sufficient to reach list-decoding capacity. This algorithm will form the basis for our algorithm in Chapter 3.

As in the Reed-Solomon case, the decoding algorithm has two steps, interpolation and root-finding. The details of the interpolation step can be found in Chapter 3.

Write the received word as an $m \times N$ matrix over $\mathbb{F}_q$:

$$\begin{pmatrix} y_0 & y_m & \cdots & y_{n-m+1} \\ y_1 & y_{m+1} & \cdots & y_{n-m+2} \\ y_2 & y_{m+2} & \cdots & y_{n-m+3} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m-1} & y_{2m-1} & \cdots & y_{n-1} \end{pmatrix}$$

Step one (Interpolation): We would like to interpolate a nonzero, $(s+1)$-variate polynomial $Q(X, Y_1, Y_2, \ldots, Y_s)$ of the form

$$Q(X, Y_1, Y_2, \ldots, Y_s) = A_0(X) + A_1(X)Y_1 + A_2(X)Y_2 + \cdots + A_s(X)Y_s.$$

We will require the following:

- $\deg(A_i) \leq D$ for $i = 1, 2, \ldots, s$ and $\deg(A_0) \leq D + k - 1$ for some degree parameter D, and
- $Q(\gamma^{im+j}, y_{im+j}, y_{im+j+1}, \cdots, y_{im+j+s-1}) = 0$ for $i = 0, 1, \ldots, N-1$ and $j = 0, 1, \ldots, m-s$.

The degree parameter can be chosen such that a nonzero Q exists, and we have the following analogue of Lemma 2.17, whose proof appears in Chapter 3.

Lemma 2.20. If $f \in \mathbb{F}[X]$ is a polynomial of degree at most $k-1$ whose FRS encoding (2.1) agrees with the received word $\mathbf{y}$ in at least t columns for $t > \frac{D+k-1}{m-s+1}$, then

$$Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X)) = 0\,. \tag{2.3}$$

Step two (Root finding): To solve the functional equation 2.3, we observe that $f(\gamma X) = f(X)^q \pmod{E(X)}$, where $E(X) = X^{q-1} - \gamma$. Thus $f \pmod{E}(X)$ can be found by finding the roots of the univariate polynomial

$$T(Y) = Q(X, Y, Y^q, \ldots, Y^{q^{s-1}}) \mod E(X)$$

with coefficients from $L = \mathbb{F}_q[X]/(E(X))$. The polynomial $E(X)$ is irreducible over $\mathbb{F}_q$ and therefore L is an *extension field*. The parameter choices ensure that $T \neq 0$, and thus T cannot have more than q^{s-1} roots, and these roots may all be found in polynomial time.

Remark. It is easy to see that the algorithm given above can be implemented in polynomial time, so we have constructed list-decodable codes which achieve the optimal rate-distance trade-off efficiently. However, a comparison with our existential list-decoding results shows that the list size of $q^{O(1/\varepsilon)}$ is larger than the best possible. It is unknown whether we can prove a tighter list size bound for FRS codes themselves, but in Chapter 3 we will show how to modify our codes to improve the list size.

2.4 Deletion codes

In the deletion model, we transmit a string $c \in \Sigma^n$, and rather than receiving another string $r \in \Sigma^n$ which differs in some number of coordinates, we receive a string $r' \in \Sigma^t$, where $t \leq n$, which is a *substring* of c. In other words, some of the coordinates of c have been *deleted*. (This stands in contrast to the well-studied model of erasures, where the affected symbol is not removed, but replaced with a "blank" placeholder.)

Recovering from deletions is particularly challenging because we have not only lost the information in the deleted coordinates, but we have lost the positional information in the remaining coordinates. To see why this might be tricky, recall that the Reed-Solomon decoding algorithm presented earlier depended on knowing which evaluation point each received coordinate corresponded to.

There is an easy way around that particular difficulty; simply augment the alphabet of the code so that each coordinate consists of an ordered pair, representing $\big(a, f(a)\big)$. However, this requires the alphabet size to grow with the block length. In this thesis, we will be focusing on understanding deletion codes when the alphabet size is a *fixed constant*, and we expect a constant fraction of (adversarial) deletions. For example, if deletions happen at the bit level (alphabet size 2), what fraction of deletions could we hope to correct with positive rate? Interestingly, we don't know the answer.

Definition 2.21. Let $C \subseteq \Sigma^n$ be a code. We say that C is *correctable from t deletions* if the longest common subsequence (LCS) between any two distinct codewords in C has length less than $n - t$.

As in the standard Hamming distance model, we would like to understand the achievable trade-off between the rate of a code and its correctable deletion fraction. More ambitiously, we would also like to construct codes for which encoding and decoding can

be performed efficiently. Our results in these directions are given in Chapter 4; we now outline some of the previous work on deletions.

2.4.1 Previous work on deletions

The problem of communicating over the *binary deletion channel*, in which each transmitted bit is deleted independently with a fixed probability p, has been a subject of much study (see the excellent survey by Mitzenmacher [Mit09] for more background and references). However, even this easier case is not well-understood. In particular, the capacity of the binary deletion channel remains open, although it is known to approach $1 - h(p)$ as p goes to 0, where $h(p)$ is the binary entropy function (see [DG06, Gal61, Zig69] for lower bounds and [KMS10, KM13] for upper bounds), and it is known to be positive (at least $(1-p)/9$) [MD06]) even as $p \to 1$.

The more difficult problem of correcting from adversarial rather than random deletions has also been studied, but with a focus on correcting a constant *number* (rather than fraction) of deletions [Lev66]. Codes that can correct a single deletion have received a fair bit of attention (see the survey [Slo02]), but it turns out that even correcting two deletions poses significant challenges and is not well understood, with efficient codes with low redundancy discovered only very recently [BGZ15].

Coding for a constant *fraction* of adversarial deletions, which is the focus of the results in this thesis, has been considered previously by Schulman and Zuckerman in [SZ99]. They construct constant-rate binary codes which are efficiently decodable from a small constant fraction of worst-case deletions and insertions, and can also handle a small fraction of transpositions. The rate of these codes are bounded away from 1, whereas existentially one can hope to achieve a rate approaching 1 for a small deletion fraction. In the following sections, we will sketch their construction in the deletion case.

2.4.2 Concatenated codes

As we mentioned, it is fairly easy to give codes over growing alphabets which can correct a constant fraction of deletions, with a simple modification to Reed-Solomon codes. Concatenation is a simple but powerful tool which allows us to convert a code over a large alphabet into a "concatenated code" over a smaller alphabet, while preserving most of its rate and distance.

Definition 2.22. If $C_{\text{out}} \subseteq \Sigma_{\text{out}}^n$ is an "outer" code, and $C_{\text{in}} \subseteq \Sigma_{\text{in}}^m$ is an "inner" code with encoding function $\text{Enc} : \Sigma_{\text{out}} \to \Sigma_{\text{in}}^m$, the **concatenated** code $C_{\text{out}} \circ C_{\text{in}} \subseteq \Sigma_{\text{in}}^{nm}$ is a code

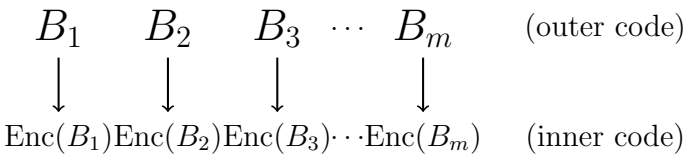

Figure 2.2: A concatenated codeword: We begin with a codeword $B_1 B_2 \cdots B_m \in C_{\text{out}}$, and map each symbol B_i to its inner encoding $\mathrm{Enc}(B_i)$.

whose codewords are obtained by applying the inner encoder Enc to each symbol of a fixed codeword in C_{out}, then concatenating the results.

The encoding of a concatenated codeword is illustrated in Figure 2.4.2.

The following simple observation shows that if both the outer and inner codes have good rate, then so does their concatenation.

Lemma 2.23. The rate of a concatenated code $C_{\text{out}} \circ C_{\text{in}}$ is $R(C_{\text{out}}) \cdot R(C_{\text{in}})$.

In the Hamming case, with symbol errors, we also have the following bound.

Lemma 2.24. The Hamming distance δ of the concatenated code $C_{\text{out}} \circ C_{\text{in}}$ satisfies

$$\delta \geq \delta(C_{\text{out}}) \cdot \delta(C_{\text{in}}).$$

Proof. Any two outer codewords c_1 and c_2 differ on a $\delta(C_{\text{out}})$ fraction of coordinates. For each of these coordinates, the inner encoding differs on a $\delta(C_{\text{in}})$ fraction of coordinates. □

Concatenated codes come with a natural decoding algorithm from less than $\delta(C_{\text{out}}) \cdot \delta(C_{\text{in}})/4$ symbol errors — attempt to decode each inner codeword, then use the successfully decoded coordinates to decode the outer codeword. If there are fewer than $\delta(C_{\text{out}}) \cdot \delta(C_{\text{in}})/4$ erroneous coordinates, then fewer than a $\delta(C_{\text{out}})/2$ fraction of inner codewords can have $\delta(C_{\text{in}})/2$ errors, so a $1 - \delta(C_{\text{out}})/2$ fraction of outer codeword symbols are correctly decoded, and the outer decoder will succeed. (It is possible to improve the decoding radius to $\delta(C_{\text{out}}) \cdot \delta(C_{\text{in}})/2$ efficiently, but this idea will be sufficient for our purposes.)

The problem with trying to apply this strategy directly to deletion codes is that we don't know which symbols belong to which inner codeword. In particular, once deletions have occurred, we don't know how long the first inner codeword is. It could have no deletions, or be completely deleted, or anywhere in between, and as we have stated the construction, there is no way to know. Trying every possible codeword length is quite costly, and so using these codes for deletions requires additional tricks to try to distinguish different inner codewords from each other.

One more remark on concatenation: An extremely useful property of these concatenated codes is that the block length of the inner code is quite small, roughly $\log|\Sigma_{\text{out}}|$. This means that we do not necessarily need the inner code to be efficient; even algorithms which are *exponential* in the block length will run in polynomial time.

In other words, concatenation allows us to combine a good, efficient code over a large alphabet with a good, not necessarily efficient code over a small alphabet, and get out a good, efficient code over a small alphabet — the best of both worlds. This observation will be useful both in the construction of the next section, and in our constructions in Chapter 4.

2.4.3 A code construction

We now give a high-level view of the concatenated construction of good binary deletion codes, due to Schulman and Zuckerman ([SZ99]). Our construction of high-rate binary deletion codes in Chapter 4 is based on a refinement of this construction which keeps the rate of the code high. The code constructed in [SZ99] is actually proven to correct from a certain number of insertions and transpositions as well, but we will only discuss deletions.

Theorem 2.25 ([SZ99])**.** There exists a binary explicit code of positive rate which can correct a constant fraction of deletions. Moreover, this code can be encoded and decoded in polynomial time.

The outer code for the concatenation will be a so-called Indexed Reed-Solomon code, where we fix a set of distinct evaluation points $a_1, \ldots, a_n$ in $\mathbb{F}_q$, and encode a polynomial f by the string in $\big(\{1, \ldots, n\} \times \mathbb{F}_q\big)^n$ whose ith coordinate is $\big(i, f(i)\big)$. The addition of the index decreases the rate, but we can still take the rate to be constant. This code can correct from a constant fraction of errors and deletions.

The inner code will be a greedily constructed code (similar to the way we proved the Gilbert-Varshamov bound earlier). They show that the greedy algorithm can construct a binary code of constant rate and constant deletion distance.

Concatenating these two codes gives a code of constant rate. In order to help distinguish the inner codewords, they insert a 1 after every symbol of the inner code, then insert a "buffer" of 0's whose length is comparable to (big-O of) the block length of the inner code. This decreases the rate by only a constant factor.

The decoding then proceeds by looking for long runs of 0's and marking those as buffers. The symbols between the buffers are decoded using the inner decoder, and the results are passed to the outer decoder.

The analysis proceeds by observing that if the allowed deletion fraction is sufficiently small, then *most* buffers will survive and be marked by the decoder, allowing the decoder to locate many blocks correctly. As the inner code can handle a constant fraction of deletions, enough of the correctly located blocks will survive to be correctly decoded, and the outer decoder will succeed.

Chapter 3

Linear-algebraic list-decoding

In which we revisit folded Reed-Solomon codes • Linear algebra comes in handy • Folding with derivatives works • Subspaces are easy to avoid

In Chapter 2, we saw a "Welch-Berlekamp style" list-decoding algorithm for folded Reed-Solomon codes which interpolates a polynomial $Q(X, Y_1, Y_2, \ldots, Y_s)$, then solves the functional equation $Q\big(X, f(X), f(\gamma X), f(\gamma^2 X), \ldots, f(\gamma^{s-1} X)\big) = 0$ to obtain a list of possible message polynomials f.

In this chapter, we make a simple observation about this list (namely, that it's contained in an small affine subspace), and use this to give a streamlined decoding algorithm. We also define a new family of codes, known as *derivative codes*, which can be list-decoded in the same framework. Finally, we explore how the structure of the list can be used to construct subcodes with an improved list size guarantee.

The results in this chapter, based on joint work with Guruswami, appeared in [GW13].

3.1 List-decoding folded Reed-Solomon codes

Recall (Definition 2.18) that for a fixed γ generating $\mathbb{F}_q$, the folded Reed-Solomon (FRS) code $\mathrm{FRS}_q^{(m)}[n, k]$ encodes a polynomial f of degree $k - 1$ as

$$\left(\begin{bmatrix} f(1) \\ f(\gamma) \\ \vdots \\ f(\gamma^{m-1}) \end{bmatrix}, \begin{bmatrix} f(\gamma^m) \\ f(\gamma^{m+1}) \\ \vdots \\ f(\gamma^{2m-1}) \end{bmatrix}, \ldots, \begin{bmatrix} f(\gamma^{n-m}) \\ f(\gamma^{n-m+1}) \\ \vdots \\ f(\gamma^{n-1}) \end{bmatrix} \right) .$$

Suppose a codeword of the m-folded RS code was transmitted and we received a string $\mathbf{y} \in (\mathbb{F}_q^m)^N$ which we view as an $m \times N$ matrix over $\mathbb{F}_q$ (recall $N = n/m$):

$$\begin{pmatrix} y_0 & y_m & \cdots & y_{n-m+1} \\ y_1 & y_{m+1} & \cdots & y_{n-m+2} \\ y_2 & y_{m+2} & \cdots & y_{n-m+3} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m-1} & y_{2m-1} & \cdots & y_{n-1} \end{pmatrix} \tag{3.1}$$

We would like to recover a list of all polynomials $f \in \mathbb{F}_q[X]$ of degree $k-1$ whose FRS encoding (2.1) agrees with $\mathbf{y}$ in at least $N - e$ columns, for some error bound e. Note that an agreement in some column means that all m values in that column match.

3.1.1 A Welch-Berlekamp style interpolation

Given a received word as in (3.1) we will interpolate a nonzero polynomial of the form

$$Q(X, Y_1, Y_2, \ldots, Y_s) = A_0(X) + A_1(X)Y_1 + A_2(X)Y_2 + \cdots + A_s(X)Y_s \tag{3.2}$$

over $\mathbb{F}_q$ with the degree restrictions $\deg(A_i) \leq D$ for $i = 1, 2, \ldots, s$ and $\deg(A_0) \leq D + k - 1$, where the degree parameter D is chosen to be

$$D = \left\lfloor \frac{N \cdot (m - s + 1) - k + 1}{s + 1} \right\rfloor . \tag{3.3}$$

The number of monomials in a polynomial Q with these degree restrictions equals

$$(D+1)s + D + k = (D+1)(s+1) + k - 1 > N(m - s + 1) \tag{3.4}$$

for the above choice (3.3) of D. The interpolation requirements on $Q \in \mathbb{F}_q[X, Y_1, \ldots, Y_s]$ are the following:

$$Q(\gamma^{im+j}, y_{im+j}, y_{im+j+1}, \cdots, y_{im+j+s-1}) = 0 \quad \text{for } i = 0, 1, \ldots, N-1,\ j = 0, 1, \ldots, m - s\,. \tag{3.5}$$

(Again, $N = n/m$.) We then have the following.

Lemma 3.1. Let

$$D = \left\lfloor \frac{N \cdot (m - s + 1) - k + 1}{s + 1} \right\rfloor . \tag{3.6}$$

A nonzero $Q \in \mathbb{F}_q[X, Y_1, \ldots, Y_s]$ of the form (3.2) satisfying the interpolation conditions (3.5) with $\deg(A_0) \leq D + k - 1$ and $\deg(A_j) \leq D$ for $1 \leq j \leq s$ exists and can be found by solving a homogeneous linear system over $\mathbb{F}_q$ with at most n constraints and variables. Further, this interpolation can be performed in $O(n \log^2 n \log \log n)$ operations over $\mathbb{F}_q$.

Proof. This holds since the number of interpolation conditions $N \cdot (m - s + 1)$ is less than the number of degrees of freedom (monomials) in Q. Regarding the claimed runtime, even though the best known algorithms for solving a general $n \times n$ linear system take time n^ω where ω is the exponent of matrix multiplication (currently $\approx 2.37\ldots$), the above linear system has a special structure (involving evaluations at powers of γ). This can be exploited to solve the system in near-linear runtime as shown in [Bra10] (see Proposition 5.11 in Chapter 5). □

The following lemma shows that any such polynomial Q gives an algebraic condition that the message polynomials $f(X)$ we are interested in list decoding must satisfy.

Lemma 3.2. Let Q satisfy the conclusion of Lemma 3.1. If $f \in \mathbb{F}[X]$ is a polynomial of degree at most $k-1$ whose FRS encoding (2.1) agrees with the received word $\mathbf{y}$ in at least t columns for $t > \frac{D+k-1}{m-s+1}$, then

$$Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1} X)) = 0 . \tag{3.7}$$

Proof. Define $\Lambda(X)$ to be the polynomial $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1} X))$. Due to the degree restrictions on Q, the degree of $\Lambda(X)$ is easily seen to be at most $D + k - 1$. If the FRS encoding of f agrees with $\mathbf{y}$ in the i'th column (for some $i \in \{0, 1, \ldots, N-1\}$), we have

$$f(\gamma^{im}) = y_{im}, \quad f(\gamma^{im+1}) = y_{im+1}, \quad \ldots \quad , f(\gamma^{im+m-1}) = y_{im+m-1} .$$

Together with the interpolation conditions (3.5), this implies $\Lambda(\gamma^{im+j}) = 0$ for $j = 0, 1, \ldots, m - s$. In other words, Λ has at least $m - s + 1$ distinct roots for each such column i. Thus Λ must have at least $t(m - s + 1)$ roots in all. Since $\deg(\Lambda) \leq D + k - 1$, if $t > (D + k - 1)/(m - s + 1)$, we must have $\Lambda = 0$. □

For the choice of D in (3.3), the requirement on t in Lemma 3.2 is met if $t \cdot (m-s+1) > \frac{N\cdot(m-s+1)+s(k-1)}{s+1}$, and hence if the fractional agreement t/N satisfies

$$\begin{aligned}\frac{t}{N} &\geq \frac{1}{s+1} + \frac{s}{s+1}\frac{k}{N(m-s+1)} \\ &= \frac{1}{s+1} + \frac{s}{s+1}\frac{mR}{m-s+1}\,. \end{aligned} \tag{3.8}$$

In other words, the fractional agreement needed is $\frac{1}{s+1} + \frac{s}{s+1}\frac{mR}{m-s+1}$. (Recall that $R = k/n$ is the rate of the code.) Note that by the AM-GM inequality, this agreement is always higher than the agreement fraction $\left(\frac{mR}{m-s+1}\right)^{s/(s+1)}$ needed in [GR08a].[1] Thus this variant corrects a smaller fraction of errors, although the fraction of errors corrected can still exceed $1-R-\varepsilon$, with the choice $s \approx 1/\varepsilon$ and $m \approx 1/\varepsilon^2$. Further, as we see next, it offers some advantages when it comes to retrieving the solutions f to (3.7).

3.1.2 Retrieving candidate polynomials f

By the preceding section, to complete the list decoding we need to find all polynomials $f \in \mathbb{F}_q[X]$ of degree at most $k-1$ that satisfy

$$A_0(X) + A_1(X)f(X) + A_2(X)f(\gamma X) + \cdots + A_s(X)f(\gamma^{s-1}X) = 0\,. \tag{3.9}$$

We note the following simple but very useful fact:

Observation. The above forms a system of linear equations over $\mathbb{F}_q$ in the coefficients $f_0, f_1, \cdots, f_{k-1}$ of the polynomial $f(X) = f_0 + f_1 X + \cdots + f_{k-1}X^{k-1}$. Thus, the set of solutions $(f_0, f_1, \ldots, f_{k-1})$ of (3.9) form an affine subspace of $\mathbb{F}_q^k$.

In particular, the above immediately gives an efficient algorithm to find a compact representation of all the solutions to (3.9) — simply solve the linear system! This simple observation is the starting point driving our analysis.

We next prove that when γ is primitive, the space of solutions has dimension at most $s-1$. Note that we already knew this by the earlier argument in Chapter 2 over the extension field $\mathbb{F}_q[X]/(X^{q-1}-\gamma)$. But it is instructive to give a direct proof of this working only over $\mathbb{F}_q$. The proof in fact works when the order of γ is at least k. Further, it exposes

[1] Recall that for Reed-Solomon codes ($m = 1$) this was also exactly the case: the classical algorithms unique decoded the codeword when the agreement fraction was at least $\frac{1+R}{2}$, and the list decoding algorithm in [GS99] list decoded from agreement fraction $\sqrt{R}$.

the simple structure of the linear system, which can be used to find a basis for the solutions in quadratic time.

Lemma 3.3. If the order of γ is at least k (in particular when γ is primitive), the affine space of solutions to (3.9) has dimension d at most $s-1$.

Further, one can compute using $O\big(sk^2\big)$ field operations over $\mathbb{F}_q$ a lower-triangular matrix $H \in \mathbb{F}_q^{k\times k}$ of rank at least $k-s+1$ and a vector $\mathbf{z} \in \mathbb{F}_q^k$, such that the coefficient (column) vectors $\mathbf{f} = (f_0, f_1, \ldots, f_{k-1})^T$ of solutions to (3.9) are contained in the affine space $H\mathbf{f} = \mathbf{z}$.

Proof. First, by factoring out the common powers of X which divide each of the polynomials $A_0(X), A_1(X), \ldots, A_s(X)$, we can assume that at least one $A_{i^*}(X)$ for some $i^* \in \{0, 1, \ldots, s\}$ is not divisible by X, and has nonzero constant term. Furthermore, if $A_1(X), \ldots, A_s(X)$ are all divisible by X, then so is $A_0(X)$, so we can take $i^* > 0$.

Let us denote $A_i(X) = \sum_{\ell=0}^{D+k-1} a_{i,\ell} X^\ell$ for $0 \le i \le s$. (We know that the degree of $A_i(X)$ for $i \ge 1$ is at most D, so $a_{i,\ell} = 0$ when $i \ge 1$ and $\ell > D$, but for notational ease let us introduce these coefficients.) For $j = 0, 1, 2, \ldots, k-1$, define the polynomial

$$B_j(X) = a_{1,j} + a_{2,j}X + a_{3,j}X^2 + \cdots + a_{s,j}X^{s-1}\,. \tag{3.10}$$

We know that $a_{i^*,0} \neq 0$, and therefore $B_0 \neq 0$.

By Condition (3.9), for each $r = 0, 1, 2, \ldots$, the coefficient of X^r in the polynomial

$$\Lambda(X) = A_0(X) + A_1(X)f(X) + A_2(X)f(\gamma X) + \cdots + A_s(X)f(\gamma^{s-1}X)$$

equals 0.

The constant term of $\Lambda(X)$ equals $a_{0,0} + a_{1,0}f_0 + a_{2,0}f_0 + \cdots + a_{s,0}f_0 = a_{0,0} + B(1)f_0$. Thus if $B(1) \neq 0$, then f_0 is uniquely determined as $-a_{0,0}/B(1)$. If $B(1) = 0$, then $a_{0,0} = 0$ or else there will be no solutions to (3.9) and in that case f_0 can take an arbitrary value in $\mathbb{F}_q$.

The coefficient of X^r of $\Lambda(X)$, for $0 \le r < k$, equals

$$\begin{aligned}
&a_{0,r} + f_r \cdot (a_{1,0} + a_{2,0}\gamma^r + \cdots + a_{s,0}\gamma^{(s-1)r}) \\
&\quad + f_{r-1} \cdot (a_{1,1} + a_{2,1}\gamma^{r-1} + \cdots + a_{s,1}\gamma^{(s-1)(r-1)}) \\
&\quad + \cdots + f_1 \cdot (a_{1,r-1} + a_{2,r-1}\gamma + \cdots + a_{s,r-1}\gamma^{s-1}) \\
&\quad + f_0 \cdot (a_{1,r} + \cdots + a_{s,r}) \\
&\quad = B_0(\gamma^r)f_r + \Big(\sum\nolimits_{j=1}^{r} B_j(\gamma^{r-j})f_{r-j}\Big) + a_{0,r},
\end{aligned} \tag{3.11}$$

recalling the definition (3.10) of the polynomials B_j. The linear form (3.11) must thus equal 0. The key point is that if $B_0(\gamma^r) \neq 0$, then this implies that f_r is an affine combination of $f_0, f_1, \ldots, f_{r-1}$ and in particular is uniquely determined given values of $f_0, f_1, \ldots, f_{r-1}$.

Thus the dimension of the space of solutions is at most the number of r, $0 \leq r < k$, for which $B_0(\gamma^r) = 0$. Since γ has order at least k, the powers γ^r for $0 \leq r < k$ are all distinct. Also we know that B_0 is a *nonzero* polynomial of degree at most $s-1$. Thus $B_0(\gamma^r) = 0$ for at most $s-1$ values of r.

We have thus proved that the solution space has dimension at most $s-1$. To justify the claim about the decoding complexity and structure of solution space, note that the linear system satisfied by candidate solutions $\mathbf{f} = (f_0, f_1, \ldots, f_{k-1})^T$ is $H\mathbf{f} = \mathbf{z}$, where $\mathbf{z} = (-a_{0,0}, -a_{0,1}, \ldots, -a_{0,k-1})^T$ and the (r,j)-th entry of H equals $B_{r-j}(\gamma^j)$ for $j \leq r$ and 0 otherwise. The computation of H amounts to evaluating the polynomials B_j, $0 \leq j < k$, each of which has degree less than s, at the points $\{1, \gamma, \ldots, \gamma^{k-1}\}$. This can be accomplished in $O(sk^2)$ operations over $\mathbb{F}_q$. □

Combining Lemmas 3.1 and 3.3 and the decoding bound (3.8), we can conclude the following.

Theorem 3.4. For the folded Reed-Solomon code $\mathrm{FRS}_q^{(m)}[n,k]$ of block length $N = n/m$ and rate $R = k/n$, the following holds for all integers s, $1 \leq s \leq m$. Given a received word $\mathbf{y} \in (\mathbb{F}_q^m)^N$, using $O(n^2 + sk^2)$ operations over $\mathbb{F}_q$, one can find a subspace of dimension at most $s-1$ that contains all message polynomials $f \in \mathbb{F}_q[X]$ of degree less than k whose FRS encoding (2.1) differs from $\mathbf{y}$ in at most a fraction

$$\frac{s}{s+1}\left(1 - \frac{mR}{m-s+1}\right)$$

of the N codeword positions.

Remark. When $s = m = 1$, the above just reduces to a unique decoding algorithm up to a fraction $(1-R)/2$ of errors.

Choosing $s \approx 1/\varepsilon$ and $m \approx 1/\varepsilon^2$ suffices to ensure decoding from a $1 - R - \varepsilon$ fraction of errors, summarized in the following theorem. Note that the decoding guarantee of Theorem 3.4 improves with s and as m increases relative to s. However, as s increases, so does our worst-case list size guarantee of q^{s-1}. For fixed parameters k and n, as m increases, the absolute number of errors which can be corrected decreases.

Theorem 3.5. For every $\varepsilon > 0$ and $0 < R < 1$, there is a family of folded Reed-Solomon codes which have rate at least R and which can be list-decoded up to a fraction $1 - R - \varepsilon$ of errors in time $\mathrm{poly}(N)$, where N is the block length of the code.

3.1.3 Some remarks

We now make some salient remarks about the above linear-algebra based method for list decoding folded Reed-Solomon codes..

List size and runtime. To get the actual list of close-by codewords, one can prune the solution subspace, which unfortunately may take $q^s > n^s$ time in the worst-case. This quantity is about $n^{O(1/\varepsilon)}$ for the parameter choice $s \approx 1/\varepsilon$ which achieves a list decoding radius of $1 - R - \varepsilon$. The next remark shows that we may not be able to improve the worst-case list size bound of $\approx n^{1/\varepsilon}$ in this regime. This motivates our results in Section 3.3 where we show that using a carefully chosen subset of all possible degree at most $k - 1$ polynomials as messages, one can ensure that the list-size is much smaller while losing only a tiny amount in the rate.

Except for final step of pruning the subspace of candidate solutions, the decoding takes only quadratic time (and is perhaps even practical, as it just involves solving two structured linear systems). If some side information about the true message f is available that disambiguates the true message in the list [Gur03], that might also be useful to speed up the pruning.

Tightness of q^{s-1} bound. For folded Reed-Solomon codes, the upper bound of q^{s-1} on the number of solutions f to the Equation (3.9) cannot be improved in general. Indeed, let $A_0 = 0$, and A_i for $1 \le i \le s$ be the coefficient of Y^{i-1} in the polynomial $(Y-1)(Y-\gamma)\cdots(Y-\gamma^{s-2})$. Then for $0 \le \ell \le s-2$, we have

$$\begin{aligned} & A_1 \cdot X^\ell + A_2 \cdot (\gamma X)^\ell + \cdots + A_s \cdot (\gamma^{s-1} X)^\ell \\ &\quad = X^\ell \cdot \left(A_1 + A_2 \cdot \gamma^\ell + A_3 \cdot (\gamma^\ell)^2 + \cdots + A_s \cdot (\gamma^\ell)^{s-1} \right) \\ &\quad = 0 \,. \end{aligned}$$

By linearity, every polynomial $f \in \mathbb{F}_q[X]$ of degree at most $s-2$ satisfies (3.9).

We should add that this does *not* lead to any non-trivial list-size lower bound for decoding, as we do not know if such bad polynomials can occur as the output of the interpolation step, and moreover the pruning step could potentially reduce the size of the list further.

Requirement on γ. The argument in Lemma 3.3 only required that the order of γ is at least k, and not that γ is primitive. In particular, Theorem 3.4 holds as long as the order

of γ is at least k. The polynomial $X^{q-1} - \gamma$ is irreducible if and only if γ is primitive, and therefore the approach based on extension fields discussed in Chapter 2 requires γ to be primitive. Usually in constructions of Reed-Solomon codes, one takes the block length $n \approx q$ and therefore the dimension k is linear in q (for constant rate codes). So this weakened requirement on γ does not buy much flexibility in this case. However, in settings where one uses RS codes of small (say sub-constant) rate, for example in complexity-theoretic applications of list decoding, the new argument applies to a broader set of choices of evaluation points for the RS codes.

In Chapter 6, we will investigate list-decoding of folded Reed-Solomon codes when the order of the folding parameter γ is as small as *constant*. Although we are not able to prove a polynomial list size bound in this case, we will show how to construct a large subcode which guarantees a polynomial list size.

3.2 List-decoding derivative codes

We saw how the algebraic structure of folded Reed-Solomon codes gave enough power to allow for list-decoding up to the optimal error rate. In this section, we show that folding using *derivatives* allows us to achieve the same guarantee.

For a polynomial $f \in \mathbb{F}_q[X]$, we denote by f' its formal derivative, i.e. if $f(X) = f_0 + f_1 X + \ldots + f_\ell X^\ell$, then $f'(X) = \sum_{i=1}^{\ell} i f_i X^{i-1}$, where the coefficient i is $\underbrace{1 + \cdots + 1}_{i \text{ times}}$. We denote by $f^{(i)}$ the i'th formal derivative of f.

Definition 3.6 (m'th order derivative code)**.** Let $0 \leq m \in \mathbb{Z}$. Let $a_1, \ldots, a_N \in \mathbb{F}_q$ be distinct, let $n = Nm$, and let the parameters satisfy $m \leq k < n \leq q$. Further assume that $\mathrm{char}(\mathbb{F}_q) > k$.

The derivative code $\mathrm{Der}_q^{(m)}[n, k]$ over the alphabet $\mathbb{F}_q^m$ encodes a polynomial $f \in \mathbb{F}_q[X]$ of degree at most $k - 1$ by

$$\left(\begin{bmatrix} f(a_1) \\ f'(a_1) \\ \vdots \\ f^{(m-1)}(a_1) \end{bmatrix}, \begin{bmatrix} f(a_2) \\ f'(a_2) \\ \vdots \\ f^{(m-1)}(a_2) \end{bmatrix}, \ldots, \begin{bmatrix} f(a_N) \\ f'(a_N) \\ \vdots \\ f^{(m-1)}(a_N) \end{bmatrix} \right) . \tag{3.12}$$

Remark. This code has block length N and rate $R = k/n$. The minimum distance is $N - \lfloor \frac{k-1}{m} \rfloor \approx (1 - R)N$.

Note that, as with FRS codes, the case $m = 1$ is a Reed-Solomon code. These are also the univariate version of the *multiplicity codes* of [KSY14], where they were analyzed in the context of local decoding.

3.2.1 List decoding derivative codes

Suppose we have received the corrupted version of a codeword from the derivative code $\mathrm{Der}_q^{(m)}[n, k]$ as a string $\mathbf{y} \in (\mathbb{F}_q^m)^N$, which we will, as in the folded Reed-Solomon case, consider as an $m \times N$ matrix over $\mathbb{F}_q$ (recall $N = n/m$):

$$\begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mN} \end{pmatrix}. \tag{3.13}$$

The goal is to recover all polynomials f of degree at most $k - 1$ whose derivative encoding (3.12) agrees with $\mathbf{y}$ in at least t columns. This corresponds to decoding from $N - t$ symbol errors for the derivative code $\mathrm{Der}_q^{(m)}[n, k]$. When $t > (N + k/m)/2$, the polynomial f, if it exists, is unique, and in this regime an efficient decoding algorithm was given in [KSY14] by adapting the Welch-Berlekamp algorithm for Reed-Solomon codes [WB86, GS92].

We adapt the algebraic list-decoding method of Theorem 3.4 to the derivative code setting. As in the folded Reed-Solomon setting, the algorithm is a higher-dimensional analog of the Welch-Berlekamp algorithm consisting of two steps — (i) interpolation of an algebraic condition (that must be obeyed by all candidate polynomials f), and (ii) retrieving the list of candidate solutions f (from the algebraic condition found by the interpolation step). For the same settings of parameters as in Section 3.1, we achieve the same decoding radius, but the runtime bound we show for the interpolation and retrieval steps is a larger polynomial.

Recently, a different list decoding algorithm for derivative codes was given in [Kop12]. Similar to the relationship between the algorithms in Theorem 2.19 and Theorem 3.4, this algorithm uses multiplicities to achieve a higher decoding radius than our algorithm for a fixed setting of parameters, at the cost of a more complicated algorithm and analysis.

Interpolation

Let $\mathcal{W}$ denote the $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q[X, Y_1, \ldots, Y_m]$ consisting of polynomials that have total degree at most 1 in the Y_i's, i.e, $\mathcal{W}$ contains polynomials of the form $B_0(X) + B_1(X)Y_1 + B_2(X)Y_2 + \cdots + B_m(X)Y_m$ for some polynomials $B_i \in \mathbb{F}_q[X]$.

Let Δ be the $\mathbb{F}_q$-linear map on $\mathcal{W}$ defined as follows: For $p \in \mathbb{F}_q[X]$, and $1 \leq i \leq m$,

$$\Delta(p)(X, Y_1, \ldots, Y_m) = p'(X) \tag{3.14}$$

and

$$\Delta(pY_i)(X, Y_1, \ldots, Y_m) = p'(X)Y_i + p(X)Y_{i+1}. \tag{3.15}$$

where we take $Y_{m+1} = 0$ for definitiveness.

It is not a coincidence that this map is reminiscent of the chain rule for derivatives, and the following lemma shows why Δ is useful to us.

Lemma 3.7. Suppose $P(X, Y_1, \ldots, Y_m) \in \mathbb{F}_q[X, Y_1, \ldots, Y_m]$ has degree at most 1 in $Y_1, \ldots, Y_i$ for some $i < m$ and degree 0 in $Y_{i+1}, \ldots, Y_m$. Then

1. $\frac{d}{dx}P\big(X, f(X), f'(X), f^{(2)}(X), \ldots, f^{(m)}(X)\big) =$
   $(\Delta P)\big(X, f(X), f'(X), \ldots, f^{(m)}(X)\big)$, and
2. ΔP has degree at most 1 in $Y_1, \ldots, Y_{i+1}$ and degree 0 in $Y_{i+2}, \ldots, Y_m$.

Proof. By linearity, for $p(X) \in \mathbb{F}_q[X]$, it suffices to check this for $p(X)$ and $p(X)Y_j$ for $j \leq i$.

We have $\Delta p = p'$, and $\Delta(pY_j) = p'Y_j + pY_{j+1}$, which both have degree at most 1 in $Y_1, \ldots, Y_{i+1}$ and degree 0 in $Y_{i+2}, \ldots, Y_m$. Moreover,

$$\Delta(pY_j)\big(X, f(X), f'(X), \ldots, f^{(m)}(X)\big) = p'f^{(j-1)} + pf^{(j)} = \frac{d}{dx}pf^{(j-1)},$$

as desired. □

Let s, $1 \leq s \leq m$, be an integer parameter in the decoding algorithm. The goal in the interpolation step is to interpolate a **nonzero** polynomial $Q \in \mathbb{F}_q[X, Y_1, Y_2, \ldots, Y_m]$ of the form

$$A_0(X) + A_1(X)Y_1 + A_2(X)Y_2 + \cdots + A_s(X)Y_s \tag{3.16}$$

satisfying the following conditions for each i, $1 \le i \le N$:

$$Q(a_i, y_{1i}, \ldots, y_{si}) = 0 \text{ and } (\Delta^j Q)(a_i, y_{1i}, \ldots, y_{mi}) = 0 \quad \text{for } j = 1, \ldots, m-s, \tag{3.17}$$

where Δ^j denotes the j-fold composition of the map Δ.

Observation. For each i, the conditions (3.17) are a collection of $(m-s+1)$ homogeneous linear constraints on the coefficients of the polynomial Q.

Lemma 3.7 gives us the following.

Corollary 3.8. Suppose Q of the form (3.16) satisfies the conditions (3.17). If the received word (3.13) agrees with the encoding of f at location i, that is, $f^{(j)}(a_i) = y_{j+1,i}$ for $0 \le j < m$, then the univariate polynomial $\Lambda(X) := Q(X, f(X), \ldots, f^{(s-1)}(X))$ satisfies $\Lambda(a_i) = 0$ as well as $\Lambda^{(k)}(a_i) = 0$ for $k = 1, \ldots, m-s$, where $\Lambda^{(k)}(X)$ is the k'th derivative of Λ.

We next argue, similarly to Lemma 3.1 in the folded Reed-Solomon case, that a nonzero interpolation polynomial Q exists and can be found efficiently. In this case, we only claim cubic runtime for solving the linear system (although we can also state a runtime of $O(n^\omega)$ using faster matrix multiplication).

Lemma 3.9. Let

$$D = \left\lfloor \frac{N(m-s+1) - k + 1}{s+1} \right\rfloor. \tag{3.18}$$

Then a nonzero Q of the form (3.16) satisfying the interpolation conditions (3.17) with $\deg(A_0) \le D + k - 1$ and $\deg(A_j) \le D$ for $1 \le j \le s$ exists and can be found in $O(n^3)$ field operations over $\mathbb{F}_q$ by solving a homogeneous linear system over $\mathbb{F}_q$ with at most n constraints and variables.

Proof. Under the stated degree restrictions, the number of monomials in Q is

$$(D+1)s + D + k = (D+1)(s+1) + k - 1 > N(m-s+1). \tag{3.19}$$

where the last inequality follows from the choice (3.18) of D. The number of homogeneous linear equations imposed on the coefficients of Q in order to meet the interpolation conditions (3.17) is $N \cdot (m-s+1)$. As this is less than the number of monomials in Q, the existence of a nonzero Q follows, and it can be found by solving a linear system over $\mathbb{F}_q$ with at most $Nm = n$ constraints and at most $N \cdot (m-s+1) + (s+1) < Nm = n$ variables. □

Suppose we have a polynomial $Q(X, Y_1, \ldots, Y_s)$ satisfying the interpolation conditions (3.17). The following lemma gives an identity satisfied by any f which has good agreement with the received word.

Lemma 3.10. Let Q satisfy the conclusion of Lemma 3.9. If $f \in \mathbb{F}_q[X]$ is a polynomial of degree at most $k-1$ whose derivative encoding (3.12) agrees with the received word $\mathbf{y}$ in at least t columns for $t > \frac{D+k-1}{m-s+1}$, then

$$Q\big(X, f(X), f'(X), \ldots, f^{(s-1)}(X)\big) = 0.$$

Proof. Let $\Lambda(X) = Q(X, f(X), \ldots, f^{(s-1)}(X))$. By Corollary 3.8, an agreement in column i means that $\Lambda(X)$ satisfies $\Lambda(a_i) = 0$ and that the kth derivative $\Lambda^{(k)}(a_i)$ is also zero for $k = 1, \ldots, m-s$. In particular, t column agreements yield at least $t \cdot (m-s+1)$ roots (counting multiplicities) for Λ.

The degree of Λ is at most $D + k - 1$, as f and each of its derivatives has degree at most $k-1$. Then as Λ is univariate of degree at most $D+k-1$, Λ has at most $D+k-1$ roots if it is nonzero. Thus if $t > (D+k-1)/(m-s+1)$, it must be that $\Lambda(X) = 0$. $\square$

Retrieving candidate polynomials

With our chosen value of D from (3.18), the preceding section implies that any f which agrees with $\mathbf{y}$ on at least

$$\frac{N}{s+1} + \frac{s}{s+1}\frac{k}{m-s+1} \tag{3.20}$$

columns satisfies $Q\big(X, f(X), f'(X), \ldots, f^{(s-1)}(X)\big) = 0$. So in the second step, our goal is to find all polynomials f of degree at most $k-1$ such that

$$A_0(X) + A_1(X)f(X) + A_2(X)f'(X) + \ldots + A_s(X)f^{(s-1)}(X) = 0. \tag{3.21}$$

Let $A_i(X) = \sum_{j=0}^{\deg(A_i)} a_{i,j}X^j$ for each i. Note that the above constraint (3.21) gives a *linear system* over $\mathbb{F}$ in the coefficients of $f = f_0 + f_1X + \cdots + f_{k-1}X^{k-1}$. In particular, the set of solutions $(f_0, f_1, \ldots, f_{k-1})$ is an affine space, and we can find it by solving the linear system. Our goal now is to bound the dimension of the space of solutions by exposing its special structure and also use this to efficiently find an explicit basis for the space.

Lemma 3.11. It suffices to give an algorithm in the case that the constant term a_{s0} of A_s is nonzero.

Proof. If $A_s(X) \not\equiv 0$, then since $\deg(A_s) \leq D < Nm \leq q$, there is some $\alpha \in \mathbb{F}_q$ such that $A_s(\alpha) \neq 0$. This means that we can consider a "translate" of this problem by α; that is, $A_s(X+\alpha)$ has nonzero constant term, so we can solve the system with the translated polynomial $Q(X+\alpha, Y_1, \ldots, Y_m)$ and recover candidate messages by translating each solution $g(X)$ to $f(X) = g(X-\alpha)$.

If $A_s(X) = 0$, we simply reduce the problem to a smaller one with s rather than $s+1$ interpolation variables. Note that this must terminate since Q is nonzero and so there is at least one A_i for $i \geq 1$ which is nonzero. □

We can now show the following analogue of Lemma 3.3:

Lemma 3.12. Assume that $\mathrm{char}(\mathbb{F}_q) > k$. If $a_{s,0} \neq 0$, the solution space to (3.21) has dimension at most $s-1$. Furthermore, a basis for this subspace can be found in $O(sk^2)$ operations over $\mathbb{F}_q$.

Proof. For each power X^i, the coefficient of X^i in the polynomial $A_0(X)+A_1(X)f(X)+\cdots+A_s(X)f^{(s-1)}(X)$ is

$$\begin{aligned}
&a_{0,i} + \big(a_{1,0}f_i + a_{1,1}f_{i-1} + \cdots + a_{1,i}f_0\big) \\
&\quad + \big(a_{2,0}(i+1)f_{i+1} + a_{2,1}if_i + \cdots + a_{2,i}f_1\big) \\
&\quad + \cdots + \big(a_{s,0}(i+s-1)(i+s-2)\cdots(i+1)f_{i+s-1} \\
&\quad + \cdots + a_{s,i}(s-1)!f_{s-1}\big) \\
&= a_{0,i} + \sum_{j=1}^{s}\sum_{\ell=0}^{i} \frac{(\ell+j-1)!}{\ell!} a_{j,i-\ell} f_{\ell+j-1}\,. \qquad (3.22)
\end{aligned}$$

If $(f_0, \ldots, f_{k-1})$ is a solution to (3.21), then this coefficient must be zero for every i.

Our expression for the coefficient of X^i for each i depends only on f_j for $j < i+s$. Furthermore, the coefficient of f_{i+s-1} in this expression is $a_{s,0}\cdot(i+s-1)(i+s-2)\cdots(i+1)$, which is nonzero when $i+s \leq k$ since $\mathrm{char}(\mathbb{F}_q) > k$. Thus, if we fix $f_0, f_1, \ldots, f_{s-2}$, the rest of the message symbols $f_{s-1}, \ldots, f_{k-1}$ are uniquely determined. In particular, the dimension of the solution space is at most $s-1$. Also, by (3.22), each f_l, $l \geq s-1$, is specified as a linear combination of f_i for $i < l$, and implies that we can compute a basis of the solution space $(f_0, \ldots, f_{k-1})$ using $O(sk^2)$ field operations. □

Combining Lemmas 3.10 and 3.12, and recalling the bound (3.20) on the number of agreements for successful decoding, we have our result on list-decoding derivative codes.

Theorem 3.13. For the derivative code $\mathrm{Der}_q^{(m)}[n,k]$ (where $\mathrm{char}(\mathbb{F}_q) > k$) of block length $N = n/m$ and rate $R = k/n$, the following holds for all integers s, $1 \le s \le m$. Given a received word $\mathbf{y} \in \mathbb{F}_q^{m \times N}$, in $O\big(n^3 + sk^2\big)$ operations over $\mathbb{F}_q$, one can find a basis for a subspace of dimension at most $s-1$ that contains all message polynomials $f \in \mathbb{F}_q[X]$ of degree less than k whose derivative encoding (3.12)differs from $\mathbf{y}$ in at most a fraction

$$\frac{s}{s+1}\left(N - \frac{k}{(m-s+1)}\right)$$

of the N codeword positions.

Now by setting $s \approx 1/\varepsilon$ and $m \approx 1/\varepsilon^2$, and recalling that the rate of $\mathrm{Der}_q^{(m)}[n,k]$ is $k/n = k/(Nm)$, we can conclude the following.

Corollary 3.14. For all $R \in (0,1)$ and all $\varepsilon > 0$, for a suitable choice of parameters, there are derivative codes $\mathrm{Der}_q^{(m)}[n,k]$ of rate at least R which can be list decoded from a fraction $1 - R - \varepsilon$ of errors with a list-size of $q^{O(1/\varepsilon)}$.

3.2.2 Some remarks

Tightness of q^{s-1} bound. As in the folded Reed-Solomon case, the bound of Lemma 3.12 is tight for arbitrary linear systems. Indeed, if

$$Q(X, Y_1, \ldots, Y_s) = \sum_{i=0}^{s-1} \frac{(-1)^i}{i!} X^i Y_{i+1},$$

then *any* polynomial $f(X)$ of degree less than s with zero constant term satisfies the identity $Q(X, f(X), \ldots, f^{(s-1)}(X)) = 0$. This is because any monomial $f(X) = X^j$ for $0 < j \le s-1$ is a solution, and our solution space is linear. Again as in the FRS case, we do not know if such a bad polynomial can occur as the output of the interpolation step when decoding a noisy codeword of the derivative code.

Connection with the Wronskian. Lemma 3.12 can also be justified by appealing to the Wronskian criterion for linear independence.[2] The Wronskian $W(g_1, g_2, \ldots, g_s)$ of s polynomials $g_1, g_2, \ldots, g_s \in \mathbb{F}[X]$ which have order $s-1$ derivatives is defined to be the

[2]We thank Swastik Kopparty for pointing this out.

determinant

$$\begin{vmatrix} g_1(X) & g_2(X) & \cdots & g_s(X) \\ g_1'(X) & g_2'(X) & \cdots & g_s'(X) \\ \vdots & \vdots & \ddots & \vdots \\ g_1^{(s-1)}(X) & g_2^{(s-1)}(X) & \cdots & g_s^{(s-1)}(X) \end{vmatrix}.$$

If the polynomials $g_1, g_2, \ldots, g_s$ are linearly dependent over $\mathbb{F}$, then the Wronskian is clearly 0. Conversely, for finite fields, it can be shown that if the characteristic is bigger than the degree and $g_1, g_2, \ldots, g_s$ are linearly independent over $\mathbb{F}$, then the Wronskian is not zero. If $g_1, g_2, \ldots, g_s$ are all solutions to

$$A_1(X)f(X) + A_2(X)f'(X) + \cdots + A_s(X)f^{(s-1)}(X) = 0$$

(the linear system underlying (3.21)), then the rows of the matrix defining the Wronskian are linearly dependent (over the rational function field $\mathbb{F}(X)$), which implies that the Wronskian is zero. Thus we cannot have s linearly independent solutions to the above system, and therefore the rank of the affine space of solutions to (3.21) is at most $s-1$.

Decoding derivative codes with side information. The decoding described in the previous section consists of trying all choices for the coefficients $f_0, \ldots, f_{s-2}$ and using each to uniquely determine a candidate for f. Note however that for each i, the f_i is essentially the ith derivative of f evaluated at 0, and can be recovered as $f^{(i)}(0)/i!$. Thus if the decoder somehow knew the correct values of f and its first $s-1$ derivatives at 0, f could be recovered uniquely (as long as $A_s(0) \neq 0$).

Now, suppose the encoder could send a small amount of information along a noiseless side channel in addition to sending the (much longer) codeword on the original channel. In such a case, the encoder could choose $\alpha \in \mathbb{F}_q$ uniformly at random and transmit $f(\alpha), f'(\alpha), \ldots, f^{(s-1)}(\alpha)$ on the noiseless channel. The decoding then fails only if $A_i(\alpha) = 0$ for i which is the largest index such that $A_i(X) \neq 0$. As the $A_i(X)$ have bounded degree, by increasing the field size q, f can be uniquely recovered with probability arbitrarily close to 1. More precisely, we have the following claim.

Theorem 3.15. Given a uniformly random $\alpha \in \mathbb{F}_q$ and the values $f(\alpha), f'(\alpha), \ldots, f^{(s-1)}(\alpha)$ of the message polynomial f, the derivative code $\mathrm{Der}_q^{(m)}[n,k]$ can be uniquely decoded from up to

$$\frac{s}{s+1}\left(N - \frac{k}{m-s+1}\right)$$

errors with probability at least $1 - n/(sq)$ over the choice of α.

Proof. As in the proof of Lemma 3.11, as long as $A_s(\alpha) \neq 0$, we may translate the problem by α and use the values $f(\alpha), f'(\alpha), \ldots, f^{(s-1)}(\alpha)$ to uniquely determine the shifted coefficients $g_0, \ldots, g_{s-1}$.

As $A_s \neq 0$, and A_s is univariate of degree at most D, A_s has at most D roots, and so the probability that $A_s(\alpha) \neq 0$ is at least $1 - D/q \geq 1 - \frac{n}{sq}$, where the last inequality follows from our choice of $D \leq n/s$ in (3.18). □

Remark. In the context of communicating with side information, there is a generic, black-box solution combining list-decodable codes with hashing to guarantee unique recovery of the correct message with high probability [Gur03]. In such a scheme, the side information consists of a random hash function h and its value $h(f)$ on the message f. The advantage of the solution in Theorem 3.15 is that there is *no need* to compute the full list (which is the computationally expensive step, since the list size bound depends exponentially on s) and then prune it to the unique solution. Rather, we can uniquely identify the first $(s-1)$ coefficients of the polynomial $f(X+\alpha)$ in the linear system (3.21), after applying the shift $X \mapsto X + \alpha$, as $f(\alpha), f'(\alpha), \ldots, f^{(s-2)}(\alpha)$. Then, as argued in the proof of Lemma 3.12, the remaining coefficients are determined as linear combinations of these $s-1$ coefficients. So the whole algorithm can be implemented in cubic time.

Note that we do not know how to apply the approach of Theorem 3.15 to the case of folded Reed-Solomon codes. The key difference is that in the derivative code case, it is known that the decoder will need the first $s-1$ message coefficients. In the folded Reed-Solomon case, the required coefficients could depend on the interpolated polynomial Q, which would mean that the correct values could not be sent ahead of time.

Remark. The decoder could use the columns of the received word $\mathbf{y}$ as a guess for the side information $f(a_i), f'(a_i), \ldots, f^{(s-2)}(a_i)$ for $i = 1, 2, \ldots, N$. Since f agrees with $\mathbf{y}$ on more than $t > RN$ positions, as long as $A_s(a_i) = 0$ for less than t of the evaluation points a_i, we will recover every solution f this way. This would lead to a list size bound of at most $N - t < N$. Unfortunately, however, there seems to be no way to ensure that A_s does not vanish at most (or even all) of the points a_i used for encoding. But perhaps some additional ideas can be used to make the list size polynomial in both q, s, or at least $\exp\big(O(s)\big)q^c$ for some absolute constant c.

3.3 Improved list size via subspace-evasive sets

Based on Theorems 3.4 and 3.13, in this section we pursue one possible approach to improve the provable worst-case list size bound for list decoding up to a fraction $1 - R - \varepsilon$

of errors. Instead of allowing all polynomials $f_0 + f_1 X + \cdots + f_{k-1} X^{k-1}$ of degree less than k as messages, the idea is to restrict the coefficient vector $(f_0, f_1, \ldots, f_{k-1})$ to belong to some special subset $\mathcal{V} \subseteq \mathbb{F}_q^k$, satisfying the following two conflicting demands:

Largeness: The set $\mathcal{V}$ must be large, say $|\mathcal{V}| \geq q^{(1-\varepsilon)k}$, so that the rate is reduced by at most a $(1-\varepsilon)$ factor.

Low intersection with subspaces: For every subspace $S \subset \mathbb{F}_q^k$ of dimension s, $|S \cap \mathcal{V}|$ is small.

If $\mathcal{V}$ satisfies this property, we will say that $\mathcal{V}$ is (s, L)**-subspace-evasive**. The field $\mathbb{F}_q$ and the ambient dimension k will be fixed in our discussion.

Using such a set $\mathcal{V}$ will ensure that after pruning an affine subspace output by the algorithms of Theorem 3.4 and 3.13, the number of codewords will be at most L. (Note that an affine subspace of dimension $s-1$ is contained in a subspace of dimension s.) Thus the list size will go down from q^{s-1} to L.

Subspace-evasive subsets were used in [PR04] to construct bipartite Ramsey graphs, and in fact we borrowed the term *evasive* from that work. In their work, the underlying field was $\mathbb{F}_2$ and the subsets had to be evasive for dimension $s \approx k/2$. Our interest is in a different regime — we can work over large fields, and are interested in evasiveness with respect to s-dimensional subspaces for constant s.

A random large subset of $\mathbb{F}_q^k$ meets the low subspace intersection requirement very well, as shown below. The argument is straightforward; a similar bound appears in [BK03] in the geometric context of point-subspace incidences.

Lemma 3.16. Let k be a large enough positive integer, and let $s < k/4$ be a positive integer. For some α with $0 < \alpha < k/4$, let $\mathcal{W}$ be a random subset of $\mathbb{F}_q^k$ chosen by including each $x \in \mathbb{F}_q^k$ in $\mathcal{W}$ with probability $q^{-s-\alpha}$. Then with probability at least $1 - q^{-k}$, $\mathcal{W}$ satisfies both the following conditions: (i) $|\mathcal{W}| \geq q^{k-s-\alpha}/2$, and (ii) $\mathcal{W}$ is $(s, 2sk/\alpha)$-subspace-evasive.

Proof. The first part follows by a standard Chernoff bound calculation: the expected value of $|\mathcal{W}|$ equals $q^{k-s-\alpha}$, and thus the probability that it is less than half the expected value is at most $\exp(-q^{k-s-\alpha}/8) \leq q^{-k}$.

For the second part, fix a subspace $S \subseteq \mathbb{F}_q^k$ of dimension s, and a subset $T \subseteq S$ of size $t = \lceil 2ks/\alpha \rceil$. The probability that $\mathcal{W} \supseteq T$ equals $q^{-(s+\alpha)t}$. By a union bound over the at most q^{ks} choices for the s-dimensional subspace S, and the at most q^{st} choices of t-element subsets T of S, we get that the probability that $\mathcal{W}$ is not $(s, t-1)$-subspace-evasive is at most $q^{ks+st} \cdot q^{-(s+\alpha)t} \leq q^{-ks}$ since $t \geq 2ks/\alpha$. □

Picking $\alpha \approx \varepsilon k$, the above guarantees the existence of subsets $\mathcal{W}$ of $\mathbb{F}_q^k$ of size $q^{(1-\varepsilon)k-s-1}$ which are $\big(s, O(s/\varepsilon)\big)$-subspace-evasive. Restricting the coefficient vector $(f_0, f_1, \ldots, f_{k-1})$ of the message polynomial to belong to such a subset will guarantee a list-size upper bound of $O(s/\varepsilon)$ in Theorem 3.4 or Theorem 3.13. This list-size bound is independent of q, and for the choice $s \approx 1/\varepsilon$ which enables list decoding a fraction $1 - R - \varepsilon$ of errors, it is $O(1/\varepsilon^2)$. This is quite close to the bound of $O(1/\varepsilon)$ achieved by random codes in [GHSZ02].

Unfortunately, an explicit construction of subspace-evasive subsets approaching the trade-off guaranteed by the probabilistic construction of Lemma 3.16 is not known. This appears to be a challenging and extremely interesting question. One natural choice for such a subset would be some *variety* $\mathcal{V} \subseteq \mathbb{F}_q^k$ defined by a collection of polynomial equations, i.e., $\mathcal{V} = \{\mathbf{a} \in \mathbb{F}_q^k \mid g_1(\mathbf{a}) = g_2(\mathbf{a}) = \cdots = g_l(\mathbf{a}) = 0\}$ for some polynomials $g_1, g_2, \ldots, g_l \in \mathbb{F}_q[Z_1, Z_2, \ldots, Z_k]$. Indeed for $s = 1$ and $s = k - 1$, varieties in $\mathbb{F}_q^k$ (the modular moment surface and modular moment curve) with low intersection with s-dimensional affine subspaces are known [BK03]. In Section 3.4, we discuss a construction, due to Dvir and Lovett, of subspace-evasive sets based on varieties which, though far from the probabilistic bounds, are sufficient to reduce the list size to constant in our case.

3.3.1 Pseudorandom construction of subspace-evasive subsets

The construction of Lemma 3.16 takes exponential time and produces a random unstructured set that takes exponential space to store. In this section, we show that a subset with similar guarantees can be constructed in probabilistic polynomial time, producing a polynomial size representation of the constructed subspace-evasive set. The idea is to note that the probabilistic argument to argue about (s, t)-subspace-evasiveness only needed t-wise independence and not complete independence of different elements of $\mathbb{F}_q^k$ landing in the random subset $\mathcal{W}$. We now describe such a pseudorandom construction.

For some parameter $\zeta \in (0, 1/2)$, let $k' = (1 - \zeta)k$. Let $\mathbb{K}$ be the extension field $\mathbb{F}_{q^{k'}}$, and fix an arbitrary basis B of $\mathbb{K}$ over $\mathbb{F}_q$.

We will define a subspace-evasive embedding of $\mathbb{F}_q^{k'}$ into $\mathbb{F}_q^k$ as follows. Because we have fixed the basis B, we can consider any $v \in \mathbb{F}_q^{k'}$ as an element of $\mathbb{K} = \mathbb{F}_{q^{k'}}$. Let $P \in \mathbb{K}[X]$ be a polynomial of degree at most t, and let $Q(v)$ be the first ζk coordinates of $P(v)$ with respect to the basis B.

Then we will map $v \in \mathbb{F}_q^{k'}$ to $\big(v, Q(v)\big) \in \mathbb{F}_q^k$. Let $\mathcal{W}_{\zeta,k}(P)$ be the image of $\mathbb{F}_q^{k'}$ under this map. As the map is injective, $|\mathcal{W}_{\zeta,k}(P)| = q^{k'} = q^{(1-\zeta)k}$.

Lemma 3.17. Let $k \geq 1$ be an integer. Let $\zeta \in (0, 1/2)$, and let s be an integer satisfying $1 \leq s \leq \zeta k/2$. Let $t \geq \lceil 4s/\zeta \rceil$ be a positive integer and $P \in \mathbb{K}[X]$ be a random polynomial of degree at most t.

Define $\mathcal{W} = \mathcal{W}_{\zeta,k}(P)$. Then with probability at least $1 - q^{-ks}$ over the choice of P, $\mathcal{W}$ is a $(s, 4s/\zeta)$-subspace-evasive subset of $\mathbb{F}_q^k$ of size $q^{(1-\zeta)k}$.

Proof. We already argued that $|\mathcal{W}| = q^{(1-\zeta)k}$. For each $\mathbf{x} \in \mathbb{F}_q^k$, the probability that $\mathbf{x}$ is in $\mathcal{W}$ is the probability that the last ζk coordinates are Q evaluated at the first $(1-\zeta)k$ coordinates. As P was random, this is $q^{-\zeta k}$.

Since the values of P at any t distinct points in $\mathbb{K}$ are independent, the events $\mathbf{x} \in \mathcal{W}$ are t-wise independent as long as no two share the same initial k' coordinates.

Now fix a subspace $S \subseteq \mathbb{F}_q^k$ of dimension s, and a subset $T \subseteq S$ of size t. Let us compute the probability that $T \subseteq \mathcal{W}$. As $\mathcal{W}$ contains exactly one element for each setting of the initial k' coordinates, we may assume no two elements in T share the same initial k' coordinates. The t events $\beta \in \mathcal{W}$ for various $\beta \in T$ are independent due to the above t-wise independence property. Thus the probability that $T \subseteq \mathcal{W}$ equals $q^{-\zeta kt}$. The remaining calculation is as in Lemma 3.16 and involves a union bound over the at most q^{ks} choices for the s-dimensional subspace S, and the at most q^{st} choices of t-element subsets T of S. □

Note that the set $\mathcal{W}$ has a compact representation, and given P of degree $t \leq O(s/\zeta)$, the bijection from $\mathbb{F}_{q^{k'}}$ to $\mathcal{W}$ can be computed using $\mathrm{poly}(k, s, 1/\zeta)$ $\mathbb{F}_q$-operations, and membership in $\mathcal{W}$ can be checked in the same time. This implies that we can efficiently encode any $v \in \mathbb{F}_q^{k'}$ by computing its representative in $\mathcal{W}$ and then applying either the folded Reed-Solomon or derivative encoding. Combining this with Theorems 3.4 and 3.13, we can conclude the following final result.

Theorem 3.18. For any ζ, $0 < \zeta < 1/2$, there is a Monte Carlo construction of a subcode C of $\mathrm{FRS}_q^{(m)}[n, k]$ or $\mathrm{Der}_q^{(m)}[n, k]$ of rate $(1-\zeta)R$ where $R = k/n$, consisting of encodings of polynomials whose coefficients belong to a subspace-evasive subset $\mathcal{W} \subset \mathbb{F}_q^k$, such that

(i) there is a efficient encoder computing a bijection $\mathbb{F}_q^{(1-\zeta)k} \to C$ using $\mathrm{poly}(n, m, 1/\zeta)$ $\mathbb{F}_q$-operations, and

(ii) with high probability C can be list decoded from error fraction $\frac{s}{s+1}\left(1 - \frac{mR}{m-s+1}\right)$ for any $1 \leq s \leq m$ in $q^{O(s)}$ time with an output list size of at most $O(s/\zeta)$.

In particular, picking $\zeta = \Theta(\varepsilon)$, $s = \Theta(1/\varepsilon)$ and $m = \Theta(1/\varepsilon^2)$, for any desired $R' \in (0,1)$, the construction yields codes of rate R' which can be list decoded from a fraction $1 - R' - \varepsilon$ of errors in $q^{O(1/\varepsilon)}$ time, with at most $O(1/\varepsilon^2)$ codewords output in the list.

3.4 Epilogue: Subsequent work

Explicit subspace-evasive sets. Following the work in this chapter, the authors of [DL12] gave an *explicit* construction of a set $S \subseteq \mathbb{F}_q^n$ of size at least $q^{(1-\varepsilon)n}$ which is $\left(s, (s/\varepsilon)^s\right)$-subspace-evasive. Their construction uses so-called *everywhere-finite varieties*, and has the nice property that encoding can be done efficiently. Additionally, the intersection of S with any s-dimensional subspace can be computed in time which is polynomial in the size of the intersection which avoids the q^s time search in the pruning step of the decoding. Using this construction, they are able to show the following.

Theorem 3.19 ([DL12])**.** For every R and ε, there exists an explicit family of codes $C \subset \Sigma^n$ with rate R that can be list-decoded from a fraction $1 - R - \varepsilon$ of errors in quadratic time and with list size $(1/\varepsilon)^{O(1/\varepsilon)}$.

However, it is not known how to improve the list size bound to match the existential $O(s/\varepsilon)$ bound of Lemma 3.17.

This construction will come up again in Chapter 6, where we use them as an ingredient to construct different subspace-evasive objects. The details of the construction can be found there, as Theorem 6.11.

Folded algebraic-geometric codes. The linear-algebraic approach given in this work was refined in [GX12] to list-decode certain folded algebraic-geometric codes. One advantage of using algebraic-geometric codes is that the alphabet size can be kept small. Note that the alphabet size for both folded Reed-Solomon codes and derivative codes grows with the block length of the code.

The authors of [GX12] use an extension of subspace-evasive sets called *hierarchical* subspace-evasive sets to prune the subspace of candidate messages. Instantiated with one of the optimal function field towers due to Garcia and Stichtenoth, this enables list-decoding up to a fraction $(1-R-\varepsilon)$ of errors with a list size of $O(1/\varepsilon)$ over an alphabet of size $(1/\varepsilon)^{O(1/\varepsilon^2)}$, almost matching the random coding bound in all aspects simultaneously.

Chapter 4

Deletion codes

In which Mallory hits backspace • Greedy codes work • Concatenated codes are efficient

This chapter begins a study of the trade-off between the rate of a code and its correctable deletion fraction, for constant alphabet size. As mentioned in Chapter 2, this is an area about which we know surprisingly little. For example, we still do not know the largest deletion fraction which is correctable by a positive-rate binary code.

The results in this chapter make some progress towards our goal of understanding the optimal trade-off. We first analyze what is possible existentially, giving some basic combinatorial constructions of deletion codes. Then we use these constructions to give deletion codes which can be encoded and decoded *efficiently*. Our codes achieve the following qualitative goals for small ε:

- We give efficient codes of positive rate which can correct an error fraction of $1 - \varepsilon$, and

- we give efficient *binary* codes of rate $1 - \varepsilon$ which can correct a positive fraction of deletions.

Our explicit constructions exploit the idea of code concatenation, introduced in Chapter 2, which allows us to combine the positive aspects of two kinds of deletion codes, obtaining constant-alphabet codes with good rate and deletion correction ability.

The results in this chapter appear in joint work with Guruswami in [GW14].

4.1 Existential bounds for deletion codes

In this section, we show the existence of deletion codes in certain ranges of parameters, without the requirement of efficient encoding or decoding. The proofs (found in Section 4.5) follow from standard probabilistic arguments, but to the best of our knowledge, these bounds were not known previously. The codes of Theorem 4.4 will be used as inner codes in our final concatenated constructions.

Throughout, we will write $[k]$ for the set $\{1, \ldots, k\}$. We will also use the binary entropy function, defined for $\delta \in [0, 1]$ as $h(\delta) = \delta \log \frac{1}{\delta} + (1 - \delta) \log \frac{1}{1-\delta}$.

Recall that constructing a large code in $[k]^m$ which can correct from a δ fraction of deletions is equivalent to constructing a large set of strings such that for each pair, their longest common subsequence (LCS) has length less than $(1 - \delta)m$.

We first consider how well a random code performs, using the following theorem from [KLM04], which upper bounds the probability that a pair of randomly chosen strings has a long LCS.

Theorem 4.1 ([KLM04], Theorem 1)**.** For every $\gamma > 0$, there exists $c > 0$ such that if k and $m/\sqrt{k}$ are sufficiently large, and u, v are chosen independently and uniformly from $[k]^m$, then

$$\Pr\left[\left|\mathrm{LCS}(u, v) - 2m/\sqrt{k}\right| \geq \frac{\gamma m}{\sqrt{k}}\right] \leq e^{-cm/\sqrt{k}}.$$

Fixing γ to be 1, we obtain the following.

Proposition 4.2. Let $\varepsilon > 0$ be sufficiently small and let $k = (4/\varepsilon)^2$. There exists a code $C \subseteq [k]^m$ of rate $R = \Omega\big(\varepsilon/\log(1/\varepsilon)\big)$ which can correct a $1 - \varepsilon = 1 - 4/\sqrt{k}$ fraction of deletions.

The following results, and in particular Corollary 4.6, show that we can nearly match the performance of random codes using a simple greedy algorithm.

We first bound the number of strings which can have a fixed string s as a subsequence.

Lemma 4.3. Let $\delta \in (0, 1/k)$, set $\ell = (1 - \delta)m$, and let $s \in [k]^\ell$. The number of strings $s' \in [k]^m$ containing s as a subsequence is at most

$$\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} k^{m-t}(k-1)^{t-\ell} \leq k^{m-\ell}\binom{m}{\ell}.$$

When $k = 2$, we have the estimate

$$\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} 2^{m-t} \leq \delta m \binom{m}{\ell}.$$

Theorem 4.4. Let $\delta, \gamma > 0$. Then for every m, there exists a code $C \subseteq [k]^m$ of rate $R = 1 - \delta - \gamma$ such that:

- C can be corrected from a δ fraction of worst-case deletions, provided $k \geq 2^{2h(\delta)/\gamma}$.
- C can be found, encoded, and decoded in time $k^{O(m)}$.

Moreover, when $k = 2$, we may take $R = 1 - 2h(\delta) - \log(\delta m)/m$.

Remark. The authors of [KMTU11] show a similar result for the binary case, but use the weaker bound in Lemma 4.3 to get a rate of $1 - \delta - 2h(\delta)$.

With a slight modification to the proof of Theorem 4.4, we obtain the following construction, which will be used in Section 4.3. The so-called "β-dense" property will help us to distinguish codewords, which have high Hamming weight, from long strings of zeroes.

Proposition 4.5. Let $\delta, \beta \in (0, 1)$. Then for every m, there exists a code $C \subseteq \{0,1\}^m$ of rate $R = 1 - 2h(\delta) - O(\log(\delta m)/m) - 2^{-\Omega(\beta m)}/m$ such that:

- For every string $s \in C$, s is "β-dense": every interval of length βm in s contains at least $\beta m/10$ ones,
- C can be corrected from a δ fraction of worst-case deletions, and
- C can be found, encoded, and decoded in time $2^{O(m)}$.

In the high-deletion regime, we have the following corollary to Theorem 4.4, obtained by setting $\delta = 1 - \varepsilon$ and $\gamma = (1 - \theta)\varepsilon$, and noting that $h(\varepsilon) \leq \varepsilon \log(1/\varepsilon) + 2\varepsilon$ when $\varepsilon < 1/2$.

Corollary 4.6. Let $1/2 > \varepsilon > 0$ and $\theta \in (0, 1/3]$. There for every m, there exists a code $C \subseteq [k]^m$ of rate $R = \varepsilon \cdot \theta$ which can correct a $1 - \varepsilon$ fraction of deletions in time $k^{O(m)}$, provided $k \geq 64/\varepsilon^{\frac{2}{1-\theta}}$.

4.2 Coding against $1 - \varepsilon$ deletions

In this section, we construct codes for the high-deletion regime. We will use a concatenated coding approach, with an enlarged alphabet to help us determine the location of inner codewords. By choosing the parameters carefully, we are able to correct a large fraction of deletions. More precisely, we have the following theorem.

Theorem 4.7. Let $1/2 > \varepsilon > 0$. There is an explicit code of rate $\Omega(\varepsilon^2)$ and alphabet size $\mathrm{poly}(1/\varepsilon)$ which can be corrected from a $1 - \varepsilon$ fraction of worst-case deletions.

Moreover, this code can be constructed, encoded, and decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$, where N is the block length of the code.

We first define the code. Theorem 4.7 is then a direct corollary of Lemmas 4.8 and 4.9.

The code: Our code will be over the alphabet $\{0, 1, \ldots, D-1\} \times [k]$, where $D = 8/\varepsilon$ and $k = O(1/\varepsilon^3)$.

We first define a code C' over the alphabet $[k]$ by concatenating a Reed-Solomon code with an inner code over $[k]$ which can correct a slightly higher fraction of deletions.

More specifically, let $\mathbb{F}_q$ be a finite field. For any $n' \leq n \leq q$, the Reed-Solomon code of length $n \leq q$ and dimension n' is a subset of $\mathbb{F}_q^n$ which admits an efficient algorithm to uniquely decode from t errors and r erasures, provided $r + 2t < n - n'$ (see, for example, [WB86]).

In our construction, we will take $n = q = 2n'/\varepsilon$. We first encode our message to a codeword $c = (c_1, \ldots, c_n)$ of the Reed-Solomon code. For each i, we then encode the pair (i, c_i) using an inner code over some alphabet $[k]$ which can correct a $1 - \varepsilon/2$ fraction of deletions.

To obtain our final code C, we replace every symbol s in C' which encodes the ith RS coordinate by the pair $\big(i \pmod D, s\big) \in \{0, 1, \ldots, D-1\} \times [k]$. The first coordinate, $i \pmod D$, contains the location of the codeword symbol modulo D, and we will refer to it as a **header**.

In order to obtain the parameters stated in Theorem 4.7, we will instantiate the inner code using Corollary 4.6, setting $\theta = 1/3$. This gives an inner code $C_1 \colon [n] \times \mathbb{F}_q \to [k]^m$, where $m = 12 \log q/\varepsilon$ and $k = O(1/\varepsilon^3)$, which can correct a $1 - \varepsilon/2$ fraction of deletions.

Lemma 4.8. For an inner code of rate R_{in}, the rate of C is $\Omega(\varepsilon R_{\text{in}})$. In particular, the rate of C can be taken to be $\Omega(\varepsilon^2)$.

Proof. The rate of the outer Reed-Solomon code, labeled with indices, is at least $\varepsilon/4$. Finally, the alphabet increase in transforming C' to C decreases the rate by a factor of $\frac{\log(k)}{\log(Dk)} = \Omega(1)$.

By Corollary 4.6, the rate of the inner code can be taken to be $\Omega(\varepsilon)$.This gives us a final rate of $\Omega(\varepsilon^2)$. □

Lemma 4.9. Let the inner code have block length m and be decodable from a $1 - \varepsilon/2$ fraction of worst-case deletions in time $T(m)$. Then the concatenated code C can be decoded from a $1 - \varepsilon$ fraction of worst-case deletions in time $\mathrm{poly}(N) \cdot T(m)$, where N is the block length of C.

In particular, the concatenated code using the inner code of Corollary 4.6 can be decoded in time $N^{O(\mathrm{poly}\, 1/\varepsilon)}$.

Proof. We apply the following algorithm to decode C.

- We partition the received word into *blocks* as follows: The first block begins at the first coordinate, and each subsequent block begins at the next coordinate whose header differs from its predecessor. This takes time $\mathrm{poly}(N)$.

- We begin with an empty set L.

  For each block which is of length between $\varepsilon m/2$ and m, we remove the headers by replacing each symbol (a, b) with the second coordinate b. We then apply the decoder from Corollary 4.6 to the block. If this succeeds, outputting a pair (i, r_i), then we add (i, r_i) to L. This takes time $\mathrm{poly}(N) \cdot T(m)$.

- If for any i, L contains multiple pairs with first coordinate i, we remove all such pairs from L. L thus contains at most one pair (i, r_i) for each index i. We apply the Reed-Solomon decoding algorithm to the string r whose ith coordinate is r_i if $(i, r_i) \in L$ and erased otherwise. This takes time $\mathrm{poly}(N)$.

Analysis: For any i, we will decode a correct coordinate $\big(i, c_i\big)$ if there is a block of length at least $\varepsilon m/2$ which is a subsequence of $C_1(i, c_i)$. (Here and in what follows we abuse notation by disregarding headers on codeword symbols.)

Thus, the Reed-Solomon decoder will receive the correct value of the ith coordinate unless one of the following occurs:

1. (Erasure) The adversary deletes a $\geq 1 - \varepsilon/2$ fraction of $C_1(i, c_i)$.

2. (Merge) The block containing (part of) $C_1(i, c_i)$ also contains symbols from other codewords of C_1, because the adversary has erased the codewords separating $C_1(i, c_i)$ from its neighbors with the same header.

3. (Conflict) Another block decodes to (i, r) for some r. Note that an erasure cannot cause a coordinate to decode incorrectly, so a conflict can only occur from a merge.

We would now like to bound the number of errors and erasures the adversary can cause.

- If the adversary causes an erasure without causing a merge, this requires at least $(1-\varepsilon/2)m$ deletions within the block which is erased, and no other block is affected.

- If the adversary merges t inner codewords with the same label, this requires at least $(t-1)(D-1)m$ deletions, of the intervening codewords with different labels. The merge causes the fully deleted inner codewords to be erased, and causes the t merged codewords to resolve into at most one (possibly incorrect) value. This value, if incorrect, could also cause one conflict.

  In summary, in order to cause one error and $r \leq (t-1)D+2$ erasures, the adversary must introduce at least $(t-1)(D-1)m \geq (2+r)(1-\varepsilon/2)m$ deletions.

In particular, if the adversary causes s errors and r_1 erasures by merging, and r_2 erasures without merging, this requires at least

$$\geq (2s+r_1)(1-\varepsilon/2)m + r_2(1-\varepsilon/2)m = (2s+r)(1-\varepsilon/2)m$$

deletions. Thus, when the adversary deletes at most a $(1-\varepsilon)$ fraction of codeword symbols, we have that $2s+r$ is at most $(1-\varepsilon)mn/(1-\varepsilon/2)m < n(1-\varepsilon/2)$. Recalling that the Reed-Solomon decoder in the final step will succeed as long as $2s+r < n(1-\varepsilon/2)$, we conclude that the decoding algorithm will output the correct message. □

Remark (Improving the encoding and decoding complexity). Our decoding algorithm requires only that the inner code C_1 be correctable from a $1-\varepsilon/2$ fraction of deletions. By using the concatenated code of Theorem 4.7 as the inner code in our construction (that is, with two levels of concatenation), we can reduce the time complexity significantly, at the cost of a polynomial reduction in other parameters of the code. This is summarized in the following theorem.

Theorem 4.10. Let $1/2 > \varepsilon > 0$. There is an explicit code of rate $\Omega(\varepsilon^3)$ and alphabet size $\mathrm{poly}(1/\varepsilon)$ which can be corrected from a $1-\varepsilon$ fraction of worst-case deletions. Moreover, this code can be constructed, encoded, and decoded in time $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$, where N is the block length of the code.

4.3 Binary codes against ε deletions

4.3.1 Construction overview

The goal in our constructions is to allow the decoder to approximately locate the boundaries between codewords of the inner code, in order to recover the symbols of the outer code. In the previous section, we were able to achieve this by augmenting the alphabet and letting each symbol encode some information about the block to which it belongs. In the binary case, we no longer have this luxury.

The basic idea of our code is to insert long runs of zeros, or "buffers," between adjacent inner codewords. The buffers are long enough that the adversary cannot destroy many of them. If we then choose the inner code to be dense (in the sense of Proposition 4.5), it is also difficult for a long interval in any codeword to be confused for a buffer. This approach optimizes the construction of [SZ99] described in Section 2.4.3, which uses an inner code of rate $1/2$ and thus has final rate bounded away from 1.

The balance of buffer length and inner codeword density seems to make buffered codes unsuited for high deletion fractions, and indeed our results only hold as the deletion fraction goes to zero.

4.3.2 Our construction

We now give the details of our construction. For simplicity, we will not optimize constants in the analysis.

Theorem 4.11. Let $\varepsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ which is decodable from an ε fraction of deletions with rate $1 - \tilde{O}(\sqrt{\varepsilon})$ in time $N^{\mathrm{poly}(1/\varepsilon)}$.

Moreover, C can be constructed and encoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

The code: We again use a concatenated construction with a Reed-Solomon code as the outer code, choosing one which can correct a $12\sqrt{\varepsilon}$ fraction of errors and erasures. For each i, we replace the ith coordinate c_i with the pair (i, c_i). In order to ensure that the rate stays high, we use a RS code over $\mathbb{F}_{q^h}$, with block length $n = q$, where we will take $h = 1/\varepsilon$.

The inner code will be a good binary deletion code C_1 of block length m correcting a $\delta = 40\sqrt{\varepsilon}$ fraction of deletions. We will also require the codewords of C_1 to be β-dense,

for $\beta = \delta/4$. Recall that a string of length m is β-dense if any interval of length βm contains at least $\beta m/10$ 1's. We will assume each codeword begins and ends with a 1.

Now, between each pair of adjacent inner codewords of C_1, we insert a ***buffer*** of δm zeros. This gives us our final code C.

In order to obtain the final parameters stated in Theorem 4.11, we will construct the inner code C_1 using Proposition 4.5. This gives a code of rate $1 - 2h(\delta) - o(1)$ satisfying the requirements of our construction.

Lemma 4.12. For an inner code of rate R_{in}, the rate of the concatenated code C is $R_{\text{in}} \cdot (1 - O(\sqrt{\varepsilon})$.

In particular, the rate of the concatenated code using Proposition 4.5 is $1 - \tilde{O}(\sqrt{\varepsilon}))$.

Proof. The rate of the outer (labeled) Reed-Solomon code is $(1 - 24\sqrt{\varepsilon}) \cdot \frac{h}{h+1}$. Finally, adding buffers reduces the rate by a factor of $\frac{1}{1+\delta}$.

Combining these with our choice of δ, we get that the rate of C is $R_i(1 - \tilde{O}(\sqrt{\varepsilon}))$.

The rate of the inner code C_1 can be taken to be $1 - 2h(\delta) - o(1)$, by Proposition 4.5, giving a final rate of $1 - \tilde{O}(\sqrt{\varepsilon})$. □

Lemma 4.13. Let the inner code have block length m and be decodable from a δ fraction of worst-case deletions in time $T(m)$. Then the concatenated code C can be decoded from a ε fraction of worst-case deletions in time $\text{poly}(N) \cdot T(m)$, where N is the block length of C.

In particular, the concatenated code with inner code constructed using Proposition 4.5 can be decoded in time $N^{O(\text{poly } 1/\varepsilon)}$.

The algorithm:

- The decoder first locates all runs of at least $\delta m/2$ contiguous zeroes in the received word. These runs ("buffers") are removed, dividing the codeword into blocks of contiguous symbols which we will call *decoding windows*. Any leading zeroes of the first decoding window and trailing zeroes of the last decoding window are also removed. This takes time $\text{poly}(N)$.

- We begin with an empty set L.

  For each decoding window, we apply the decoder from Proposition 4.5 to attempt to recover a pair (i, r_i). If we succeed, this pair is added to L. This takes time $\text{poly}(N) \cdot T(m)$.

- If for any i, L contains multiple pairs with first coordinate i, we remove all such pairs from L. L thus contains at most one pair (i, r_i) for each index i. We apply the Reed-Solomon decoding algorithm to the string r whose ith coordinate is r_i if $(i, r_i) \in L$ and erased otherwise, attempting to recover from a $12\sqrt{\varepsilon}$ fraction of errors and erasures. This takes time $\mathrm{poly}(N)$.

Analysis: Notice that if no deletions occur, the decoding windows will all be codewords of the inner code C_1, which will be correctly decoded. At a high level, we will show that the adversary cannot corrupt many of these decoding windows, even with an ε fraction of deletions.

We first show that the number of decoding windows considered by our algorithm is close to n, the number of windows if there are no deletions.

Lemma 4.14. If an ε fraction of deletions have occurred, then the number of decoding windows considered by our algorithm is between $(1 - 2\sqrt{\varepsilon})n$ and $(1 + 2\sqrt{\varepsilon})n$.

Proof. Recall that the adversary can cause at most $\varepsilon nm(1 + \delta) \leq 2\varepsilon nm$ deletions.

Upper bound: The adversary can increase the number of decoding windows only by creating new runs of $\delta m/2$ zeroes (that are not contained within a buffer). Such a new run must be contained entirely within an inner codeword $w \in C_1$. However, as w is $\delta/4$-dense, in order to create a run of zeroes of length $\delta m/2$, at least $\delta m/20 = 2\sqrt{\varepsilon}$ 1's must be deleted for each such run. In particular, at most $\sqrt{\varepsilon}n$ blocks can be added.

Lower bound: The adversary can decrease the number of decoding windows only by decreasing the number of buffers. He can achieve this either by removing a buffer, or by merging two buffers. Removing a buffer requires deleting $\delta m/2 = 20\sqrt{\varepsilon}m$ zeroes from the original buffer. Merging two buffers requires deleting all 1's in the inner codewords between them. As inner codewords are $\delta/4$-dense, this requires at least $\sqrt{\varepsilon}m$ deletions for each merged buffer. In particular, at most $2\sqrt{\varepsilon}n$ buffers can be removed. □

We now show that almost all of the decoding windows being considered are decoded correctly by the inner decoder.

Lemma 4.15. The number of decoding windows which are incorrectly decoded is at most $4\sqrt{\varepsilon}n$.

Proof. The inner decoder will succeed on each decoding window which is a subsequence of a valid inner codeword $w \in C_1$ of length at least $(1 - \delta)m$. This will happen unless:

1. The window is too short:
   (a) a subsequence of w has been marked as a (new) buffer, or
   (b) a ρ fraction of w has been marked as part of the adjacent buffers, combined with a $\delta - \rho$ fraction of deletions within w.
2. The window is not a subsequence of a valid inner codeword: the window contains buffer symbols and/or a subsequence of multiple inner codewords.

We first show that (1) holds for at most $3\sqrt{\varepsilon}n$ windows.

From the proof of Lemma 4.14, there can be at most $\sqrt{\varepsilon}n$ new buffers introduced, thus handling Case 1(a). In Case 1(b), if $\rho < \delta/2$, then there must be $\delta/2$ deletions within w. On the other hand, if $\rho \geq \delta/2$, one of two buffers adjacent to w must have absorbed at least $\delta m/4$ symbols of w, so as w is $\delta/4$-dense, this requires $\delta m/40 = \sqrt{\varepsilon}m$ deletions, so can occur in at most $2\sqrt{\varepsilon}n$ windows.

We also have that (2) holds for at most $\sqrt{\varepsilon}n$ windows, as at least $\delta m/2$ symbols must be deleted from a buffer in order to prevent the algorithm from marking it as a buffer. As in Lemma 4.14, this requires $20\sqrt{\varepsilon}$ deletions for each merged window, and so there are at most $\sqrt{\varepsilon n}$ windows satisfying case (2). □

We now have that the inner decoder outputs $(1-6\sqrt{\varepsilon})n$ correct values. After removing possible conflicts in the last step of the algorithm, we have at least $(1 - 12\sqrt{\varepsilon})n$ correct values, so that the Reed-Solomon decoder will succeed and output the correct message.

Remark (Improving the encoding and decoding efficiency). Our decoding algorithm succeeds as long as the inner code can correct a δ fraction of deletions, and consists of codewords which are $\delta/4$-dense. As in the high deletion case, the time complexity of Theorem 4.11 can be improved using a more efficient inner code, at the cost of a reduction in rate.

Because of the addition of buffers, the code of Theorem 4.11 may not be dense enough to use as an inner code. However, we can modify the construction to obtain a dense inner code (details can be found in Section 4.5). In particular, these modifications give us the following.

Theorem 4.16. Let $\varepsilon > 0$. There is an explicit binary code $C \subseteq \{0,1\}^N$ which is decodable from an ε fraction of deletions with rate $1 - \tilde{O}(\sqrt[4]{\varepsilon})$ in time $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$.

Moreover, C can be constructed and encoded in time $\mathrm{poly}(N) \cdot (\log N)^{\mathrm{poly}(1/\varepsilon)}$.

4.4 List-decoding binary deletion codes

The results of Section 4.3 show that we can have good explicit binary codes when the deletion fraction is low. In this section, we address the opposite regime, of high deletion fraction. As a first step, notice that in any reasonable model, including list-decoding, we cannot hope to efficiently decode from a $1/2$ deletion fraction with a polynomial list size and constant rate. With block length n and $n/2$ deletions, the adversary can ensure that what is received is either $n/2$ 1's or $n/2$ 0's.

Thus, for binary codes and $\varepsilon > 0$, we will consider the question of whether it is possible to list decode from a fraction $1/2 - \varepsilon$ of deletions.

Definition 4.17. We say that a code $C \subseteq \{0,1\}^m$ is list-decodable from a δ deletion fraction with list size L if every sequence of length $(1-\delta)m$ is a subsequence of at most L codewords. If this is the case, we will call C (δ, L) *list-decodable from deletions*.

Remark. Although the results of this section are proven in the setting of list-decoding, it is *not* known that we cannot have unique decoding of binary codes up to deletion fraction $1/2 - \varepsilon$.

4.4.1 List-decodable binary deletion codes (existential)

In this section, we show that good list-decodable codes exist. This construction will be the basis of our explicit construction of list-decodable binary codes. The proof appears in Section 4.5.

Theorem 4.18. Let $\delta, L > 0$. Let $C \subseteq \{0,1\}^m$ consist of 2^{Rm} independently, uniformly chosen strings, where $R \leq 1 - h(\delta) - 3/L$. Then C is $\big(\delta, L\big)$ list-decodable from deletions with probability at least $1 - 2^{-m}$.

Moreover, such a code can be constructed and decoded in time $2^{\mathrm{poly}(mL)}$.

In particular, when $\delta = 1/2 - \varepsilon$, we can construct and decode in time $2^{\mathrm{poly}(m/\varepsilon)}$ a code $C \subseteq \{0,1\}^m$ of rate $\Omega(\varepsilon^2)$ which is $\big(\delta, O(1/\varepsilon^2)\big)$ list-decodable from deletions.

4.4.2 List-decodable binary deletion codes (explicit)

We now use the existential construction of Theorem 4.18 to give an explicit construction of constant-rate list-decodable binary codes. Our code construction uses Parvaresh-Vardy codes ([PV05]) as outer codes, and an inner code constructed using Section 4.4.1.

The idea is to list-decode "enough" windows and then apply the list recovery algorithm of Theorem 4.20.

Theorem 4.19. Let $0 < \varepsilon < 1/2$. There is an explicit binary code $C \subseteq \{0,1\}^N$ of rate $\tilde{\Omega}(\varepsilon^3)$ which is list-decodable from a $1/2 - \varepsilon$ fraction of deletions with list size $(1/\varepsilon)^{O(\log\log\varepsilon)}$.

This code can be constructed, encoded, and list-decoded in time $N^{\mathrm{poly}(1/\varepsilon)}$.

We will appeal in our analysis to the following theorem, which can be found in [GR08b].

Theorem 4.20 ([GR08b], Corollary 5)**.** For all integers $s \geq 1$, for all prime powers r, every pair of integers $1 < K \leq N \leq q$, there is an explicit $\mathbb{F}_r$-linear map $E\colon \mathbb{F}_q^K \to \mathbb{F}_{q^s}^N$ whose image C' is a code satisfying:

- There is an algorithm which, given a collection of subsets $S_i \subseteq \mathbb{F}_{q^s}$ for $i \in [N]$ with $\sum_i |S_i| \leq N\ell$, runs in $\mathrm{poly}\big((rs)^s, q, \ell\big)$ time, and outputs a list of size $O\big((rs)^s N\ell/K\big)$ that includes precisely the set of codewords $(c_1, \ldots, c_N) \in C'$ that satisfy $c_i \in S_i$ for at least αN values of i, provided

$$\alpha > (s+1)(K/N)^{s/(s+1)}\ell^{1/(s+1)}.$$

The code: We set $s = O(\log 1/\varepsilon)$, $r = O(1)$, and $N = K\,\mathrm{poly}\big(\log(1/\varepsilon)\big)/\varepsilon$ in Theorem 4.20 in order to obtain a code $C' \subseteq \mathbb{F}_{q^s}^N$. We modify the code, replacing the ith coordinate c_i with the pair (i, c_i) for each i, in order to obtain a code C''. This latter step only reduces the rate by a constant factor.

Recall that we are trying to recover from a $1/2 - \varepsilon$ fraction of deletions. We use Theorem 4.18 to construct an inner code $C_1\colon [N] \times \mathbb{F}_q^s \to \{0,1\}^m$ of rate $\Omega(\varepsilon^2)$ which recovers from a $1/2 - \delta$ deletion fraction (where we will set $\delta = \varepsilon/4$). Our final code C is a concatenation of C'' with C_1, which has rate $\tilde{\Omega}(\varepsilon^3)$.

Theorem 4.21. C is list-decodable from a $1/2 - \varepsilon$ fraction of deletions in time $N^{\mathrm{poly}(1/\varepsilon)}$.

Proof. Our algorithm first defines a set of "decoding windows". These are intervals of length $(1/2 + \delta)m$ in the received codeword which start at positions $1 + t\delta m$ for $t = 0, 1, \ldots, N/\delta - (1/2 + \delta)/\delta$, in addition to one interval consisting of the last $(1/2 + \delta)m$ symbols in the received codeword.

We use the algorithm of Theorem 4.18 to list-decode each decoding window, and let $\mathcal{L}$ be the union of the lists for each window. Finally, we apply the algorithm of Theorem 4.20 to $\mathcal{L}$ to obtain a list containing the original message.

Correctness: Let $c = (c_1, \ldots, c_N)$ be the originally transmitted codeword of C'. If an inner codeword $C_1(i, c_i)$ has suffered fewer than a $1/2 - 2\delta$ fraction of deletions, then one of the decoding windows is a substring of $C_1(i, c_i)$, and $\mathcal{L}$ will contain the correct pair (i, c_i).

When $\delta = \varepsilon/4$, by a simple averaging argument, we have that an ε fraction of inner codewords have at most $1/2 - 2\delta$ fraction of positions deleted. For these inner codewords, $\mathcal{L}$ contains a correct decoding of the corresponding symbol of c.

In summary, we have list-decoded at most N/δ windows, with a list size of $O(1/\delta^2)$ each. We also have that an ε fraction of symbols in the outer codeword of C' is correct. Setting $\ell = O(1/\delta^3)$ in the algorithm of Theorem 4.20, we can take $\alpha = \varepsilon$. Theorem 4.20 then guarantees that the decoder will output a list of $\mathrm{poly}(1/\varepsilon)$ codewords, including the correct codeword c. □

4.5 Omitted proofs

Existential bounds on deletion codes

In this section, we give the omitted proofs of Sections 4.1 and 4.4.

Proof of Lemma 4.3. We will give a way to generate all strings s' containing s as a subsequence, and bound the number of possible outcomes. We do this by considering the lexicographically first occurrence of s in t.

First choose ℓ locations $n_1 < \cdots < n_\ell$ in $[m]$, which will be the locations of the ℓ symbols of s. If the ith symbol of s is a, we allow all symbols between locations n_{i-1} and n_i to take any value but a. This ensures that the locations n_i are the *earliest* occurrence of s as a subsequence. The rest of the symbols after n_ℓ are filled in arbitrarily.

It is clear that this process generates any string having s as a subsequence, so we will bound the number of ways this can happen. Fix $n_\ell = t$. There are

- $\binom{t-1}{\ell-1}$ ways to choose $n_1, \ldots, n_{\ell-1}$,
- $(k-1)^{t-\ell}$ ways to fill in symbols between the n_i's,
- and k^{m-t} ways to fill in the last $m - t$ symbols.

Summing over all possible values of t, the total number of strings with s as a subsequence is at most

$$\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} k^{m-t}(k-1)^{t-\ell}.$$

As $\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} = \binom{m}{\ell}$, the claimed bound follows.

When $\ell > m/k$, the term $\binom{t-1}{\ell-1}k^{m-t}(k-1)^{t-\ell}$ increases with t, so the sum is at most

$$\delta m \cdot \binom{m-1}{\ell-1}(k-1)^{m-\ell},$$

giving us our bound for $k = 2$.

□

Proof of Theorem 4.4. We construct such a code using a greedy algorithm. We begin with an arbitrary string in $[k]^m$, and then iteratively add strings whose LCS with all previously chosen strings has length less than $(1-\delta)m$. The LCS of two length m strings can be computed in time $\mathrm{poly}(m)$, so this takes time $k^{O(m)}$.

It remains to show that we can choose k^{Rm} strings.

For a fixed string $u \in [k]^m$, it has at most $\binom{m}{(1-\delta)m}$ subsequences of length $(1-\delta)m$, so by Lemma 4.3, the number of strings whose LCS with u has length at least $(1-\delta)m$, and which therefore cannot be chosen, is at most

$$\binom{m}{(1-\delta)m}^2 k^{\delta m}.$$

Thus if the target rate is R, we will succeed if

$$\binom{m}{\delta m}^2 k^{\delta m} \cdot k^{Rm} \leq k^m. \qquad (*)$$

It suffices to have

$$2mh(\delta) + \delta m \log k + Rm \log k \leq m \log k.$$

Setting $R = 1 - \delta - \gamma$, we have

$$2h(\delta) + (1-\gamma)\log k \leq \log k \Leftrightarrow 2h(\delta) \leq \gamma \log k,$$

so we can choose k^{Rm} strings as long as the alphabet size k satisfies

$$k \geq 2^{2h(\delta)/\gamma}.$$

In the case of $k = 2$, we may use the tighter estimate from Lemma 4.3 in Equation (*) to obtain the claimed bound. □

Proof of Proposition 4.5. The greedy algorithm of Theorem 4.4 applies, but now we must choose strings from the set of β-dense strings. We first bound the number of strings which are *not* β-dense. The number of strings of length βm with less than $\beta m/10$ 1's is

$$\sum_{j=0}^{\beta m/10-1} \binom{\beta m}{j} \leq 2^{h(1/10)\beta m}.$$

Since there are at most m intervals of length βm in a string, the probability that a randomly chosen string of length m is not β-dense is at most

$$m \cdot \frac{2^{h(1/10)\beta m}}{2^{\beta m}} \leq 2^{-\Omega(\beta m)}.$$

The algorithm of Theorem 4.4 then succeeds if

$$\binom{m}{\delta m}^2 \cdot \delta m \cdot 2^{Rm} \leq 2^m\big(1 - 2^{-\Omega(\beta m)}\big),$$

or $R \leq 1 - 2h(\delta) - O(\log(\delta m)/m) - 2^{-\Omega(\beta m)}/m$. □

Proof of Theorem 4.18. By Lemma 4.3, the probability that a set of L independent, uniform strings all share a common substring of length ℓ is at most

$$2^\ell \cdot \left(\sum_{t=\ell}^{m} \binom{t-1}{\ell-1} 2^{-t}\right)^L \leq 2^\ell \left[m^L \cdot 2^{-mL} \cdot \binom{m-1}{\ell-1}^L\right].$$

For a random code C of rate R, we union bound over all possible subsets of L codewords to upper bound the probability that C is *not* (δ, L) list-decodable from deletions.

$$\Pr[C \text{ fails}] < 2^{RmL} \cdot 2^\ell \cdot 2^{L \log m} \cdot 2^{-mL} \cdot 2^{Lmh(1-\delta)}.$$

This is at most 2^{-m}, provided

$$R \leq 1 - h(\delta) - \frac{2-\delta}{L} - \frac{\log m}{m},$$

which holds for our choice of R.

When $\delta = 1/2 - \varepsilon$, we can set $R = \Omega(\varepsilon^2)$ to see that

$$L > \frac{3/2 + \varepsilon}{2\varepsilon^2/\ln 2 - R - O(\varepsilon^3)}$$

so we can take L to be $O(1/\varepsilon^2)$.

Similarly to Theorem 4.4, this argument shows that we can construct a $\left(\delta, O(1/\varepsilon^2)\right)$ list-decodable code using a greedy algorithm, which successively adds strings who do not share a common subsequence of length ℓ with $L-1$ previously chosen strings. □

Obtaining dense inner codes

In this section, we show how the code construction of Section 4.3 can be modified to obtain a code which is also *dense*, allowing it to be used as an inner code. More precisely, we will show:

Proposition 4.22. For every block length n and $\delta \in (0,1)$, there is a binary code $C \subseteq \{0,1\}^n$ of rate $1 - \tilde{O}(\sqrt{\delta})$ which is decodable from an δ fraction of deletions in time $n^{\mathrm{poly}(1/\delta)}$.

Moreover, C can be constructed and encoded in time $n^{\mathrm{poly}(1/\delta)}$, and consists of strings which are $\delta/4$-dense, in the sense of Proposition 4.5.

Proof. The proof of Proposition 4.5 works, with a lower rate guarantee, for strings of length m which have at least $\beta m/5$ 1's in every interval of length βm. This allows us to construct, encode, and decode in time $2^{O(m)}$ a code of rate $1 - 2h(\delta) - o(1)$ which can correct a δ fraction of deletions, and which consists of strings satisfying the stronger density property.

Using this as the inner code in the construction of Section 4.3, for any $\delta \in (0,1)$ we obtain a binary code of block length n and rate $1 - \tilde{O}(\delta)$ which is decodable from a δ fraction of deletions in time $n^{\mathrm{poly}(1/\delta)}$.

It remains to show that this code is $\delta/4$-dense, in the sense of Proposition 4.5. Recall that each codeword is the concatenation of $\log n$ inner codewords of block length m, separated by buffers of length δm.

We need to show that any interval of length $\delta N/4$ contains at least $\delta N/40$ 1's. By construction, each inner codeword has Hamming weight at least $m/5$.

An interval of length $\delta n/4$ consists of at least $(\delta n/4)/(m(1+\delta)) - 2$ inner codewords with buffers. As each inner codeword has at least $m/5$ 1's, this gives a total of $\frac{\delta n}{20(1+\delta)} - 2m/5$ 1's. Recalling that $m \approx \log n$, this is at least $\delta n/40$ for large enough N. $\square$

Chapter 5

Rank-metric and subspace codes

In which linear network coding debuts • The operator channel and rank-metric appear • Reed-Solomon goes linearized

In this chapter, we present motivation and basic background for the two closely-related models of rank-metric and subspace codes, as a prelude to our construction of high-rate list-decodable codes in Chapter 6. These models are interesting both for their unusual notion of errors, and because the codes themselves are so different. For example, although *every* code for the Hamming metric is list-decodable past half the distance, this is not true in the rank-metric case.

Some of the results on subspace codes in this chapter appeared in joint work with Guruswami and Narayanan in [GNW12].

5.1 Linear network coding

We now present an abbreviated overview of noncoherent linear network coding, which motivates the definition of the operator channel in Section 5.2.

We model a network as a directed flow graph with some number of sources and sinks. Each source will have some message (packet) in $\mathbb{F}_h^t$ for some field $\mathbb{F}_h$, and the goal is for each sink to receive a copy of every message (the *multicast* setting). The edges only have the capacity to transmit one packet's worth of information. It is easy to see that this is not always possible, for example if the minimum cut separating any sink from the sources is smaller than the number of sources. This is not always tight, however, if the

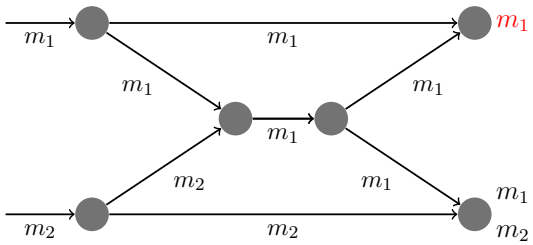

Figure 5.1: The butterfly network ([ACLY00]). When the intermediate nodes must pick a packet to forward, one of the sinks receives two copies of the same packet and so only one packet is successfully transmitted. On the other hand, the minimum cut in this graph has size 2.

intermediate nodes in the network can only forward packets. This fact can be seen in the famous butterfly network of [ACLY00] (Figure 5.1), where only one packet can be transmitted, even though the network's minimum cut has size 2.

On the other hand, the same authors note that if we allow the intermediate nodes to perform basic operations on the messages (in this case, addition), then it is possible for the sinks to recover both packets (Figure 5.1).

It turns out that not much more power is required in general: if we allow intermediate nodes to transmit *linear combinations* of packets, then we can achieve the minimum cut capacity of the network, and indeed this works even if we send *random* linear combinations, with probability of success increasing as we increase the size of the field $\mathbb{F}_h$ ([HKM$^+$03]).

Random linear network coding turns out to be an extremely useful solution to this problem. It imposes very few requirements on the nodes in the network, as they only have to apply linear operations, and it's easy to adapt if the network topology changes. In fact, if each message packet is given a "header" (say, the ith packet appends the ith unit vector e_i to its packet), then the packets will record which linear operations have been performed, and the decoder can recover the original packets without needing to know the exact behavior of the network.

One drawback of this approach, however, is that it is not resilient to errors in the network. For example, if a package is forwarded incorrectly, then as it propagates through the network, it could get added to *every* packet which reaches each sink, preventing the receivers from recovering the information. Another concern is that random network coding

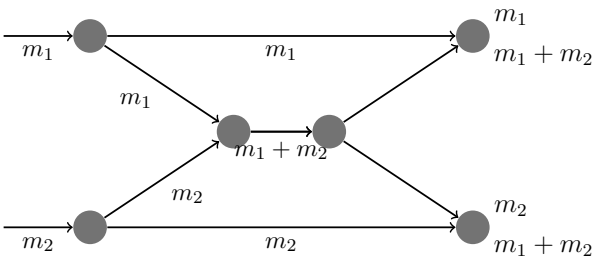

Figure 5.2: The butterfly network revisited. If intermediate nodes can *add* packets, then the sinks each receive one packet and the sum of two packets, which they can use to recover the missing packet.

only works with high probability; if it fails, not all packets may make it to the sinks.

As we did in the case of symbol errors in strings, we would like to add redundancy to our packets in some way, so that we can ensure recovery from errors in the network. More specifically, we will encode our information packets by a slightly larger set of (redundant) packets, with the guarantee that the information packets are decodable after passing through the faulty network. This idea is captured by the notion of *subspace codes*.

5.2 Subspace codes and the operator channel

Let us return to the butterfly network of Figure 5.1. Notice that the top sink, who was hoping to receive the two packets m_1 and m_2, has in fact received m_1 and $m_1 + m_2$. The key observation is that, either through foreknowledge of the network or through the use of headers, receiving m_1 and $m_1 + m_2$ is *just as good* as receiving m_1 and m_2. The reason for this is that the two pairs of vectors are bases for the same vector space, $\mathrm{span}(m_1, m_2)$. That is, given one basis for the space, it is easy enough to recover the original basis of interest, so we are only interested in making sure that the correct vector space is transmitted.

With this in mind, we now define subspace codes.

For a vector space W, let $\mathcal{P}(W)$ denote the set of all subspaces of W, and $\mathcal{P}_n(W)$ the set of all n-dimensional subspaces of W (often called the Grassmannian).

Definition 5.1. A **subspace code** C is a subset of $\mathcal{P}_n(\mathbb{F}_h^t)$ for some n. We define the *rate*

of a subspace code to be

$$R(C) = \frac{\log_h |C|}{nt}.$$

The distance measure we will use is the number of basis elements which must be removed or added to transform one vector space into another, defined formally below. It is easy to check (see [KK08]) that this is in fact a metric on $\mathcal{P}(W)$.

Definition 5.2. The **subspace distance** between two subspaces U and V is

$$d_S(U, V) := \dim(U) + \dim(V) - 2\dim(U \cap V).$$

The **minimum distance** of a subspace code C is $d_S(C) := \min_{U \neq V \in c} d_S(U, V)$, and the **relative distance** of C is $\frac{d_S(C)}{2n}$.

To model the kinds of errors we expect in a linear network, we now define the operator channel, which was first introduced in [KK08].

Definition 5.3. An *operator channel* C associated with the *ambient space* W is a channel with input and output alphabet $\mathcal{P}(W)$. The channel input V and output U are related by

$$U = \mathcal{H}_k(V) + E,$$

where $k = \dim(U \cap V)$, E is an error subspace (without loss of generality, E may be taken such that $E \cap V = \{0\}$), and $\mathcal{H}_k(V)$ is an operator returning an arbitrary k-dimensional subspace of V.

In transforming V to U, we say that operator channel commits $\mu = \dim(V) - k$ *deletions* and $\rho = \dim(E)$ *insertions*.

Remark. Deletions and insertions were called erasures and errors, respectively, in [KK08]. We have chosen different terminology which we feel better reflects the nature of the changes introduced.

Note that a deletion corresponds to the removal of some basis vector from the input space V, and an insertion introduces a new basis vector.

This model makes no assumptions on the structure of the network being used, and any behavior arising from linear combinations at intermediate nodes is captured in this notion of error. The relationship between the operator channel and subspace codes is shown in the following theorem.

Theorem 5.4 ([KK08])**.** Let C be a subspace code of minimum distance d. Let $V \in C$ be transmitted, and let

$$U = \mathcal{H}_k(V) + E$$

be received, where $\dim(E) = \mu$. Let ρ be the number of deletions induced by the channel. If

$$2(\mu + \rho) < d,$$

then V can be uniquely decoded from the received subspace U.

5.2.1 The Kötter-Kschischang code

In [KK08], the authors define a subspace code, which we will call the Kötter-Kschischang (or KK) code, which can be thought of as a "linearized" variant of Reed-Solomon codes.

Definition 5.5. A **linearized polynomial** over $\mathbb{F}_{h^m}$ is a polynomial f of the form

$$f(X) = \sum_{i=0}^{k} f_i X^{h^i},$$

where $f_i \in \mathbb{F}_{h^m}$. The integer k is the h-degree of f.

These polynomials are called linearized polynomials because they are linear functions over the base field $\mathbb{F}_h$; that is, if f is a linearized polynomial, then $f(aX+bY) = af(X)+bf(Y)$ for any $a, b \in \mathbb{F}_h$.

As with Reed-Solomon codes, the KK code is defined using polynomial evaluation, but we evaluate *linearized* polynomials of low h-degree, rather than arbitrary polynomials of low (standard) degree.

Definition 5.6. Let $\mathbb{F}_{h^t}$ be an extension of $\mathbb{F}_h$, and let $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}_h^t$ be linearly independent over $\mathbb{F}_h$. The KK code $\mathrm{KK}[n, k, t]$ encodes a linearized polynomial $f(X) \in \mathbb{F}_{h^t}[X]$ of h-degree at most $k-1$ by

$$f(X) \mapsto \mathrm{span}\{(\alpha_i, f(\alpha_i))\}_{i=1}^{n}.$$

Lemma 5.7. When $k < n$, this code has distance $2(n-k+1)$ and rate

$$\frac{\log_h q^{mk}}{n(n+t)} = \frac{k}{n}\left(\frac{1}{1+n/t}\right) \approx \frac{k}{n} \quad \text{(when } n \ll t\text{)}. \tag{5.1}$$

Proof. The rate computation is immediate from the definition. To see the distance property, we will show that two distinct h-linearized polynomials f and g of h-degree d can agree on at most d linearly independent points.

Suppose that f and g agree on $d+1$ linearly independent points. Recall that f and g are linear over $\mathbb{F}_h$, so they must also agree on the $d+1$-dimensional subspace spanned by these linearly independent points. In particular, they agree on at least h^{d+1} points. As f and g have (standard) degree at most h^d, they must be the same polynomial. □

By using a suitable adaptation of a Reed-Solomon decoder, the authors of [KK08] are able to decode KK codes from the optimal number of errors, as summarized in this theorem.

Theorem 5.8 ([KK08])**.** The Kötter-Kschischang code $\mathrm{KK}[n, k, t]$ can be uniquely decoded in polynomial time from ρ deletions and μ insertions, provided that

$$\rho + \mu < n - k + 1.$$

5.3 Rank-metric codes

Rank-metric codes are a kind of error-correcting code in which the distance measure is not the Hamming metric, but the *rank-metric*. These codes are of interest in cryptography, and as finite-field analogues of space-time codes. The fact of most interest to us is that any rank-metric code can be "lifted" into a subspace code, essentially by adding "headers" to its rows (see [SKK08]).

Definition 5.9. A **rank-metric code** is a set of matrices $M \in \mathbb{F}_h^{n \times t}$ over a finite field $\mathbb{F}_h$ for fixed n, t.

The rate of a rank-metric code is $\log_h |\mathcal{C}|/(nt)$, and the distance measure between two codewords is the rank over $\mathbb{F}_h$ of their difference; that is, $\mathrm{dist}(M_1, M_2) = \mathrm{rank}_{\mathbb{F}_h}(M_1 - M_2)$.

Definition 5.10. We say that a rank-metric code $\mathcal{C}$ can be decoded from e **rank errors** if any codeword $M \in \mathcal{C}$ can be recovered from $M + E$ whenever $E \in \mathbb{F}_h^{n \times t}$ has rank at most e.

Note that this model of error, like the subspace distance, is very different from the standard model using Hamming distance. For example, if the error matrix E is the all-1's

matrix, only one rank error has occurred, but for any codeword M, $M + E$ and M differ in *every* coordinate.

One rank-metric code which has been studied extensively is an adaptation of the Reed-Solomon code for the rank distance, known as the *Gabidulin code* ([Gab85]). Similar to the Kötter-Kschischang codes, the Gabidulin code is defined by evaluating linearized polynomials with low h-degree. In fact, the KK code can be thought of as a lift of a Gabidulin code, in the sense of [SKK08].

Definition 5.11. A **Gabidulin code** (denoted $\mathcal{C}_G(h; n, t, k)$) encodes h-linearized polynomials over $\mathbb{F}_{h^t}$ of h-degree less than k by

$$\big(f(\alpha_1), \ldots, f(\alpha_n)\big)^T,$$

where the $\alpha_i \in \mathbb{F}_{h^t}$ are linearly independent over $\mathbb{F}_h$, and $f(\alpha_j)$ is thought of as a column vector in $\mathbb{F}_h^t$ under a fixed basis of $\mathbb{F}_{h^t}$ over $\mathbb{F}_h$.

This is a rank-metric code of rate k/n and minimum distance $n - k + 1$.

The distance property can be shown for Gabidulin codes in the same way that we showed it for KK codes.

Many of the algorithms for uniquely decoding Reed-Solomon codes can be adapted to decode Gabidulin codes (see, for example, [Loi05]). In particular, like KK codes, Gabidulin codes can be decoded up to half of their minimum distance in polynomial time.

5.4 List-decoding subspace and rank-metric codes

Although Reed-Solomon codes adapt nicely to the settings of subspace and rank-metric codes, it turns out that the case of list-decoding is quite different. As mentioned at the beginning of this chapter, there is no analogue of the Johnson bound for rank-metric codes. In fact, not even all Gabidulin codes can be list-decoded from a number of errors which is more than half the minimum distance.

Theorem 5.12 ([RWZ15])**.** Let g, s, and n be integers such that $g \geq 2$ and $sg \mid n$, and let C be a Gabidulin code over $\mathbb{F}_q^n$ with $d = 2sg$ and evaluation points $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q^n$. Then there is an explicit word $c_R \in \mathbb{F}_{q^n}^n \setminus C$ such that the number of codewords of C within distance $\tau = \lfloor \frac{d-1}{2} \rfloor + 1$ of c_R is at least

$$\frac{q^n - 1}{q^{sg} - 1}.$$

5.4.1 List-decodability of random codes

Although we saw in the previous theorem that one of the most natural explicit code constructions cannot be list-decoded, it still turns out that there exist good list-decodable codes in these models. Indeed, as in the Hamming case, random codes will work with high probability.

We show this result first in the subspace codes case. This theorem first appeared in joint work with Guruswami and Narayanan in [GNW12].

Theorem 5.13. For every $L \geq 1$, for all large enough integers t, n with $n \leq t/2$, a random subspace code $\mathcal{C} \subseteq \mathcal{P}_n(\mathbb{F}_q^t)$ of rate R (obtained by picking q^{Rnt} subspaces uniformly and independently at random), is list decodable with high probability from μ deletions and ρ insertions with list size L, provided

$$\frac{\rho}{n} + (L+1)\frac{\mu}{n} < L - (L+1)R\,.$$

(The ratios ρ/n and μ/n are the fraction of insertions and deletions, respectively.)

Proof. Fix a subspace T of dimension d, where $n-\mu \leq d \leq n+\rho$ (the range of dimensions possible when there are up to t insertions and r deletions). Fix a subset $\mathcal{S}$ of $(L+1)$ codewords from the random code $\mathcal{C}$. The probability that each subspace in $\mathcal{S}$ differs from T by at most ρ insertions and μ deletions is at most

$$\sum_{\rho'=0}^{\rho}\sum_{\mu'=0}^{\mu}(q^d)^{n-\mu'}q^{(n-t)(n-\mu')} \leq O(\rho\mu)q^{dn+n^2}q^{-t(n-\mu)}\,.$$

Further this event is independent for different codewords in $\mathcal{S}$ by the random choice of $\mathcal{C}$. By a union bound over all choices of T and $\mathcal{S}$, the probability that $\mathcal{C}$ fails to be list-decodable is at most

$$q^{Rtn(L+1)}q^{t(n+\rho)}\left(q^{O((n+\rho)^2)}q^{-t(n-\mu)}\right)^{L+1}\,.$$

For large enough t, this quantity is $q^{-\Omega(t)}$ provided $R(L+1)+(n+\rho) < (n-\mu)(L+1)$, or equivalently if $\frac{\rho}{n} + (L+1)\frac{\mu}{n} < L - (L+1)R$. □

It was shown in [Din15] that random codes achieve the best possible list-decoding radius for the rank-metric case as well.

Theorem 5.14 ([Din15])**.** For every $0 < \varepsilon < 1$ and $0 < R < 1$, let n and t be sufficiently large positive integers satisfying $n/t \leq \varepsilon$. Then with high probability, a random rank-metric code $C \subseteq \mathbb{F}_{h^t}^n$ with rate R is list-decodable from a $1 - R - \varepsilon$ fraction of rank errors with a list size of $O(1/\varepsilon)$.

In particular, good list-decodable codes *exist*, and the challenge is to construct them efficiently.

5.4.2 Previous list-decodable constructions

List-decoding of a folded variant of the Kötter-Kschischang code was considered independently in [GNW12] and [MV12] (the latter also considered a folded Gabidulin code). However, both of these papers could only guarantee a polynomial list size when the rate of the code was polynomially small.

Let us briefly sketch our construction of list-decodable subspace codes from [GNW12].

We will write $X^{[i]}$ for the function X^{h^i}. Let γ generate a normal basis for $\mathbb{F}_{h^t}$ (that is, the set $\{1, \gamma, \gamma^{[1]}, \ldots, \gamma^{[t-1]}\}$ forms a basis).

Definition 5.15 (Linearized FRS codes)**.** Let $\alpha_i \in \mathbb{F}_{h^t}$ for $i = 1, \ldots, n$ be linearly independent over $\mathbb{F}_h$. The **linearized folded Reed-Solomon code** $\mathrm{lFRS}^{n,t,s}$ encodes $f \in \mathbb{F}_{h^t}[X]$ by

$$V = \langle \{(\alpha_i, f(\gamma\alpha_i), f(\gamma^{[1]}\alpha_i), \ldots, f(\gamma^{[s-1]}\alpha_i)\}_{i=1}^n \rangle$$

for some parameter s.

By adapting the linear-algebraic decoding algorithm used in Chapter 3, we were able to give an efficient list-decoding algorithm for the low-rate *subcode* of the linearized folded Reed-Solomon code where the polynomial coefficients come from the base field $\mathbb{F}_h$. Although our algorithm also applied to list-decode in the case of general $f \in \mathbb{F}_{h^t}[X]$, the output list was a subspace over $\mathbb{F}_{h^t}$, which is a field of exponential size. This difficulty also arose in [MV12], as they used the same algorithm.

Note that this sort of dependence is necessary. To see this, consider the case $\mu = 0$ of no deletions. Then if $g_1, \ldots, g_{n+1}$ are linearly independent (as coefficient vectors) and agree with the received subspace, any combination $\sum \lambda_i g_i$ with $\sum \lambda_i = 1$ also agrees with the received subspace, giving a list size of $(h^t)^n$.

Thus, constructing an explicit rank-metric or subspace code which achieves both positive rate and polynomial list size has been a challenge. In the next chapter, we will see

a construction of list-decodable subcodes of the Gabidulin and KK codes which not only meet the goals of positive rate and polynomial list size, but also have the optimal trade-off between rate and correctable error fraction for any desired rate.

Chapter 6

List-decodable rank-metric codes

In which we list-decode Gabidulin codes • Periodic subspaces come in handy • Linear subcodes reduce list size

In this chapter, we return to the linear-algebraic decoding of Chapter 3 and adapt that technique to the case of rank-metric codes. Specifically, we give an algorithm to decode Gabidulin codes whose output list is potentially exponential in size. We then show how to construct subcodes which reduce this list size to polynomial, using a refinement of the subspace-evasive sets which we used in Chapter 3. We also show how to apply these techniques to subspace codes and so-called low-order folded Reed-Solomon codes.

Recall that the linear-algebraic method is broken up into two steps: interpolation, where we find a condition satisfied by nearby messages, and root-finding, where we solve for these messages. In most cases when applying this method, there is a natural choice for the interpolated polynomial, and the difficulty lies in the root-finding step, where we must ensure that the output list is short. In Chapter 3, we bounded the list size by showing that the dimension of the solution space was constant.

As alluded to in Chapter 5, the difficulty in applying this type of analysis to Gabidulin codes (and rank-metric codes in general), is that for a polynomial of degree $k-1$, we require $n \geq k$ evaluation points which are *linearly independent* over the base field $\mathbb{F}_h$. This means that we are working in a field of size at least h^n, so any nontrivial subspace over this field has exponential size in the codeword dimensions. That is, merely bounding the dimension of the output list is not sufficient to give a polynomial list size.

To improve the list size guarantee, we prove that the list output by the linear-algebraic decoder has a more rigid structure than simply being a subspace of some dimension, and

Table 6.1: Parameters used in this chapter

Parameter	Meaning	Value
R	target code rate	constant in $(0,1)$
ε	gap to capacity (error rate $1-R-\varepsilon$)	constant in $(0,1)$
$\mathbb{F}_h$	base field (codes will be $\mathbb{F}_h$-linear)	h arbitrary
k	degree of polynomial being encoded	growing
$\mathbb{F}_{h^n} := \mathbb{F}_q$	extension spanned by evaluation points	$n = \Omega(k)$, growing
$\mathbb{F}_{q^m} = \mathbb{F}_{h^t}$	field of polynomial coefficients	$m \approx 1/\varepsilon^2$ is constant
s	folding parameter of algorithm (see Section 6.1)	$s \approx 1/\varepsilon$ is constant
$h^{O(ms/\varepsilon)}$	final output list size	$h^{\mathrm{poly}(1/\varepsilon)}$ (polynomial)

we use this fact to construct pseudorandom subsets which evade this structure, similar to the way we used subspace-evasive sets in Chapter 3.

As there is a lot of notation in this chapter, we summarize the main parameters in Table 6.1. We will work with polynomial evaluation codes over $\mathbb{F}_{h^t}$ where the evaluation points come from a subfield $\mathbb{F}_{h^n}$, and our goal is to list-decode rate R codes from a $1-R-\varepsilon$ error fraction.

The results in this chapter appear in joint work with Guruswami and Xing in [GWX15].

6.1 List-decoding Gabidulin codes

In this section, we show how to apply linear-algebraic list-decoding to the special case of Gabidulin codes over $\mathbb{F}_{q^m}$ where the evaluation points span the *subfield* $\mathbb{F}_q$ of $\mathbb{F}_{q^m}$. This restriction helps us to simulate the algebraic folding used in Chapter 3: if α is in $\mathbb{F}_q$, then we can use $f(\alpha)$ to compute $f^\sigma(\alpha) = f(\alpha)^q$, where σ is the Frobenius automorphism of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ (precise details below).

Although we will not be able to prove a polynomial list size bound for Gabidulin codes themselves, our goal is to show that the list output by our algorithm is highly structured, a fact we will then use in the next section.

Let us first introduce the notion of *periodic subspaces*, the structured sets which we will be using. Below, for a string $\mathbf{x} = (x_1, x_2, \ldots, x_\ell)$, we denote by $\mathrm{proj}_{[a,b]}(\mathbf{x})$ the

substring $(x_a, x_{a+1}, \ldots, x_b)$.

Definition 6.1 (Periodic subspaces)**.** For positive integers s, m, k and $\kappa := mk$, an affine subspace $H \subset \mathbb{F}_q^\kappa$ is said to be (s, m, k)-**periodic** if there exists a subspace $W \subseteq \mathbb{F}_q^m$ of dimension at most s such that for every $j = 1, 2, \ldots, k$, and every prefix $\mathbf{a} \in \mathbb{F}_q^{(j-1)m}$, the projected affine subspace of $\mathbb{F}_q^m$ defined by

$$\{\mathrm{proj}_{[(j-1)m+1,jm]}(\mathbf{x}) \mid \mathbf{x} \in H \text{ and } \mathrm{proj}_{[1,(j-1)m]}(\mathbf{x}) = \mathbf{a}\}$$

is contained in an affine subspace of $\mathbb{F}_q^m$ given by $W + \mathbf{v_a}$ for some vector $\mathbf{v_a} \in \mathbb{F}_q^m$ dependent on $\mathbf{a}$.

Definition 6.2 (Representing periodic affine subspaces)**.** The canonical representation of an (r, Λ, b)-periodic subspace H consists of a matrix $B \in \mathbb{F}_q^{\Lambda \times \Lambda}$ such that $\ker(B)$ has dimension at most r, and vectors $a_i \in \mathbb{F}_q^\Lambda$ and matrices $A_{i,j} \in \mathbb{F}_q^{\Lambda \times \Lambda}$ for $1 \le i \le b$ and $1 \le j < i$, such that $\mathbf{x} \in H$ if and only if for every $i = 1, 2, \ldots, b$ the following holds:

$$a_i + \Big(\sum_{j=1}^{i-1} A_{i,j} \cdot \mathrm{proj}_{[(j-1)\Lambda+1,j\Lambda]}(\mathbf{x})\Big) + B \cdot \mathrm{proj}_{[(i-1)\Lambda+1,i\Lambda]}(\mathbf{x}) = 0\,.$$

Very loosely, a periodic subspace can be broken up into blocks of length m which "look like" shifts of the same low-dimensional subspace. We will give a list-decoding algorithm whose output is a periodic subspace.

Recall the Gabidulin codes from Definition 5.11. We will choose n evaluations points $\alpha_1, \alpha_2, \ldots, \alpha_n$ that are linearly independent over $\mathbb{F}_h$ from the subfield $\mathbb{F}_{h^n}$. Put $q = h^n$ and $m = \frac{t}{n}$, so that $\mathbb{F}_{h^n} = \mathbb{F}_q$ and $\mathbb{F}_{h^t} = \mathbb{F}_{q^m}$. In this case, we also denote $\mathcal{C}_G(h; n, t, k)$ by $\mathcal{C}_G(q; n, m, k)$. Suppose that a codeword $M_f = (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n))^T$ is transmitted and $Y = (y_1, y_2, \ldots, y_n)^T$ is received with at most e rank errors (note that we identify every row vector in Y with an element y_i in $\mathbb{F}_{h^t} = \mathbb{F}_{q^m}$).

We will need the following fact:

Lemma 6.3. Let $X, Y \in \mathbb{M}_{n\times t}(\mathbb{F}_h)$ with $\mathrm{rank}(X - Y) \le e$. Then $\dim_{\mathbb{F}_h}(\langle X\rangle \cap \langle Y\rangle) \ge \dim_{\mathbb{F}_h}(\langle X\rangle) - e$.

Proof. First we observe that the two $\mathbb{F}_h$-spaces $\langle X\rangle + \langle Y\rangle$ and $\langle X - Y\rangle + \langle Y\rangle$ are equal. Thus,

$$\begin{aligned}\dim_{\mathbb{F}_h}(\langle X\rangle) + \dim_{\mathbb{F}_h}(\langle Y\rangle) - \dim_{\mathbb{F}_h}(\langle X\rangle \cap \langle Y\rangle)& \\ = \dim_{\mathbb{F}_h}(\langle X - Y\rangle) + \dim_{\mathbb{F}_h}(\langle Y\rangle) - \dim_{\mathbb{F}_h}(\langle X - Y\rangle \cap \langle Y\rangle).&\end{aligned}$$

This gives

$$\begin{aligned}\dim_{\mathbb{F}_h}(\langle X\rangle \cap \langle Y\rangle) &= \dim_{\mathbb{F}_h}(\langle X\rangle) - \dim_{\mathbb{F}_h}(\langle X-Y\rangle) + \dim_{\mathbb{F}_h}(\langle X-Y\rangle \cap \langle Y\rangle) \\ &\geq \dim_{\mathbb{F}_h}(\langle X\rangle) - e\ ,\end{aligned}$$

completing the proof. □

We now adapt the linear-algebraic decoding of Chapter 3 to the case of Gabidulin codes. As always, the algorithm is broken up into the two steps of interpolation and root-finding.

Interpolation step. Let $1 \leq s \leq m$ be an integer parameter of the algorithm. Choose the "degree parameter" $D = \left\lfloor \frac{n-k+1}{s+1} \right\rfloor$.

Definition 6.4. Let $\mathcal{L}$ be the space of polynomials $Q \in \mathbb{F}_{q^m}[X, Z_1, Z_2, \ldots, Z_s]$ of the form $Q(X, Z_1, Z_2, \ldots, Z_s) = A_0(X)+A_1(Z_1)+A_2(Z_2)+\cdots+A_s(Z_s)$, with each $A_i \in \mathbb{F}_{q^m}[X]$ being an h-linearized polynomial and $\deg_h(A_0) \leq D + k - 1$ and $\deg_h(A_i) \leq D$ for $i = 1, 2, \ldots, s$.

The following lemma shows that we can find a nonzero polynomial satisfying certain interpolation conditions.

Lemma 6.5. There exists a nonzero polynomial $Q \in \mathcal{L}$ such that

$$Q(\alpha_i, y_i, y_i^q, y_i^{q^2}, \cdots, y_i^{q^{s-1}}) = 0$$

for $i = 1, 2, \ldots, n$. Further such a Q can be found using $O(n^3)$ operations over $\mathbb{F}_{q^m}$.

Proof. Note that $\mathcal{L}$ is an $\mathbb{F}_{q^m}$-vector space of dimension $(D + k) + s(D + 1) = (D + 1)(s + 1) + k - 1$. This dimension is bigger than n by our choice of D. The conditions imposed by the Lemma amount to n homogeneous linear conditions on Q. Since this is smaller than the $\mathbb{F}_{q^m}$-dimension of $\mathcal{L}$, there must exist a nonzero $Q \in \mathcal{L}$ that meets the interpolation conditions $Q(\alpha_i, y_i, y_i^q, y_i^{q^2}, \cdots, y_i^{q^{s-1}}) = 0$ for $i = 1, 2, \ldots, n$. Finding such a Q amounts to solving a homogeneous linear system over $\mathbb{F}_{q^m}$ with n constraints and at most $\dim_{\mathbb{F}_{q^m}}(\mathcal{L}) \leq n + s + 2$ unknowns, which can be done in $O(n^3)$ time using Gaussian elimination. □

Lemma 6.6 below shows that any polynomial Q given by Lemma 6.5 yields an algebraic condition that the message functions f we are interested in list decoding must satisfy. For a polynomial $f(X) = f_0 + f_1X + \cdots + f_{k-1}X^{k-1}$, we will write f^σ to denote the polynomial $f^\sigma(X) = f_0^q + f_1^qX + \cdots + f_{k-1}^qX^{k-1}$, and f^{σ^i} for $\left(f^{\sigma^{i-1}}\right)^\sigma$. (In other words, σ is the Frobenius automorphism of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$.)

Lemma 6.6. Let $f \in \mathbb{F}_{q^m}[X]$ be a h-linearized polynomial with h-degree at most $k-1$. Suppose that a codeword $M_f = (f(\alpha_1), f(\alpha_2), \ldots, f(\alpha_n))^T$ is transmitted and $Y = (y_1, y_2, \ldots, y_n)^T$ is received with at most e rank errors. If $e \leq s(n-k)/(s+1)$, then $Q(X, f(X), f^{\sigma}(X), f^{\sigma^2}(X), \cdots, f^{\sigma^{s-1}}(X)) = 0$.

Proof. The polynomial $f(X)$ defines an $\mathbb{F}_h$-linear map from $\mathbb{F}_{h^n} = \mathbb{F}_q$ to $\mathbb{F}_{q^m}$. Denote by A and B the $n \times 2t$ matrices $((\alpha_1, \alpha_2, \ldots, \alpha_n)^T, M_f)$ and $((\alpha_1, \alpha_2, \ldots, \alpha_n)^T, Y)$, respectively. It is clear that $\mathrm{rank}(A-B) = \mathrm{rank}(M_f - Y) \leq e$ and $\mathrm{rank}(A) = n$. Thus, it follows from Lemma 6.3 that $\dim_{\mathbb{F}_h}(\langle A \rangle \cap \langle B \rangle) \geq n - e$. This implies that there exists an $\mathbb{F}_h$-subspace U of $\mathrm{span}\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ of dimension at least $n-e$ such that, for every $\alpha = \sum_{i=1}^n c_i \alpha_i \in U$ with $c_i \in \mathbb{F}_h$, one has $\sum_{i=1}^n c_i f(\alpha_i) = \sum_{i=1}^n c_i y_i$. Hence,

$$\begin{aligned} 0 &= \sum_{i=1}^{n} c_i Q(\alpha_i, y_i, y_i^q, y_i^{q^2}, \cdots, y_i^{q^{s-1}}) \\ &= Q(\alpha, f(\alpha), f(\alpha)^q, \ldots, f(\alpha)^{q^{s-1}}) \\ &= Q(X, f, f^{\sigma}, \ldots, f^{\sigma^{s-1}})(\alpha) \, . \end{aligned}$$

As the h-degree of $Q(X, f, f^{\sigma}, \ldots, f^{\sigma^{s-1}})$ is at most $D + k - 1$, under the condition

$$D + k - 1 < n - e, \tag{6.1}$$

we have

$$Q(X, f, f^{\sigma}, \ldots, f^{\sigma^{s-1}}) = A_0(X) + A_1(f(X)) + A_2(f^{\sigma}(X)) + \cdots + A_s(f^{\sigma^{s-1}}(X)) = 0. \tag{6.2}$$

This completes the proof since (6.1) is indeed satisfied by our given condition on e and choice of $D = \lfloor \frac{n-k+1}{s+1} \rfloor$. □

Finding candidate solutions. We want to study the structure of a linearized polynomial f satisfying the condition of Lemma 6.6.

Lemma 6.7. The set of solutions $f = \sum_{i=0}^{k-1} f_i X^{h^i} \in \mathbb{F}_{q^m}[X]$ to the equation

$$A_0(X) + A_1(f(X)) + A_2(f^{\sigma}(X)) + \cdots + A_s(f^{\sigma^{s-1}}(X)) = 0 \tag{6.3}$$

when at least one of $\{A_0, A_1, \ldots, A_s\}$ is nonzero is an $(s-1, m, k)$-periodic subspace. A canonical representation of this periodic subspace (in the sense of Definition 6.2) can be computed in $\mathrm{poly}(k, m, \log q)$ time.

Proof. If f, g are two solutions to (6.3), then so is $\alpha f + \beta g$ for any $\alpha, \beta \in \mathbb{F}_q$ with $\alpha + \beta = 1$. So the solutions to (6.3) form an affine $\mathbb{F}_q$-subspace. We now proceed to analyze the structure of the subspace.

First, replacing each $A_0, A_1, \ldots, A_s$ with $A_0^{h^j}, A_1^{h^j}, \ldots, A_s^{h^j}$ for some j, we can assume that at least one $A_{i^*}(X)$ for some $i^* \in \{0, 1, \ldots, s\}$ has a nonzero coefficient on X. Further, if each of $A_1(X), \ldots, A_s(X)$ has zero coefficient on X, then so does $A_0(X)$, so we can take $i^* > 0$.

Let us denote $A_\iota(X) = a_{\iota,0}X + a_{\iota,1}X^h + a_{\iota,2}X^{h^2} + \cdots$ for $\iota = 0, 1, 2, \ldots, s$. For $l = 0, 1, 2, \ldots, k-1$, define the linearized polynomial

$$B_l(X) = a_{1,l}X + a_{2,l}X^q + a_{3,l}X^{q^2} + \cdots + a_{s,l}X^{q^{s-1}} . \tag{6.4}$$

We know that $a_{i^*,0} \neq 0$, and therefore $B_0 \neq 0$. This implies that the solutions $\beta \in \mathbb{F}_{q^m}$ to $B_0(\beta) = 0$ is a subspace, say W, of $\mathbb{F}_{q^m}$ of dimension at most $s - 1$.

Fix an $i \in \{0, 1, \ldots, k-1\}$. Expanding the equation (6.3) and equating the coefficient of X^i to be 0, we get

$$a_{0,i} + B_i(f_0^{h^i}) + B_{i-1}(f_1^{h^{i-1}}) + \cdots + B_1(f_{i-1}^h) + B_0(f_i) = 0 . \tag{6.5}$$

This implies $f_i \in W + \theta_i$ for some $\theta_i \in \mathbb{F}_{q^m}$ that is determined by $f_0, f_1, \ldots, f_{i-1}$. Therefore, for each choice of $f_0, f_1, \ldots, f_{i-1}$, f_i must belong to a fixed coset of the subspace W of dimension at most $s - 1$. Thus, the solutions belong to an $(s-1, m, k)$-periodic subspace. Also, it is clear from (6.5) that a canonical representation of the periodic subspace can be computed in $\mathrm{poly}(k, m, \log q)$ time.

□

Combining Lemmas 6.6 and 6.7, we see that one can find an $(s-1, m, k)$-periodic subspace that contains the coefficients of all polynomials whose encodings differ from the input $(y_1, \ldots, y_n)$ by a matrix of rank at most $\frac{s}{s+1}(1-R)n$ (where R is the rate). When $s \ll m$, this dimension is much smaller than the dimension of the message space of the Gabidulin code $\mathcal{C}_G(q; n, m, k)$ over $\mathbb{F}_q$, which is km.

6.2 List size reduction via subspace designs

We now turn our attention to constructing sets which will "evade" periodic subspaces. As before, we will think of subspaces $W \subseteq \mathbb{F}_q^m$ as $\mathbb{F}_h$-subspaces of $\mathbb{F}_h^{mn}$ via some fixed basis embedding.

Definition 6.8. A collection S of $\mathbb{F}_h$-subspaces $H_1, \ldots, H_M \subseteq \mathbb{F}_h^{mn}$ is called an (s, A, n) $\mathbb{F}_h$-*subspace design* if for every $\mathbb{F}_{h^n}$-linear space $W \subset \mathbb{F}_{h^n}^m$ of dimension s,

$$\sum_{i=1}^{M} \dim_{\mathbb{F}_h}(H_i \cap W) \leq A.$$

Note that in the above definition the dimension of the input W is measured as a subspace over $\mathbb{F}_{h^n}$ whereas for the intersection, which is an $\mathbb{F}_h$-subspace, the dimension is over $\mathbb{F}_h$.

Remark. When $n = 1$, these are the (strong) subspace designs of [GK13]. We will be interested in settings where n is super-constant, so that considering W as a subspace of dimension sn over $\mathbb{F}_h$ will generally not give strong enough bounds.

The motivation for this definition is the following, which shows that such designs yield sets which have low intersection with periodic subspaces.

Proposition 6.9. Let H be an (s, m, k)-periodic affine subspace of $\mathbb{F}_q^{mk}$ (in the sense of Definition 6.1), and let $H_1, H_2, \ldots, H_k \subseteq \mathbb{F}_h^{mn}$ be distinct subspaces from an (s, A, n) $\mathbb{F}_h$-subspace design. Then $H \cap (H_1 \times \cdots \times H_k)$ is an affine subspace over $\mathbb{F}_h$ of dimension at most A.

Proof. It is clear that $H \cap (H_1 \times \cdots \times H_k)$ is an affine subspace over $\mathbb{F}_h$. Let W be the subspace associated to H as in Definition 6.1. We will show by induction that $|\mathrm{proj}_{[1,im]}(H) \cap (H_1 \times \cdots \times H_i)| \leq h^{\sum_{j=1}^{i} \dim_{\mathbb{F}_h}(W \cap H_j)}$.

In the base case, since H_1 is a subspace, $\mathrm{proj}_{[1,m]}(H) \cap H_1 = (W + v_{\mathbf{0}}) \cap H_1$ is an affine subspace whose underlying subspace lies in $W \cap H_1$. In particular, its size is at most $h^{\dim(W \cap H_1)}$.

Continuing, fix an element $\mathbf{a} \in \mathrm{proj}_{[1,im]}(H) \cap (H_1 \times \cdots \times H_i)$. Because H is periodic and H_{i+1} is linear, the possible extensions of $\mathbf{a}$ in $\mathrm{proj}_{[im+1,(i+1)m]}(H) \cap H_{i+1}$ are given by a coset of $W \cap H_{i+1}$. Thus, there are at most $h^{\dim(W \cap H_{i+1})}$ such extensions. Since by induction there were $h^{\sum_{j=1}^{i} \dim_{\mathbb{F}_h}(W \cap H_j)}$ possibilities for the prefix $\mathbf{a}$, the result follows.

In particular, the dimension of $H \cap (H_1 \times \cdots \times H_k)$ over $\mathbb{F}_h$ is at most $\sum_{i=1}^{k} \dim(W \cap H_i) \leq A$, by the subspace design property. □

6.2.1 Existential bounds

The following proposition shows that good subspace designs exist; indeed, a random collection of subspaces works with high probability. The case $n = 1$ was established in [GK13].

Proposition 6.10. Let $\varepsilon > 0$. Let S consist of $M = h^{\varepsilon mn/8}$ $\mathbb{F}_h$-subspaces of codimension εmn in $\mathbb{F}_h^{mn}$, chosen independently at random. Then for any $s < m\varepsilon/2$, with probability at least $1 - q^{-ms}$, S is an $(s, 8s/\varepsilon, n)$ $\mathbb{F}_h$-subspace design. (Here $q = h^n$.)

Proof. Set $\ell = 8s/\varepsilon$, and let $S = \{H_1, \ldots, H_M\}$. For a fixed $\mathbb{F}_{h^n}$ subspace W of dimension s and any j, the probability that $\dim_{\mathbb{F}_h}(W \cap H_j) \geq a$ is at most $q^{sa} \cdot q^{-\varepsilon ma} \leq q^{-\varepsilon ma/2}$, by assumption on s.

Since the H_i are independent, for a fixed tuple $(a_1, \ldots, a_M)$ of nonnegative integers summing to $\ell = 8s/\varepsilon$, the probability that $\dim(W \cap H_j) \geq a_j$ for each j is at most $q^{-\varepsilon m\ell/2} = q^{-4ms}$. Union bounding over the at most q^{ms} choices of W and $\binom{\ell+M}{\ell} \leq M^{2\ell}$ choices of $(a_1, \ldots, a_M)$, the probability S is *not* an $(s, 8s/\varepsilon, t)$ $\mathbb{F}_h$-subspace design is at most

$$q^{ms} M^{2\ell} \cdot q^{-4ms} = q^{ms} \cdot q^{2ms} \cdot q^{-4ms} \leq q^{-ms} \, . \qquad \square$$

6.2.2 Constructive bounds

In this section, we show how to construct an explicit large $\big(s, 2(m-1)s/\varepsilon, n\big)$ $\mathbb{F}_h$-subspace design consisting of $\mathbb{F}_h$-subspaces of $\mathbb{F}_h^{mn}$ of codimension $2\varepsilon mn$.

The idea, which is natural in hindsight, is to first use a subspace design over $\mathbb{F}_{h^n}$ to ensure that the intersection with any $\mathbb{F}_{h^n}$-subspace of dimension s has low dimension over $\mathbb{F}_{h^n}$, and then to use a subspace-evasive set to reduce the dimension further over $\mathbb{F}_h$. The final construction appears as Theorem 6.14.

Explicit subspace-evasive sets

We first describe the construction of explicit subspace-evasive sets which we will be using. Recall that we first defined subspace-evasive sets in Chapter 3, where they were used to construct list-decodable codes with small list size.

Let $q > h^{m-1}$, and let $\gamma_1, \ldots, \gamma_m$ be distinct nonzero elements of $\mathbb{F}_q$. Then Dvir and Lovett [DL12] showed the following:

Theorem 6.11 ([DL12])**.** Let $1 \leq s \leq m$. Let $d_1 > d_2 > \cdots > d_m \geq 1$ be integers. Define $f_1, \ldots, f_s \in \mathbb{F}_q[X_1, \ldots, X_m]$ as follows:

$$f_i(x_1, \ldots, x_m) = \sum_{j=1}^{m} \gamma_j^i x_j^{d_j} \, . \tag{6.6}$$

Then:

- The variety $\mathbf{V} = \{x \in \overline{F}_q^m \mid f_1(x) = \cdots = f_s(x) = 0\}$ satisfies $|\mathbf{V} \cap H| \leq (d_1)^s$ for all s-dimensional affine subspaces $H \subset \overline{\mathbb{F}}_q^m$.
- If at least s of the degrees d_i are relatively prime to $q-1$, then $|\mathbf{V} \cap \mathbb{F}_q^m| = q^{m-s}$.

Additionally, the product set $(\mathbf{V} \cap \mathbb{F}_q^m)^{n/m} \subseteq \mathbb{F}_q^n$ is $(k, (d_1)^k)$-subspace evasive for all $k \leq s$.

The below statement follows immediately from Theorem 6.11 and the fact that when the d_j's are powers of h, the polynomials f_i defined in (6.6) are $\mathbb{F}_h$-linear functions on $\mathbb{F}_q^m$.

Corollary 6.12. Setting $d_1 = h^{m-1}, d_2 = h^{m-2}, \ldots, d_m = 1$, we obtain an explicit $\mathbb{F}_h$-linear set S of size $q^{(m-s)n/m}$ over $\mathbb{F}_q^n$ which is $(k, h^{(m-1)k})$ subspace-evasive for all $1 \leq k \leq s$.

Remark. One can improve on the degree bounds and therefore the final intersection size via a standard subspace-evasive set without the $\mathbb{F}_h$-linearity requirement. For example, [DL12] gives a construction of a (non-linear) $\big(s, (s/\varepsilon)^s\big)$ subspace-evasive set over $\mathbb{F}_q^n$ of size $q^{(1-\varepsilon)n}$.

However, especially in applications for rank-metric codes, linearity is a property which is desirable and often necessary.

Combining with subspace designs

The following theorem shows how to achieve our initial goal of ensuring small intersection dimension over the larger field $\mathbb{F}_{h^n}$.

Theorem 6.13 ([GK13])**.** For $\varepsilon \in (0,1)$, positive integers s, m with $s \leq \varepsilon m/4$, and $q > m$, there is an explicit collection of $M = q^{\Omega(\varepsilon m/s)}$ subspaces in $\mathbb{F}_q^m$, each of codimension at most εm, which form an $(s, 2s/\varepsilon, 1)$ $\mathbb{F}_q$-subspace design.

Moreover, bases for $N \leq M$ elements of this collection can be computed in time $\mathrm{poly}\big(\mathrm{char}(\mathbb{F}_q), m, N\big)$.

Remark. The runtime stated in [GK13] is polynomial in the field size q, dominated by the cost of computing a representation of an extension $\mathbb{F}_{q^r}$ of the field $\mathbb{F}_q$. It has been shown (see Theorem 4.1 in [Sho88]) that this representation can be found in the faster runtime claimed above.

Combined with Corollary 6.12, we now have a construction of a $(s, 2(m-1)s/\varepsilon, n)$ $\mathbb{F}_h$-subspace design, summarized in the following statement.

Theorem 6.14. For integers $s \leq \varepsilon m/4$ and $q = h^n > m$, there exists an explicit set of $q^{\Omega(\varepsilon m/s)}$ $\mathbb{F}_h$-subspaces in $\mathbb{F}_h^{mn}$ of codimension at most $2\varepsilon mn$ forming an $(s, 2(m-1)s/\varepsilon, n)$ $\mathbb{F}_h$-subspace design.

Proof. Let $V_1, \ldots, V_M \subseteq \mathbb{F}_q^m$ be the elements of the $(s, 2s/\varepsilon, 1)$ $\mathbb{F}_q$-subspace design of Theorem 6.13. For each i, define $H_i = V_i \cap S$, where $S \subseteq \mathbb{F}_q^m$ is the $(s, h^{(m-1)s})$ subspace-evasive set of Corollary 6.12. As S and the V_i's are $\mathbb{F}_h$-linear subspaces, H_i is as well. We claim that the H_i's form the desired $\mathbb{F}_h$-subspace design.

Recall that $q = h^n$. For each i, V_i has $\mathbb{F}_h$-codimension εmn, and S has $\mathbb{F}_h$-codimension $sn \leq \varepsilon mn/4$, so the codimension of H_i is at most $2\varepsilon mn$ over $\mathbb{F}_h$.

Now let W be an $\mathbb{F}_q$-subspace of dimension s. By the $\mathbb{F}_q$-subspace design property of the V_i's we have

$$\sum_{i=1}^{M} \dim_{\mathbb{F}_q}(V_i \cap W) \leq 2s/\varepsilon \,. \tag{6.7}$$

For each i, we also have that $\dim_{\mathbb{F}_q}(W \cap V_i) = s_i \leq s$, so by the subsace evasive property of S from Corollary 6.12, $W \cap H_i = (W \cap V_i) \cap S$ has at most $h^{(m-1)s_i}$ elements. As $W \cap H_i$ is $\mathbb{F}_h$-linear, we have

$$\dim_{\mathbb{F}_h}(W \cap H_i) \leq (m-1) \dim_{\mathbb{F}_q}(W \cap V_i) \,. \tag{6.8}$$

Combining (6.7) and (6.8) we have

$$\sum_i \dim_{\mathbb{F}_h}(W \cap H_i) \leq \sum_i (m-1) \dim_{\mathbb{F}_q}(W \cap V_i) \leq (m-1) \cdot 2s/\varepsilon \,. \qquad \square$$

6.2.3 Explicit list-decodable rank-metric codes

We now apply the subspace designs of Theorem 6.14 to give explicit list-decodable subcodes of the Gabidulin code when the evaluation points span a subfield. By Lemma 6.7, the linear-algebraic decoding outputs a periodic subspace containing the original message.

By Proposition 6.9, by restricting the message polynomials $f = \sum_i f_i X^{h^i}$ to have coefficients $f_i \in H_{i+1}$ for $0 \leq i < k$, where $H_1, H_2, \ldots, H_k$ are distinct elements of the subspace design in Theorem 6.14, we obtain a fully explicit $\mathbb{F}_h$-linear code. In particular,

this code admits an efficient (linear) encoding function. The dimension of the code is at least $kt - 2\varepsilon knm$, by Theorem 6.14. Recalling that $t = nm$, the final rate of the code is at least $(1 - 2\varepsilon)k/n$.

Because our linear-algebraic decoding algorithm outputs a periodic subspace, the final list of candidate messages, which is the intersection of this periodic subspace with our code, will have dimension at most $2(m-1)s/\varepsilon$ over $\mathbb{F}_h$. In addition, as both sets are linear, this intersection can be computed efficiently in terms of $t, \log h$, and the intersection size $h^{O(ms/\varepsilon)}$.

As one can take $m = O(s/\varepsilon)$ for the necessary subspace design guaranteed by Theorem 6.14, we can conclude the following theorem.

Theorem 6.15. For every $\varepsilon \in (0, 1)$ and integer $s > 0$, there exists an explicit $\mathbb{F}_h$-linear subcode of the Gabidulin code $\mathcal{C}_G(h; n, t, k)$ with evaluation points spanning $\mathbb{F}_{h^n}$ of rate $(1 - 2\varepsilon)k/n$ which is list-decodable in polynomial time from up to $\frac{s}{s+1} \cdot (n - k)$ rank errors. The final list is contained in an $\mathbb{F}_h$-subspace of dimension at most $O(s^2/\varepsilon^2)$, and thus has size at most $h^{O(s^2/\varepsilon^2)}$.

By setting s to be $O(1/\varepsilon)$, we obtain a code of rate $R = (1 - 2\varepsilon)k/n$ which is list-decodable from up to $(1 - R - \varepsilon)n$ rank errors, with a list size of $h^{\mathrm{poly}(1/\varepsilon)}$.

Remark. It is possible to obtain an improved list size of $O(1/\varepsilon)$ using a *Monte Carlo* construction of the list-decodable subcode. For details, see [GX13].

Remark. The authors of [MV12] use a "folded" variant of Gabidulin codes to obtain an explicit code with a similar trade-off between rate and correctable error radius as in Theorem 6.15. However, the output list size for this folded variant is *exponential* in the dimension of the code when the rate is constant (see the discussion in Chapter 5).

6.3 Explicit list-decodable subspace codes

In this section, we show that a similar approach to that of the previous section also gives explicit list-decodable *subspace* codes. These codes are obtained by taking subcodes of the Kötter-Kschischang (KK) codes defined in Chapter 5.

Our results, as in the Gabidulin case, will only hold when the evaluation points are chosen to span a subfield, rather than being arbitrary linearly independent points. To fix the notation, let us recall the definition of the KK codes in this context below.

For n dividing t, let $\mathbb{F}_{h^t}$ extend $\mathbb{F}_h$, and let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_{h^t}$ generate the subfield $\mathbb{F}_{h^n} := \mathbb{F}_q$.

Set $m = t/n$. Then the (n, k, t) KK code encodes an $\mathbb{F}_h$-linearized polynomial over $\mathbb{F}_{q^m} = \mathbb{F}_{h^t}$ of h-degree $< k$ by

$$f(X) \mapsto \mathrm{span}\{(\alpha_i, f(\alpha_i))\}_{i=1}^{n}.$$

6.3.1 Linear algebraic list-decoding for subspace codes

We now present a list decoding algorithm for the above KK codes. The algorithm follows the earlier linear-algebraic list decoding algorithm for Gabidulin codes.

As in Chapter 5, the error level will be quantified by two integer parameters: (i) ρ, the maximum number of *insertions* allowed, and (ii) μ, the maximum number of *deletions* allowed.

Suppose a codeword V_f encoded from f is transmitted. In the above error model, the subspace V_f is received as $U = W \oplus E$, where $\dim_{\mathbb{F}_h}(E) \leq \rho$ and W is a subspace of V_f with $\dim_{\mathbb{F}_h}(V_f) - \dim_{\mathbb{F}_h}(W) := \nu \leq \mu$. Assume that $\dim_{\mathbb{F}_q}(U) = d$.

Consider a nonzero polynomial in $\mathbb{F}_{q^m}[X, Y_1, Y_2, \ldots, Y_s]$ with $1 \leq s \leq m$

$$Q(X, Y_1, Y_2, \ldots, Y_s) = A_0(X) + A_1(Y_1) + A_2(Y_2) + \cdots + A_s(Y_s), \tag{6.9}$$

where every $A_i \in \mathbb{F}_{q^m}[X]$ is a h-linearized polynomial with $\deg_h(A_0) \leq D + k - 1$ and $\deg_h(A_i) \leq D$ for $i = 1, \ldots, s$; and D is chosen to be

$$D = \left\lfloor \frac{d - k - s + 1}{s + 1} \right\rfloor. \tag{6.10}$$

Choose an $\mathbb{F}_h$-basis $\{(a_i, b_i)\}_{i=1}^{d}$ of U (where $a_i \in \mathbb{F}_q$ and $b_i \in \mathbb{F}_{q^m}$) and we interpolate a polynomial Q of the above form satisfying

$$Q(a_i, b_i, b_i^q, \ldots, b_i^{q^{s-1}}) = 0 \quad \text{for } i = 1, 2, \ldots, d\,.$$

There are d equations, but $D + k + s(D + 1) = (s + 1)D + k + s$ freedoms in Q. Hence, such a nonzero polynomial Q exists since $d < (s + 1)D + k + s$. It is clear that for all $(\alpha, f(\alpha)) \in W$, we have

$$0 = Q(\alpha, f(\alpha), f(\alpha)^q, \ldots, f(\alpha)^{q^{s-1}}) = Q(X, f, f^\sigma, \ldots, f^{\sigma^{s-1}})(\alpha),$$

where σ is the Frobenius automorphism of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, i.e., σ sends every element α in $\mathbb{F}_{q^m}$ to α^q.

As the h-degree of $Q(X, f, f^\sigma, \ldots, f^{\sigma^{s-1}})$ is at most $D + k - 1$, under the condition

$$D + k - 1 < n - \nu, \tag{6.11}$$

we have

$$Q(X, f, f^\sigma, \ldots, f^{\sigma^{s-1}}) = A_0(X) + A_1(f(X)) + A_2(f^\sigma(X)) + \cdots + A_s(f^{\sigma^{s-1}}(X)) = 0. \tag{6.12}$$

Note that we have

$$\rho < s(n-\mu-k+1) \Rightarrow \rho < s(n-\nu-k+1) \Rightarrow d-n+\nu < s(n-\nu-k+1) \Rightarrow D+k-1 < n-\nu.$$

Thus, Condition (6.11) is met if

$$s\mu + \rho < s(n - k + 1). \tag{6.13}$$

The above analysis shows that we can list decode up to ρ insertions and μ deletions as long as ρ and ν satisfy (6.13).

The equation (6.12) satisfied by f is identical to (6.3), and therefore one can pin down f to an affine space of solutions exactly as in Lemma 6.7.

6.3.2 Explicit list-decodable subcodes

The result of the previous section shows that the linear algebraic list-decoder outputs a *periodic* subspace.

As in the rank-metric case, we can improve the list size using subspace designs. By restricting the coefficients of the message polynomial f to come from distinct $H_1, \ldots, H_k$ from the $\big(s, 2(m-1)s/\varepsilon, t\big)$-subspace design of Theorem 6.14, and setting $m \approx s/\varepsilon$, we can prune this list down to a $\mathbb{F}_h$-subspace of dimension $O(s^2/\varepsilon^2)$.

Notice that the H_i's are $\mathbb{F}_h$-linear subspaces, so the restricted subcode is linear. In summary, we have:

Theorem 6.16. For every $\varepsilon \in (0, 1)$ and integer $s > 0$, there exists an explicit linear subcode of the $\big(n, k, sn/\varepsilon\big)$ KK code of rate $(1-\varepsilon)k/n$ which is list-decodable in polynomial time from up to ρ insertions and μ deletions, provided $\rho + s\mu < s(n - k + 1)$.

Moreover, the output list is contained in an $\mathbb{F}_h$-subspace of dimension $O(s^2/\varepsilon^2)$, and thus has size $h^{O(s^2/\varepsilon^2)}$.

We conclude by commenting on the quality of our condition (6.13) for successful decoding, which is implied by the condition

$$\mu + \frac{\rho}{s} < n\left(1 - R - \frac{t}{n}\right) , \tag{6.14}$$

where R is the rate (5.1) of the code. For comparison, the condition for successful decoding for the folded KK codes in [GNW12] is (essentially)

$$\mu + \frac{\rho}{s} < n(1 - Rt)$$

which necessitates a sub-constant rate for the code (a similar situation holds for [MV12]).

Recall that in Theorem 5.13, we showed the existence of subspace codes that can be list decoded with list size L when

$$\mu + \frac{\rho}{L+1} < n\left(1 - R - \frac{1}{L+1}\right) , \tag{6.15}$$

where R is the rate of the code. To compare this with our result for KK codes, we note that after the combination with Theorem 6.14, we can take $s = \Theta(1/\varepsilon)$, $m = t/n = \Theta(1/\varepsilon^2)$, and have a list decodable subspace code that can correct μ deletions and ρ insertions provided

$$\mu + \varepsilon\rho < n(1 - R - \varepsilon) ,$$

with a worst-case output list size of $h^{\mathrm{poly}(1/\varepsilon)}$. This essentially matches the existential trade-off (6.15), with a larger (but still polynomial) list size.

The correctable error radius shown in this work matches that shown in [MV12], allowing for decoding up to the Singleton bound. However, we are able to prove a *polynomial* list size for our code, whereas the size of the list output by the decoder of [MV12] must be exponential in the dimension of the codewords, as discussed in Chapter 5.

6.4 Application to low-order folding of Reed-Solomon codes

In this section, we show how the idea of only evading subspaces over an extension field can be used to give an algorithm for list-decoding (subcodes of) folded Reed-Solomon codes in the case when the folding parameter has low ($O(1)$) order.

As in the case of KK codes, our decoding algorithm follows the framework of interpolating a linear polynomial and then solving a linear system for candidate polynomials.

Let ℓ be an integer dividing $q-1$, and fix γ generating $\mathbb{F}_q^*$. Let $N = \frac{q-1}{\ell}$, and let $\zeta = \gamma^N$, which has order ℓ in $\mathbb{F}_q$. Then the **low-order folded Reed-Solomon code** encodes a polynomial f of degree $< k$ by

$$f \mapsto \begin{bmatrix} f(1) & f(\gamma) & \cdots & f(\gamma^{N-1}) \\ f(\zeta) & f(\zeta\gamma) & \dots & f(\zeta\gamma^{N-1}) \\ \vdots & \vdots & \ddots & \vdots \\ f(\zeta^{\ell-1}) & f(\zeta^{\ell-1}\gamma) & \dots & f(\zeta^{\ell-1}\gamma^{N-1}) \end{bmatrix}.$$

Similarly to folded Reed-Solomon codes (as seen in Chapter 3), this is a code of rate $R = \frac{k}{\ell N}$ and distance $N - (k-1)/\ell$. Whereas the results of Chapter 3 show how to list-decode these codes when the order of ζ is at least k, we are now interested in the case when ζ has constant order. This setting of parameters, when the order of the folding parameter is independent of the code size, is of interest because of its relationship to Reed-Solomon list-decoding (see Section 6.4.4). However, we will only be able to list-decode *subcodes* of these codes.

In what follows, we give a list-decoding algorithm for low-order folded Reed-Solomon codes and show that we can give explicit subcodes of rate R which are *efficiently* list-decodable up to a $1 - R - \varepsilon$ fraction of errors.

6.4.1 Interpolation

Given a received word

$$\begin{pmatrix} y_{00} & y_{01} & \cdots & y_{0(N-1)} \\ y_{10} & y_{11} & \cdots & y_{1(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ y_{(\ell-1)0} & y_{(\ell-1)1} & \cdots & y_{(\ell-1)(N-1)} \end{pmatrix},$$

and an integer parameter s, we would like to interpolate a (nonzero) polynomial

$$Q(X, Y_1, \dots, Y_s) = A_0(X) + A_1(X)Y_1 + \cdots + A_s(X)Y_s$$

such that

$$Q\left(\gamma^{iN+j}, y_{ij}, y_{(i+1)j}, \dots, y_{(i+s-1)j}\right) = 0 \qquad i \in \{0, \dots, \ell-1\},\ j \in \{0, \dots, N-1\}, \tag{6.16}$$

where all indices are taken modulo ℓ.

We will require $\deg(A_0) \le D + k - 1$, and $\deg(A_i) \le D$ for $i > 0$.

Lemma 6.17. Let

$$D = \left\lfloor \frac{\ell N - k + 1}{s+1} \right\rfloor.$$

Then a nonzero polynomial Q satisfying (6.16) exists (and can be found by solving a linear system).

Proof. The number of interpolation conditions is ℓN. The quantity $(D+1)(s+1) + k - 1 > \ell N$ is the number of degrees of freedom for the interpolation, and the conditions are homogeneous, so a nonzero solution exists. □

Lemma 6.18. If the number of agreements t is greater than $\frac{D+k-1}{\ell}$, then

$$Q\big(X, f(X), f(\zeta X), \ldots, f(\zeta^{s-1}X)\big) = 0. \tag{6.17}$$

Proof. $Q\big(X, f(x), \ldots, f(\zeta^{s-1}X)\big)$ is a univariate polynomial of degree $D + k - 1$, and each correct column j yields ℓ distinct roots γ^{iN+j} for $i \in \{0, \ldots, \ell - 1\}$. Thus if $t\ell > D + k - 1 \geq \deg Q$, Q is the zero polynomial. □

For our choice of D, the requirement on t in Lemma 6.18 is met if t satisfies

$$\frac{t}{N} \geq \frac{1}{s+1} + \frac{s}{s+1}R. \tag{6.18}$$

Remark. In ordinary folded Reed-Solomon codes, where the folding parameter is primitive of order $q - 1$, the agreement fraction required to satisfy (6.17) is

$$\frac{t}{N} \geq \frac{1}{s+1} + \frac{s}{s+1}\frac{\ell R}{\ell - s + 1},$$

which is higher than (6.18). In our case, because ζ has low order, we are able to use interpolation conditions that "wrap around," allowing us to impose ℓ conditions per coordinate rather than $\ell - s + 1$. Therefore we can satisfy Equation (6.17) with lower agreement. On the other hand, it is known how to list-decode folded Reed-Solomon codes themselves, whereas we are only able to list-decode a subcode.

6.4.2 Decoding

In this section, we describe how to solve the system

$$Q\big(X, f(X), f(\zeta X), \ldots, f(\zeta^{s-1}X)\big) = 0 \tag{6.17}$$

for candidate polynomials f.

Proposition 6.19. Let $P(X) \in \mathbb{F}_q[X]$ be an irreducible polynomial such that

- $\deg P \geq k$, and
- for some a, $\zeta X \equiv X^{q^a} \pmod{P}$.

Then the set of f of degree $< k$ satisfying (6.17) is an $\mathbb{F}_{q^a}$-affine subspace of dimension at most $s-1$.

Proof. The condition (6.17) says

$$0 = A_0(X) + A_1(X)f(X) + A_2(X)f(\zeta X) + \cdots + A_s(X)f(\zeta^{s-1}X).$$

Then we have

$$A_0(X) + A_1(X)f(X) + A_2(X)f(X)^{q^a} + \cdots + A_s(X)f(X)^{q^{(s-1)a}} \equiv 0 \pmod{P}.$$

By dividing out the highest power of P which divides every A_i, Equation (6.17) is still satisfied and we may assume that this equation is nonzero mod P.

In particular, this equation has at most $q^{(s-1)a}$ solutions for f mod P. When $\deg f < k \leq \deg P$, f is uniquely determined by its residue mod P and there are at most $q^{(s-1)a}$ solutions for f.

The fact that the solution space is $\mathbb{F}_{q^a}$-affine follows from the fact that the terms in which $f(X)$ appears all have degree q^{ai} for some i. □

Because the output space is a subspace (over the large field $\mathbb{F}_{q^a}$), by picking the message polynomials f to come from a subspace-evasive set, we can reduce the list size bound. More specifically, if ℓ is at least s/ε, [DL12] gives a construction of a $(s, (s/\varepsilon)^s)$ subspace-evasive set S over $(\mathbb{F}_{q^a})^{k/a}$ of size $q^{(1-\varepsilon)k}$. By precoding the messages to come from this set S, we are able to both encode and compute the intersection of the code with the output subspace of Proposition 6.19 in polynomial time.

Setting $s = O(1/\varepsilon)$ and $\ell = O(s/\varepsilon)$, we obtain the following.

Corollary 6.20. For every $\varepsilon > 0$ and $R \in (0,1)$, there is an explicit rate R subcode of a low-order folded Reed-Solomon code which is list-decodable from a $1 - R - \varepsilon$ fraction of errors with list size $(1/\varepsilon)^{O(1/\varepsilon)}$, given an irreducible polynomial satisfying the conditions of Proposition 6.19.

Remark. By using Corollary 6.12 instead of the results of [DL12], we can give a similar guarantee which yields a *linear* subcode, but with a larger list size guarantee of $q^{\mathrm{poly}(1/\varepsilon)}$.

The techniques of [GX13] using subspace designs could also be applied directly to the case of low-order folding, with a resulting list size of $n^{\mathrm{poly}(1/\varepsilon)}$. We are able to get an improvement using the observation that the space of candidates is actually a low-dimensional subspace over a much larger field.

6.4.3 Constructing high-degree irreducibles

The decoding algorithm of the previous section relied on working modulo a high-degree irreducible factor of $X^{q^a} - \zeta X$. In what follows, we consider the problem of finding such a factor efficiently.

Proposition 6.21. For $\zeta \in \mathbb{F}_q$ of order ℓ, the irreducible factors over $\mathbb{F}_q[X]$ of

$$X^{q^a-1} - \zeta$$

have degree dividing $a\ell$. In particular, all roots of $X^{q^a-1} - \zeta$ lie in $\mathbb{F}_{q^{a\ell}}$.

Proof. As $X^{(q^a-1)\ell} \equiv 1 \pmod{X^{q^a-1}-\zeta}$, it is enough to see that $(q^a-1)\ell$ divides $q^{a\ell}-1$. This implies that $X^{q^a-1} - \zeta$, and thus all of its irreducible factors, divides $X^{q^{a\ell}} - X$.

Consider

$$\frac{q^{a\ell}-1}{q^a-1} = q^{a(\ell-1)} + q^{a(\ell-2)} + \cdots + q^a + 1.$$

As $\ell \mid q-1$, and there are ℓ terms on the right-hand side, the entire quantity is equivalent to $0 \pmod{\ell}$, and so $(q^a-1)\ell$ divides $q^{a\ell}-1$, as desired. □

Corollary 6.22. If a and ℓ with $a > 2\ell$ are distinct primes, at least half of the roots of $X^{q^a-1} - \zeta$ have minimal polynomials of degree $a\ell$.

Proof. By Proposition 6.21, all irreducible factors of $X^{q^a-1} - \zeta$ have degrees in the set $\{1, a, \ell, a\ell\}$. No irreducible factor has degree 1 or a, because any irreducible of degree 1 or a divides $X^{q^a-1} - 1$ and therefore does not divide $X^{q^a-1} - \zeta$ for $\zeta \neq 1$.

Because $X^{q^a-1} - \zeta$ has no repeated factors, it has at most q^ℓ roots which lie in $\mathbb{F}_{q^\ell}$ (and hence have minimal polynomials of degree ℓ.

Thus, under the assumptions on a and ℓ, $X^{q^a-1} - \zeta$ has at least $(q^a - q^\ell - 1) \geq q^\ell$ roots of degree $a\ell$. Thus at least half of of $X^{q^a-1} - \zeta$'s roots have minimal polynomials of degree $a\ell$. □

In particular, by choosing a to be a prime in the range $[k/\ell, 2k/\ell]$, we have $k \leq a\ell \leq 2k$, so that an irreducible factor of $X^{q^a-1} - \zeta$ will satisfy the conditions of Proposition 6.19. The next section will show that we cannot hope to improve much on the value of a.

Given a value for a for which $X^{q^a-1} - \zeta$ has many degree $a\ell$ factors, the problem remains to compute one. In what follows, we describe one randomized approach.

Recall that a and ℓ are primes, and that we are trying to find a degree $a\ell$ factor of $X^{q^a-1} - \zeta$. The idea is to sample a root of $X^{(q^a-1)\ell} - 1$. Consider the following procedure:

1. Sample $\beta \in (\mathbb{F}_{q^a})^*$ uniformly at random.
2. Compute the roots $\rho_1, \ldots, \rho_\ell$ of $X^\ell - \beta$, which lie in $\mathbb{F}_{q^{a\ell}}$ by Proposition 6.21. This can be done in time $\tilde{O}(n^2 \log(q^a) \log^{-1} \varepsilon)$ with failure probability ε using a variant of Berlekamp's algorithm (see, for example, [Kal92]).
3. Compute $\rho_i^{q^a-1}$ for each i and output the minimal polynomial of ρ_i over $\mathbb{F}_q$ if $\rho_i^{q^a-1} = \zeta$.

First note that steps 1–2 sample each root of $X^{(q^a-1)\ell} - 1$ uniformly. Each ρ_i computed in step 2 satisfies $\rho_i^\ell \in (\mathbb{F}_{q^a})^*$, so ρ_i is a root of $X^{(q^a-1)\ell} - 1$. Conversely, each nonzero β yields ℓ distinct roots of $X^\ell - \beta$, which are distinct for distinct β, yielding $(q^a - 1)\ell$ roots.

Therefore, with probability $1/\ell$, we will find a root ρ of $X^{q^a-1} - \zeta$. By Corollary 6.22, ρ's minimal polynomial has degree $a\ell$ with probability at least $1/2$.

We can thus conclude that, with probability at least $\frac{1}{2\ell} - \varepsilon$, we find an irreducible factor of $X^{q^a-1} - \zeta$ of degree $a\ell$.

6.4.4 Relationship to Reed-Solomon list-decoding

The original motivation for studying low-order folding was the following reduction from Reed-Solomon codes.

Given a polynomial f of degree $< k/\ell$ evaluated at distinct points $1, \gamma^\ell, \gamma^{2\ell}, \ldots, \gamma^{N\ell}$, we can think of it as a degree $< k$ polynomial $g(X) = f(X^\ell)$. For ζ of order ℓ, we have that $g(\zeta^i X) = g(X)$ for every i. In particular, the associated low-order folded Reed-Solomon codeword encoding $g(X)$ is simply

$$\begin{bmatrix} f(1) & f(\gamma^\ell) & \ldots & f(\gamma^{N\ell}) \\ f(1) & f(\gamma^\ell) & \ldots & f(\gamma^{N\ell}) \\ \vdots & \vdots & \ddots & \vdots \\ f(1) & f(\gamma^\ell) & \ldots & f(\gamma^{N\ell}) \end{bmatrix}. \tag{6.19}$$

Notice that if $f(\gamma^{i\ell})$ is correct, then the entire ith column is correct, so an algorithm to list-decode the low-order folded RS code from an η fraction of errors will also list-decode the Reed-Solomon code with evaluation points $(1, \gamma^{\ell}, \ldots, \gamma^{N\ell})$ from the same error fraction.

This reduction also helps to show that the precoding used to conclude Corollary 6.20 is necessary for a polynomial list size. To see this, consider the behavior of the algorithm on a transmitted codeword as in Equation (6.19). If there is enough agreement, the algorithm will interpolate polynomials $A_i(X)$ satisfying

$$0 = A_0 + A_1(X)g(X) + A_2(X)g(\zeta X) + \cdots + A_s(X)g(\zeta^{s-1}X) \tag{6.20}$$

$$= A_0(X) + g(X)\sum_{i=1}^{s} A_i(X). \tag{6.21}$$

If $\sum_{i>0} A_i(X)$ is nonzero, then $g(X)$, and thus $f(X)$, can be recovered *uniquely* as $A_0(X)/\sum_{i>0} A_i(X)$; however, this will not be possible in general outside of the unique decoding radius. If $\sum_{i>0} A_i(X)$ is 0, then $A_0(X) = 0$ as well and *any* function which is a polynomial of X^{ℓ} satisfies Equation (6.21), and in particular the output list must have size at least $q^{k/\ell}$. Recall that ℓ is a constant in our application.

This implies that without precoding, the dimension of the list output by Proposition 6.19 over $\mathbb{F}_q$ must be $\Omega(k/\ell)$. Note that for the value $a = \theta(k/\ell)$ found in Section 6.4.3, the list size before precoding would be $O(ks/\ell)$.

Chapter 7

Conclusion and open questions

In which we wrap up this thesis • Alice and Bob look to the future

7.1 Summary

The central question of coding theory is always:

How can we communicate efficiently in the presence of errors?

In this thesis, we have seen how the tools of algebraic coding can be adapted to new settings, where they allow us to construct good codes for different models. We have studied:

Efficient coding for high error rates. In the model of list-decoding, we saw the power of algebraic folding, whether using shifted evaluation points or derivatives. We also saw the power of finding structure within our algorithms, and devising ways to evade this structure.

Our techniques give new algorithms for old codes, the folded Reed-Solomon codes, and new algorithms for new codes, the derivative codes. Our algorithms are simple, but they allow us to list-decode up to capacity, giving codes of rate R which can be decoded from a $1 - R - \varepsilon$ error rate for any positive ε.

We also observe that our algorithm outputs affine subspaces, which leads to constructions of *subspace-evasive sets* and improved list sizes.

Low-redundancy coding for deletions. In the model of deletions, we saw that even when our algebraic codes are not quite enough, we can combine them with better, if unwieldy

codes to obtain codes which achieve good trade-offs.

Using these ideas, we are able to construct efficient deletion codes which come close to the best known existential parameters in the extremal cases of high deletion fractions and high rate. Our work represents the first steps in a systematic study of what is achievable against adversarial deletions.

Efficient coding for network errors. In the models of subspace and rank-metric coding, we continued the theme of discovering structure, then avoiding it. We showed that for these models, our list-decoding techniques output subspaces which satisfy the strong property of being *periodic*, and we showed how to evade such subspaces.

Our work gives the first explicit constructions of subspace and rank-metric codes, based on algebraic codes, which are efficiently list-decodable with a constant rate. In fact, we are able to go even further, constructing such codes with the optimal trade-off between rate and correctable error fraction.

7.2 Next steps

Of course, coding theory is far from solved, and there are always new models and new questions. In this section, we focus on presenting some open directions which arise from the results presented in this thesis.

7.2.1 List-decoding

We saw in Chapter 3 that optimal-rate list-decoding can be achieved by "folding" together related Reed-Solomon evaluations. However, we know thanks to [RW14] that folding may not be necessary: there *exist* Reed-Solomon codes which can nearly achieve the same trade-offs. As in many such results, it turns out that a random code will work, and it is up to us to derandomize the result.

Question. Can we construct an explicit family of Reed-Solomon codes which is list-decodable beyond the Johnson bound?

On the subject of improving known code constructions, we can return to the (underwhelming) list sizes resulting from our FRS and derivative decoding algorithms.

Question. Can we improve the upper bounds on the list size for folded Reed-Solomon or derivative codes?

Recalling that we sidestepped this question in Chapter 3 by defining *subspace-evasive* sets, which remove some codewords from the code in order to improve the list size, we can aim to improve the parameters here as well.

Question. Can we give better explicit constructions of subspace-evasive sets? In particular, can we remove the exponential dependence of the intersection size on the dimension being evaded?

Note that we already showed this is possible using a random set; the difficulty is, as always, in making the construction efficient. Such a construction could also be useful in improving our constructions of rank-metric and subspace codes, where we constructed sets evading periodic subspaces.

Speaking of matching random parameters, we can also ask whether we can construct small-alphabet codes matching the parameters of Theorem 2.14.

Question. Can we give an explicit construction of codes of rate R which are list-decodable from a $1 - R - \varepsilon$ error fraction with list size $O(1/\varepsilon)$ and alphabet size $2^{O(1/\varepsilon)}$?

As we have mentioned, the construction of [GX12] comes tantalizingly close to achieving this goal by adapting the techniques presented in Chapter 3 to certain algebraic-geometric codes.

7.2.2 Deletion coding

There are still many things we don't know about deletions, both adversarial and random, and here we highlight some of the questions remaining for the adversarial case.

One of the most basic questions which is still open concerns the limits of binary deletion codes.

Question. For binary codes, what is the supremum p^* of all fractions p of adversarial deletions which are correctable with positive rate? Clearly $p^* \leq 1/2$; could it be that $p^* = 1/2$ and this trivial limit can be matched? Or is it the case that p^* is strictly less than $1/2$?

We showed in Chapter 4 that $p^* = 1/2$ if we allow *list-decoding*. Recent work in [BG15] shows that p^* is at least $1/3$, leaving a noticeable gap. Notice that this question is open even without requiring efficient encoding and decoding, although the constructions in Chapter 4 and [BG15] are both efficient.

It is also reasonable to ask whether our constructions of codes for high deletions and high rate can be improved. Recall that our construction suffered in parameters because of

the concatenation step.

Question. Can we construct codes of rate $1 - p - \gamma$ to efficiently correct a fraction p of deletions over an alphabet size that only depends on γ?

Note that this requires a relative distance of p, and currently we only know algebraic-geometric and expander-based codes which achieve such a trade-off between rate and relative distance.

Question. Can one improve the rate of the binary code construction to correct a fraction ε of deletions to $1 - \varepsilon \operatorname{poly}(\log(1/\varepsilon))$, approaching more closely the existential $1 - O(\varepsilon \log(1/\varepsilon))$ bound?

In the case of errors, an approach using expander graphs gives the analogous tradeoff (see [Gur04] and references therein). Could such an approach be adapted to the setting of deletions? The loss of positional information inherent in the deletion model seems to be particularly challenging in these expander-based constructions.

7.2.3 Rank-metric coding

As mentioned in Chapter 5, list-decoding for rank-metric codes poses unique challenges, not least of which is the existence of rank-metric codes which are not list-decodable at *any* radius beyond the unique decoding radius. Considering that even Gabidulin codes are not exempt from this strange behavior, we can only do our best to salvage the situation.

Question. Can we construct a family of Gabidulin codes which are list-decodable beyond half their minimum distance?

The negative results of [RWZ15] apply only in certain ranges of parameters, and there is some hope for codes outside of these parameters, for examples codes with very low rate. Given this, there is no harm in being ambitious.

Question. Can we construct a family of Gabidulin codes which are *efficiently* list-decodable beyond half their minimum distance?

Although we did give an algorithm which attempts to list-decode Gabidulin codes in Chapter 6, this approach has several drawbacks. Firstly, it only applies to codes where the evaluation points are drawn from a subfield, and secondly, the output list has exponential size. This means that a naïve pruning step would be too computationally expensive, and if we wanted to use this algorithm, we would also need a better way to prune the list down to polynomial size.

And, of course, we can seek to improve upon our old adversary, the list size.

Question. Can we give explicit codes of rate R which are list-decodable from a $1 - R - \varepsilon$ fraction of rank errors with list size $\mathrm{poly}(1/\varepsilon)$?

One possible approach, as mentioned in the Hamming metric case, would be to improve our constructions of subspace-evasive sets. These could then be applied to construct Gabidulin subcodes in a black-box manner. Note that, again, we know randomized constructions which achieve these parameters.

Bibliography

[ACLY00] R. Ahlswede, Ning Cai, S.-Y.R. Li, and R.W. Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, Jul 2000. 5.1, 5.1

[BG15] Boris Bukh and Venkatesan Guruswami. An improved bound on the fraction of correctable deletions. *Electronic Colloquium on Computational Complexity (ECCC)*, 2015. 7.2.2

[BGZ15] J. Brakensiek, V. Guruswami, and S. Zbarsky. Efficient low-redundancy codes for correcting multiple deletions, 2015. In preparation. 2.4.1

[BK03] Peter Braß and Christian Knauer. On counting point-hyperplane incidences. *Comput. Geom.*, 25(1-2):13–20, 2003. 3.3, 3.3

[Bra10] Kristian Brander. *Interpolation and list decoding of algebraic codes*. PhD thesis, Technical University of Denmark, 2010. 3.1.1

[BSKR10] E. Ben-Sasson, S. Kopparty, and J. Radhakrishnan. Subspace polynomials and limits to list decoding of reed-solomon codes. *Information Theory, IEEE Transactions on*, 56(1):113–120, Jan 2010. 2.3.1

[DG06] S. Diggavi and Matthias Grossglauser. On information transmission over a finite buffer channel. *Information Theory, IEEE Transactions on*, 52(3):1226–1237, March 2006. 2.4.1

[Din15] Yang Ding. On list-decodability of random rank metric codes and subspace codes. *Information Theory, IEEE Transactions on*, 61(1):51–59, Jan 2015. 5.4.1, 5.14

[DL12] Zeev Dvir and Shachar Lovett. Subspace evasive sets. In *Proceedings of the 44th ACM Symposium on Theory of Computing*, pages 351–358, 2012. 3.4, 3.19, 6.2.2, 6.11, 6.2.2, 6.4.2, 6.4.2

[Gab85] E. M. Gabidulin. Theory of codes with maximal rank distance. *Problems of Information Transmission*, 21(7):1–12, 1985. 5.3

[Gal61] R.G. Gallager. Sequential decoding for binary channels with noise and synchronization errors, October 1961. Lincoln Lab. Group Report. 2.4.1

[GHSZ02] V. Guruswami, J. Håstad, M. Sudan, and D. Zuckerman. Combinatorial bounds for list decoding. *IEEE Transactions on Information Theory*, 48(5):1021–1035, 2002. 3.3

[GK13] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 608–617, 2013. 6.2, 6.2.1, 6.13, 6.2.2

[GNW12] Venkatesan Guruswami, Srivatsan Narayanan, and Carol Wang. List decoding subspace codes from insertions and deletions. In *Proceedings of ITCS 2012*, pages 183–189, January 2012. 5, 5.4.1, 5.4.2, 6.3.2

[GR08a] V. Guruswami and A. Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008. 2.3.2, 2.3.2, 2.19, 3.1.1

[GR08b] Venkatesan Guruswami and Atri Rudra. Soft decoding, dual bch codes, and better list-decodable e-biased codes. In *Proceedings of the 2008 IEEE 23rd Annual Conference on Computational Complexity*, CCC '08, pages 163–174, Washington, DC, USA, 2008. IEEE Computer Society. 4.4.2, 4.20

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for multivariate polynomials. *Information Processing Letters*, 43(4):169–174, 1992. 3.2.1

[GS99] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and Algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999. 2.3.1, 1

[Gur03] V. Guruswami. List decoding with side information. In *Proceedings of the 18th IEEE Conference on Computational Complexity (CCC)*, pages 300–309, 2003. 3.1.3, 3.2.2

[Gur04] Venkatesan Guruswami. Guest column: error-correcting codes and expander graphs. *SIGACT News*, 35(3):25–41, 2004. 7.2.2

[GW13] Venkatesan Guruswami and Carol Wang. Linear-algebraic list decoding for variants of Reed-Solomon codes. *IEEE Transactions on Information Theory*, 59(6):3257–3268, 2013. 3

[GW14] Venkatesan Guruswami and Carol Wang. Deletion codes in the high-noise and high-rate regimes. *CoRR*, abs/1411.6667, 2014. 4

[GWX15] Venkatesan Guruswami, Carol Wang, and Chaoping Xing. List-decodable rank-metric and subspace codes via subspace designs, 2015. IEEE Transactions on Information Theory, submitted. 6

[GX12] Venkatesan Guruswami and Chaoping Xing. Folded codes from function field towers and improved optimal rate list decoding. In *Proceedings of the 44th ACM Symposium on Theory of Computing*, pages 339–350, 2012. 3.4, 7.2.1

[GX13] Venkatesan Guruswami and Chaoping Xing. List decoding Reed-Solomon, Algebraic-Geometric, and Gabidulin subcodes up to the Singleton bound. *Electronic Colloquium on Computational Complexity (ECCC), 19:146*, 2013. Extended abstract appeared in the *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC'13)*. 6.2.3, 6.4.2

[HKM+03] Tracey Ho, R. Koetter, M. Medard, D.R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. In *Information Theory, 2003. Proceedings. IEEE International Symposium on*, pages 442–, June 2003. 5.1

[Kal92] Erich Kaltofen. Polynomial factorization 1987–1991. *Proceedings of LATIN '92, LNCS*, 583:294–313, 1992. 2

[KK08] Ralf Koetter and Frank R. Kschischang. Coding for errors and erasures in random network coding. *IEEE Transactions on Information Theory*, 54(8):3579–3591, 2008. 5.2, 5.2, 5.2, 5.4, 5.2.1, 5.2.1, 5.8

[KLM04] Marcos Kiwi, Martin Loebl, and Jiří Matoušek. Expected length of the longest common subsequence for large alphabets. *Advances in Mathematics*, 197:480–498, November 2004. 4.1, 4.1

[KM13] Y. Kanoria and A. Montanari. Optimal coding for the binary deletion channel with small deletion probability. *Information Theory, IEEE Transactions on*, 59(10):6192–6219, Oct 2013. 2.4.1

[KMS10] Adam Kalai, Michael Mitzenmacher, and Madhu Sudan. Tight asymptotic bounds for the deletion channel with small deletion probabilities. In *ISIT*, pages 997–1001, 2010. 2.4.1

[KMTU11] I.A. Kash, M. Mitzenmacher, J. Thaler, and J. Ullman. On the zero-error capacity threshold for deletion channels. In *Information Theory and Applications Workshop (ITA), 2011*, pages 1–5, Feb 2011. 4.1

[Kop12] Swastik Kopparty. List-decoding multiplicity codes. *Electronic Colloquium on Computational Complexity, TR12-044*, 2012. 3.2.1

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-rate codes with sublinear-time decoding. *J. ACM*, 61(5):28, 2014. 3.2, 3.2.1

[Lev66] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Dokl. (English Translation)*, 10(8):707–710, 1966. 2.4.1

[Loi05] Pierre Loidreau. A Welch-Berlekamp like algorithm for decoding Gabidulin codes. In Øyvind Ytrehus, editor, *WCC*, volume 3969 of *Lecture Notes in Computer Science*, pages 36–45. Springer, 2005. 5.3

[MD06] Michael Mitzenmacher and Eleni Drinea. A simple lower bound for the capacity of the deletion channel. *IEEE Transactions on Information Theory*, 52(10):4657–4660, 2006. 2.4.1

[Mit09] Michael Mitzenmacher. A survey of results for deletion channels and related synchronization channels. *Probability Surveys*, 6:1–33, 2009. 2.4.1

[MV12] Hessam Mahdavifar and Alexander Vardy. List-decoding of subspace codes and rank-metric codes up to Singleton bound. *CoRR*, abs/1202.0866, 2012. 5.4.2, 5.4.2, 6.2.3, 6.3.2, 6.3.2

[PR04] Pavel Pudlák and Vojtech Rödl. Pseudorandom sets and explicit constructions of Ramsey graphs. In *Complexity of Computations and Proofs. Quad. Mat., 13, Dept. Math., Seconda Univ. Napoli, Caserta*, pages 327–346, 2004. 3.3

[PV05] Farzad Parvaresh and Alexander Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 285–294, 2005. 4.4.2

[RW14] Atri Rudra and Mary Wootters. Every list-decodable code for high noise has abundant near-optimal rate puncturings. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 764–773, New York, NY, USA, 2014. ACM. 2.3.1, 7.2.1

[RWZ15] Netanel Raviv and Antonia Wachter-Zeh. Some Gabidulin codes cannot be list decoded efficiently at any radius. *CoRR*, abs/1501.04272, 2015. 5.12, 7.2.3

[Sho88] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 283–290, 1988. 6.2.2

[SKK08] Danilo Silva, Frank R. Kschischang, and Ralf Koetter. A rank-metric approach to error control in random network coding. *IEEE Transactions on Information Theory*, 54(9):3951–3967, 2008. 5.3, 5.3

[Slo02] Neil J. A. Sloane. On single-deletion-correcting codes. *CoRR*, arxiv.org/abs/math/0207197, 2002. 2.4.1

[SZ99] Leonard Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, November 1999. 2.4.1, 2.4.3, 2.25, 4.3.1

[Vad11] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(13):1–336, 2011. 2.3.2

[WB86] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction of algebraic block codes. *US Patent Number 4,633,470*, December 1986. 3.2.1, 4.2

[Zig69] Kamil'Shamil'evich Zigangirov. Sequential decoding for a binary channel with drop-outs and insertions. *Problemy Peredachi Informatsii*, 5(2):23–30, 1969. 2.4.1