List Decoding and Property Testing of Error Correcting Codes

Atri Rudra

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

University of Washington

2007

Program Authorized to Offer Degree: Computer Science and Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Atri Rudra

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of the Supervisory Committee:

_____

Venkatesan Guruswami

Reading Committee:

_____

Paul Beame

_____

Venkatesan Guruswami

_____

Dan Suciu

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

List Decoding and Property Testing of Error Correcting Codes

Atri Rudra

Chair of the Supervisory Committee:
Associate Professor Venkatesan Guruswami
Department of Computer Science and Engineering

Error correcting codes systematically introduce redundancy into data so that the original information can be recovered when parts of the redundant data are corrupted. Error correcting codes are used ubiquitously in communication and data storage.

The process of recovering the original information from corrupted data is called decoding. Given the limitations imposed by the amount of redundancy used by the error correcting code, an ideal decoder should efficiently recover from as many errors as information-theoretically possible. In this thesis, we consider two relaxations of the usual decoding procedure: *list decoding* and *property testing*.

A list decoding algorithm is allowed to output a small list of possibilities for the original information that could result in the given corrupted data. This relaxation allows for efficient correction of significantly more errors than what is possible through usual decoding procedure which is always constrained to output the transmitted information.

- We present the first explicit error correcting codes along with efficient list-decoding algorithms that can correct a number of errors that approaches the information-theoretic limit. This meets one of the central challenges in the theory of error correcting codes.

- We also present explicit codes defined over smaller symbols that can correct significantly more errors using efficient list-decoding algorithms than existing codes, while using the same amount of redundancy.

- We prove that an existing algorithm for a specific code family called Reed-Solomon codes is optimal for "list recovery," a generalization of list decoding.

Property testing of error correcting codes entails "spot checking" corrupted data to quickly determine if the data is very corrupted or has few errors. Such spot checkers are closely related to the beautiful theory of Probabilistically Checkable Proofs.

- We present spot checkers that only access a nearly optimal number of data symbols for an important family of codes called Reed-Muller codes. Our results are the first for certain classes of such codes.

- We define a generalization of the "usual" testers for error correcting codes by endowing them with the very natural property of "tolerance," which allows slightly corrupted data to pass the test.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I moved to Seattle from Austin to work with Venkat Guruswami and I have never regretted my decision. This thesis is a direct result of Venkat's terrific guidance over the last few years. Venkat was very generous with his time and was always patient when I told him about my crazy ideas (most of the time gently pointing out why they would not work), for which I am very grateful. Most of the results in this thesis are the outcome of our collaborations. I am also indebted to Don Coppersmith, Charanjit Jutla, Anindya Patthak and David Zuckerman for collaborations that lead to portions of this thesis.

I would like to thank Paul Beame and Dan Suciu for agreeing to be on my reading committee and their thoughtful and helpful comments on the earlier drafts of this thesis. Thanks also to Anna Karlin for being on my supervisory committee.

I gratefully acknowledge the financial support from NSF grant CCF-0343672 and Venkat Guruswami's Sloan Research Fellowship.

My outlook on research has changed dramatically since my senior year at IIT Kharagpur when I was decidedly a non-theory person. Several people have contributed to this wonderful change and I am grateful to all of them. First, I would like to thank my undergraduate advisor P. P. Chakrabarti for teaching the wonderful course on Computational Complexity that opened my eyes to the beauty of theoretical computer science. My passion for theory was further cemented in the two wonderful years I spent at IBM India Research Lab. Thanks to Charanjit Jutla and Vijay Kumar for nurturing my nascent interest in theoretical computer science for being wonderful mentors and friends ever since. Finally, I am grateful to David Zuckerman for his wonderful courses on Graph theory and Combinatorics and Pseudorandomness at Austin, which gave me the final confidence to pursue theory.

I have been very lucky to have the privilege of collaborating with many wonderful researchers over the years. Thanks to all the great folks at IBM Research with whom I had the pleasure of working as a full time employee (at IBM India) as well as an intern (at IBM Watson and IBM Almaden). Thanks in particular to Nikhil Bansal, Don Coppersmith, Pradeep Dubey, Lisa Fleischer, Rahul Garg, T. S. Jayram, Charanjit Jutla, Robi Krauthgamer, Vijay Kumar, Aranyak Mehta, Vijayshankar Raman, J.R. Rao, Pankaj Rohatgi, Baruch Schieber, Maxim Sviridenko and Akshat Verma for the wonderful time I had during our collaborations.

My stay at UT Austin was wonderful and I am grateful to Anna Gál, Greg Plaxton and David Zuckerman for their guidance and our chats about sundry research topics. Thanks a bunch to my room mate and collaborator Anindya Patthak for the wonderful times. Also a big thanks to my friends in Austin for the best possible first one and a half years I could

have had in the US: Kartik, Sridhar, Peggy, Chris, Kurt, Peter, Walter, Nick, Maria.

I thought it would be hard to beat the friendly atmosphere at UT but things were as nice (if not better) at UW. A big thanks to Anna Karlin and Paul Beame for all their help and advice as well as the good times we had during our research collaborations. Special thanks to Paul and Venkat for their kind and helpful advice on what to do with the tricky situations that cropped up during my job search. I had a great time collaborating with my fellow students at UW (at theory night and otherwise): Matt Cary, Ning Chen, Neva Cherniavsky, Roee Engelberg, Thach Nguyen, Prasad Raghavendra, Ashish Sabharwal, Gyanit Singh and Erik Vee. Thanks to my office mate Eytan Adar and Chris Ré for our enjoyable chats. The CSE department at UW is an incredibly wonderful place to work– thanks to everyone for making my stay at UW as much as fun as it has been.

Thanks to my parents and sister for being supportive of all my endeavors. Finally, the best thing about my move to Seattle was that I met my wife here. Carole, thanks for everything: you are more than what this *desi* could have dreamed for in a wife.

DEDICATION

To my family, in the order I met them: Ma, Baba, Purba and Carole

Chapter 1

INTRODUCTION

Corruption of data is a fact of life. Error-correcting codes (or just codes) are clever ways of representing data so that one can recover the original information even if parts of it are corrupted. The basic idea is to judiciously introduce redundancy so that the original information can be recovered even when parts of the (redundant) data have been corrupted.

Perhaps the most natural and common application of error correcting codes is for communication. For example, when packets are transmitted over the Internet, some of the packets get corrupted or dropped. To deal with this, multiple layers of the TCP/IP stack use a form of error correction called CRC Checksum [87]. Codes are used when transmitting data over the telephone line or via cell phones. They are also used in deep space communication and in satellite broadcast (for example, TV signals are transmitted via satellite). Codes also have applications in areas not directly related to communication. For example, codes are used heavily in data storage. CDs and DVDs work fine even in presence of scratches precisely because they use codes. Codes are used in Redundant Array of Inexpensive Disks (RAID) [24] and error correcting memory [23]. Codes are also deployed in other applications such as paper bar codes, for example, the bar code used by UPS called MaxiCode [22].

In this thesis, we will think of codes in the communication scenario. In this framework, there is a sender who wants to send (say) k message symbols over a noisy channel. The sender first *encodes* the k message symbols into n symbols (called a *codeword*) and then sends it over the *channel*. The receiver gets a *received word* consisting of n symbols. The receiver then tries to *decode* and recover the original k message symbols. The main challenge in coding theory is to come up with "good" codes along with efficient encoding and decoding algorithms. In the next section, we will define more precisely the notion of codes and the noise model.

Typically, the definition of a code gives the encoding algorithm "for free." The decoding procedure is generally the more challenging algorithmic task. In this thesis, we concentrate more on the decoding aspect of the problem. In particular, we will consider two relaxations of the "usual" decoding problem in which either the algorithm outputs the original message that was sent or gives up (when too many errors have occurred). The two relaxations are called *list decoding* and *property testing*. The motivations for considering these two notions of decoding are different: list decoding is motivated by a well known limit on the number of errors one can decode from using the usual notion of decoding while property

testing is motivated by a notion of "spot-checking" of received words that has applications in complexity theory. Before we delve into more details of these notions, let us first review the basic definitions that we will need.

1.1 Basics of Error Correcting Codes

We will now discuss some of the basic notions of error correcting codes that are needed to put forth the contributions of this thesis.[1] These are the following.

- Encoding The *encoding function* with parameters k, n is a function $E : \Sigma^k \to \Sigma^n$, where Σ is called the *alphabet*. The encoding function E takes a *message* $m \in \Sigma^k$ and converts it into a *codeword* $E(m)$. We will refer to the algorithm that implements the encoding function as an *encoder*.

- Error Correcting Code An *error correcting code* or just a *code* corresponding to an encoding function E is just the image of the encoding function. In other words, it is the collection of all the codewords. A code C with encoding function $E : \Sigma^k \to \Sigma^n$ is said to have *dimension* k and *block length* n. In this thesis, we will focus on codes of large block length.

- Rate The ratio $R = k/n$ is called the *rate* of a code. This notion captures the amount of redundancy used in the code. This is an important parameter of a code which will be used throughout this thesis.

- Decoding Consider the basic setup for communication. A *sender* has a message that it sends as a codeword after encoding. During transmission the codeword gets distorted due to errors. The *receiver* gets a noisy *received word* from which it has to recover the original message. This "reverse" process of encoding is achieved via a *decoding function* $D : \Sigma^n \to \Sigma^k$. That is, given a received word, the decoding function picks a message that it thinks was the message that was sent. We will refer to the algorithm that implements the decoding function as a *decoder*.

- Distance The *minimum distance* (or just *distance*) of a code is a parameter that captures how much two different codewords differ. More formally, the distance between any two codewords is the number of coordinates in which they differ. The (minimum) distance of a code is the minimum distance between any two distinct codewords in the code.

---

[1] We will define some more "advanced" notions later.

1.1.1 Historical Background and Modeling the Channel Noise

The notions of encoding, decoding and the rate appeared in the seminal work of Shannon [94]. The notions of codes and the minimum distance were put forth by Hamming [67].

Shannon modeled the noise *probabilistically*. For such a channel, he also defined a real number called the *capacity*, which is an upper bound on the rate of a code for which one can have reliable communication. Shannon also proved the converse result. That is, there *exist* codes for any rate less than the capacity of the channel for which one can have reliable communication. This striking result essentially kick-started the fields of information theory and coding theory.

Perhaps an undesirable aspect of Shannon's noise model is that its effectiveness depends on how well the noise is modeled. In some cases it might not be possible to accurately model the channel. In such a scenario, one option is to model the noise *adversarialy*. This was proposed by Hamming. In Hamming's noise model, we think of the channel as an adversary who has the full freedom in picking the location as well as nature of errors to be introduced. The only restriction is on the number of errors. We will consider this noise model in the thesis.

Alphabet Size and the Noise Model

We would like to point out that the noise model is intimately tied with the alphabet. A symbol in the alphabet is the "atomic" unit on which the noise acts. In other words, a symbol that is fully corrupted and a symbol that is partially corrupted are treated as the same. That is, the smaller the size of the alphabet, the more fine-grained the noise. This implies that the decoder has to take care of more error patterns for a code defined over a smaller alphabet. As a concrete example, say we want to design a decoder that can handle 50% of errors. Consider a code C that is defined over an alphabet of size 4 (i.e., each symbols consists of two bits). Now, let e be an error pattern in which every alternate bit of a codeword in C is flipped. Note that this implies that *all* the symbols of the codeword have been corrupted and hence the decoder does not need to recover from e. However, if C were defined over the binary alphabet then the decoder would have to recover from e. Thus, it is harder to design decoders for codes over smaller alphabets.

Further, the noise introduced by the channel should be independent of the message length. However, in this thesis, we will study codes that are defined over alphabets whose size depends on the message length. In particular, the number of bits required to represent any symbol in the alphabet would be logarithmic in the message length. The reason for this is two-fold: As was discussed in the paragraph above, designing decoding algorithms is strictly easier for codes over larger alphabets. Secondly, we will use such codes as a starting point to design codes over fixed sized alphabets.

With the basic definitions in place, we now turn our attention to the two relaxations of the decoding procedure that will be the focus of this thesis.

1.2 List Decoding

Let us look at the decoding procedure in more detail. Upon getting the noisy received word, the decoder has to output a message (or equivalently a codeword) that it thinks was actually transmitted. If the output message is different from the message that was actually transmitted then we say that a decoding error has taken place. For the first part of the thesis, we will consider decoders that do not make any decoding error. Instead, we will consider the following notion called *unique decoding*. For any received word, a unique decoder either outputs the message that was transmitted by the sender or reports a decoding failure.

One natural question to ask is how many errors such a unique decoder can tolerate. That is, is there a bound on the number of errors (say $\rho_U n$, so ρ_U is the fraction of errors) such that for any error pattern with total error at most $\rho_U n$, the decoder always outputs the transmitted codeword?

We first argue that $\rho_U \leqslant 1 - R$. Note that the codeword of n symbols really contains k symbols of information. Thus, the receiver should have at least k uncorrupted symbols among the n symbols in the received word to have any hope of recovering the transmitted message. In other words, the information theoretic limit on the number of errors from which one can recover is $n - k$. This implies that $\rho_U \leqslant (n-k)/n = 1 - R$. Can this information theoretic limit be achieved ?

Before answering the question above, we argue that the limit also satisfies $\rho_U < d/(2n)$, where we assume that the distance of the code d is even. Consider two distinct messages m_1, m_2 such that the distance between $E(m_1)$ and $E(m_2)$ is exactly d. Now say that the sender sends the codeword $E(m_1)$ over the channel and the channel introduces $d/2$ errors and distorts the codeword into a received word $\mathbf{y}$ that is at a distance of $d/2$ from *both* $E(m_1)$ and $E(m_2)$ (see Figure 1.1).

Now, when the decoder gets $\mathbf{y}$ as an input it has no way of knowing whether the original transmitted codeword was $E(m_1)$ or $E(m_2)$.[2] Thus, the decoder has to output a decoding failure when it receives $\mathbf{y}$ and so we have $\rho_U < d/(2n)$. How far is $d/(2n)$ from the information theoretic bound of $1 - R$? Unfortunately the gap is quite big. By the so called Singleton bound, $d \leqslant n - k + 1$ or $d/n < 1 - R$. Thus, the limit of $d/(2n)$ is at most half the information theoretic bound. We note that even though the limits differ by "only a small constant," in practice the potential to correct twice the number of errors is a big gain.

Before we delve further into this gap between the information theoretic limit and half the distance bound, we next argue that the the bound of $d/2$ is in fact tight in the following sense. If $\rho_U n = d/2 - 1$, then for an error pattern with at most $\rho_U n$ errors, there is always a unique transmitted codeword. Suppose that this were not true and let $E(m_1)$ be the transmitted codeword and let $\mathbf{y}$ be the received word such that $\mathbf{y}$ is within distance $\rho_U n$ from both $E(m_1)$ and $E(m_2)$. Then by the triangle inequality, the distance between $E(m_1)$ and $E(m_2)$ is at most $2\rho_U n = d - 2 < d$, which contradicts the fact that d is the

[2]Throughout this thesis, we will be assuming that the only communication between the sender and the receiver is through the channel and that they do not share any side information/channel.

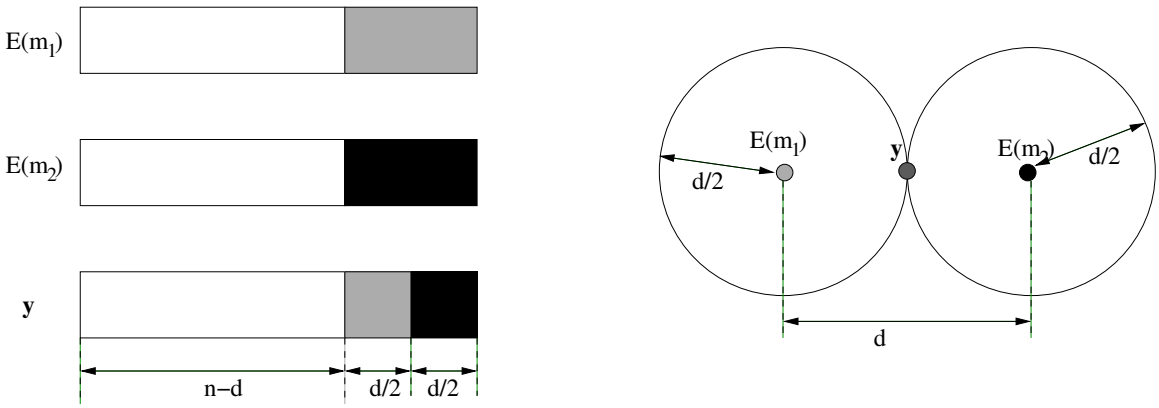


Figure 1.1: Bad example for unique decoding. The picture on the left shows two codewords $E(m_1)$ and $E(m_2)$ that differ in exactly d positions while the received word $\mathbf{y}$ differs from both $E(m_1)$ and $E(m_2)$ in $d/2$ many positions. The picture on the right is another view of the same example. Every n-symbol vector is now drawn on the plane and the distance between any two points is the number of positions they differ in. Thus, $E(m_1)$ and $E(m_2)$ are at a distance d and $\mathbf{y}$ is at a distance $d/2$ from both. Further, note that any point that is strictly contained within one of the balls of radius $d/2$ has a unique closest-by codeword.

minimum distance of the code (also see Figure 1.1). Thus, as long as $\rho_U n = d/2 - 1$, the decoder can output the transmitted codeword. So if one wants to do unique decoding then one can correct up to half the distance of the code (but no further). Due to this "half the distance barrier", much effort has been devoted to designing codes with as large a distance as possible.

However, all the discussion above has not addressed one important aspect of decoding. We argued that for $\rho_U n = d/2 - 1$, there *exists* a unique transmitted codeword. However, the argument sheds no light on whether the decoder can find such a codeword *efficiently*. Of course, before we can formulate the question more precisely, we need to state what we mean by efficient decoding. We will formulate the notion more formally later on but for now we will say that a decoder is efficient if its running time is polynomial in the block length of the code (which is the number of symbols in the received word). As a warm up, let us consider the following naive decoding algorithm. The decoder goes through all the codewords in the code and outputs the codeword that is closest to the received word. The problem with this brute-force algorithm is that its running time is exponential in the block length for constant rate codes (which will be the focus of the first part of the thesis) and thus, is not an efficient algorithm. There is a rich body of beautiful work that focuses on designing efficient algorithms for unique decoding for many families of codes. These are discussed in detail in any standard coding theory texts such as [80, 104].

We now return to the gap between the half the distance and the information theoretic limit of $n - k$.

1.2.1 Going Beyond Half the Distance Bound

Let us revisit the bound of half the minimum distance on unique decoding. The bound follows from the fact that there exists an error pattern for which one cannot do unique decoding. However, such bad error patterns are rare. This follows from the nature of the space that the codewords (and the received words) "sit" in. In particular, one can think of a code of block length n as consisting of non-overlapping spheres of radius $d/2$, where the codewords are the centers of the spheres (see Figure 1.2). The argument for half the distance bound uses the fact that at least two such spheres touch. The touching point corresponds to the received word $\mathbf{y}$ that was used in the argument in the last section. However, the way the spheres pack in high dimension (recall the dimension of such a space is equal to the block length of the code n), almost every point in the ambient space has a unique by closest codeword at distances well beyond $d/2$ (see Figure 1.2).

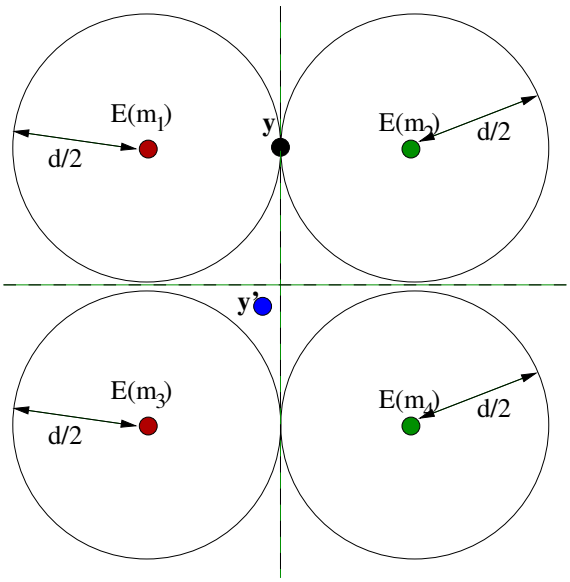


Figure 1.2: Four close by codewords $E(m_1), E(m_2), E(m_3)$ and $E(m_4)$ with two possible received words $\mathbf{y}$ and $\mathbf{y}'$. $E(m_1), E(m_2)$ and $\mathbf{y}$ form the bad example of Figure 1.1. However, the bad examples lie on the dotted lines. For example, $\mathbf{y}'$ is at a distance more than $d/2$ from its (unique) closest codewords $E(m_3)$. In high dimension, the space outside the balls of radius $d/2$ contains almost the entire ambient space.

Thus, by insisting on *always* getting back the original codeword, we are giving up on

correcting from error patterns from which we can recover the original codeword. One natural question one might ask is if one can somehow meaningfully relax this stringent constraint.

In the late 1950s, Elias and Wozencraft independently proposed a nice relaxation of unique decoding that gets around the barrier of half the distance bound [34, 106]. Under *list decoding*, the (list) decoder needs to output a "small" list of answers with the guarantee that the transmitted codeword is present in the list.[3] More formally, for a given error bound ρn and a received word $\mathbf{y}$, the list-decoding algorithm has to output all codewords that are at a distance at most ρn from $\mathbf{y}$. Note that when ρn is an upper bound on the number of errors that can be introduced by the channel, the list returned by the list-decoding algorithm will have the transmitted codeword in the list.

There are two immediate questions that arise: (i) Is list decoding a useful relaxation of unique decoding? (ii) Can we correct a number of errors that is close to the information theoretic limit using list decoding ?

Before we address these questions, let us first concentrate on a new parameter that this new definition throws into the mix: the worst case list size. Unless mentioned otherwise, we will use L to denote this parameter. Note that the running time of the decoding algorithm is $\Omega(L)$ as the decoder has to output every codeword in the list. Since we are interested in efficient, polynomial time, decoding algorithms, this puts an *a priori* requirement that L be a polynomial in the block length of the code. For a constant rate code, which has exponentially many codewords, the polynomial bound on L is very small compared to the total number of codewords. This bound was what we meant by small lists while defining list decoding.

Maximum Likelihood Decoding

We would like to point out that list decoding is not the only meaningful relaxation of unique decoding. Another relaxation called *maximum likelihood decoding* (or MLD) has been extensively studied in coding theory. Under MLD, the decoder must output the codeword that is closest to the received word. Note that if the number of errors is at most $(d-1)/2$, then MLD and unique decoding coincide. Thus, MLD is indeed a generalization of unique decoding.

MLD and list decoding are incomparable relaxations. On the one hand, if one can list decode efficiently up to the maximum number of errors that the channel can introduce then one can do efficient MLD. On the other hand, MLD does not put any restriction on the number of errors it needs to tolerate (whereas such a restriction is necessary for efficient list decoding). The main problem with MLD is that is turns out to be computationally intractable in general [17, 79, 4, 31, 37, 91] as well as for specific families of codes [66]. In

[3]The condition on the list size being small is important. Otherwise, here is a trivial list-decoding algorithm: output all codewords in the code. This, however is a very inefficient and more pertinently a useless algorithm. We will specify more carefully what we mean by small lists soon.

fact, there is no non-trivial family of codes known for which MLD can be done in polynomial time. However, list decoding is computationally tractable for many interesting families of codes (some of which we will see in this thesis).

We now turn to the questions that we raised about list decoding.

1.2.2 Why is List Decoding Any Good ?

We will now devote some time to address the question of whether list decoding is a meaningful relaxation of the unique decoding problem. Further, what does one do when the decoder outputs a list ?

In the communication setup, where the receiver might not have any side information, the receiver can still use a list-decoding algorithm to do "normal" decoding. It runs the list-decoding algorithm on the received word. If the list returned has just one codeword in it, then the receiver accepts that codeword as the transmitted codeword. If the list has more than one codeword, then it declares a decoding failure. First we note that this is no worse than the original unique decoding setup. Indeed if the number of errors is at most $d/2 - 1$, then by the discussion in Section 1.2 the list is going to contain one codeword and we would be back in the unique decoding regime. However, as was argued in the last section, for *most* error patterns (with total number of errors well beyond $d/2$) there is a unique closest by codeword. In other words, the list size for such error patterns would be one. Thus, list decoding allows us to correct from more error patterns than what was possible with unique decoding.

We now return to the question of whether list decoding can allow us to correct errors up to the information theoretic limit of $1 - R$? In short, the answer is yes. Using random coding arguments one can show that for any $\varepsilon > 0$, with high probability a random code of rate R, has the potential to correct up to $1 - R - \varepsilon$ fraction of errors with a worst case list size of $O(1/\varepsilon)$ (see Chapter 2 for more details). Further, one can show that for such codes, the list size is one for *most* received words.[4]

Other Applications of List Decoding

In addition to the immense practical potential of correcting more than half the distance number of errors in the communication setup, list decoding has found many surprising applications outside of the coding theory domain. The reader is referred to the survey by Sudan [98] and the thesis of Guruswami [49] (and the references therein) for more details on these applications. A key feature in all these applications is that there is some side information that one can use to sift through the list returned by the list-decoding algorithm to pick the "correct" codeword. A good analogy is that of a spell checker. Whenever a word is mis-spelt, the spell checker returns to the user a list of possible words that the user might have intended to use. The user can then prune this list to choose the word that he or she had

[4]This actually follows using the same arguments that Shannon used to establish his seminal result.

intended to use. Indeed, even in the communication setup, if the sender and the receiver can use a side channel (or have some shared information) then one can use list decoding to do "unambiguous" decoding [76].

1.2.3 The Challenge of List Decoding (and What Was Already Known)

In the last section, we argued that list decoding is a meaningful relaxation of unique decoding. More encouragingly, we mentioned that random codes have the potential to correct errors up to the information theoretic limit using list decoding. However, there are two major issues with the random codes result. First, these codes are not explicit. In real world applications, if one wants to communicate messages then one needs an explicit code. However, depending on the application, one might argue that doing a brute force search for such a code might work as this is a "one-off" cost that one has to pay. The second and perhaps more serious drawback is that the lack of structure in random codes implies that it is hard to come up with efficient list decodable algorithms for such codes. Note that for decoding, one cannot use a brute-force list-decoding algorithm.

Thus, the main challenge of list decoding is to come up with explicit codes along with efficient list-decoding (and encoding) algorithms that can correct errors close to the information theoretic limit of $n - k$.

The first non-trivial list-decoding algorithm is due to Sudan [97], which built on the results in [3]. Sudan devised a list-decoding algorithm for a specific family of codes called Reed-Solomon codes [90] (widely used in practice [105]), which could correct beyond half the distance barrier for Reed-Solomon codes of rate at most $1/3$. This result was then extended to work for all rates by Guruswami and Sudan [63]. It is worthwhile to note that even though list decoding was introduced in the late 1950s, these results came nearly forty years later. There was no improvement to the Guruswami-Sudan result until the recent work of Parvaresh and Vardy [85], who designed a code that is related to Reed-Solomon codes and presented efficient list-decoding algorithms that could correct more errors than the Guruswami-Sudan algorithm. However, the result of Parvaresh and Vardy does not meet the information theoretic limit (see Chapter 3 for more details). Further, for list decoding Reed-Solomon codes there has been no improvement over [63].

This concludes our discussion on the background for list decoding. We now turn to another relaxation of decoding that constitutes the second part of this thesis.

1.3 Property Testing of Error Correcting Codes

Consider the following communication scenario in which the channel is very noisy. The decoder, upon getting a very noisy received word, does its computation and ultimately reports a decoding failure. Typically, the decoding algorithm is an expensive procedure and it would be nice if one could quickly test if the received word is "far" from any codeword (in which case it should reject the received word) or is "close" to some codeword (in which case

it should accept the received word). In the former case, we would not run our expensive decoding algorithm and in the latter case, we would then proceed to run the decoding algorithm on the received word.

The notion of efficiency that we are going to consider for such spot checkers is going to be a bit different from that of decoding algorithms. We will require the spot checker to probe only a few positions in the received word during the course of its computation. Intuitively this should be possible as spot checking is a strictly easier task than decoding. Further, the fact that the spot checkers need to make their decision based on a portion of the received word should make spot checking very efficient. For example, if one could design spot checkers that look at only constant many positions (independent of the block length of the code), then we would have a spot checkers that run in constant time. However, note that since the spot checker cannot look at the whole received word it cannot possibly predict accurately if the received word is "far" from all the codewords or is "close" to some codeword. Thus, this notion of testing is a relaxation of the usual decoding as one sacrifices in the accuracy of the answer while gaining in terms of number of positions that one needs to probe.

A related notion of such spot checkers is that of locally testable codes (LTCs). LTCs have been the subject of much research over the years and there has been heightened activity and progress on them recently [46, 11, 74, 14, 13, 44]. LTCs are codes that have spot checkers as those discussed above with one crucial difference: they only need to differentiate between the cases when the received word is far from all codewords and the case when it *is* a codeword. LTCs arise in the construction of Probabilistically Checkable Proofs (PCPs) [5, 6] (see the survey by Goldreich [44] for more details on the interplay between LTCs and PCPs). Note that in the notion of LTC, there is no requirement on the spot checker for input strings that are very close to a codeword. This "asymmetry" in the way the spot checker accepts and rejects an input reflects the way PCPs are defined, where the emphasis is on rejecting "wrong" proofs.

Such spot checkers fall under the general purview of *property testing* (see for example the surveys by Ron [92] and Fischer [38]). In property testing, for some property P, given an object as an input, the spot checker has to decide if the given object satisfies the property P or is "far" from satisfying P. LTCs are a special case of property testing in which the property P is membership in some code and the objects are received words.

The ideal LTCs are codes with constant rate and linear distance that can be tested by probing only constant many position in the received word. However, unlike the situation in list decoding (where one can show the existence of codes with the "ideal" properties), it is *not known* if such LTCs exist.

1.3.1 A Brief History of Property Testing of Codes

The field of codeword testing, which started with the work of Blum, Luby and Rubinfeld [21] (who actually designed spot checkers for a variety of numerical problems), later developed into the broader field of property testing [93, 45]. LTCs were first explicitly de-

fined in [42, 93] and the systematic study of whether ideal LTCs (as discussed at the end of the last section) was initiated in [46]. Testing for Reed-Muller codes in particular has garnered a lot of attention [21, 9, 8, 36, 42, 93, 7, 1, 74], as they were crucial building blocks in the construction of PCPs [6, 5], Kaufman and Litsyn [73] gave a sufficient condition on an important class of codes that imply that the code is an LTC. Ben-Sasson and Sudan [13] built LTCs from a variant of PCPs called the Probabilistically Checkable Proof of Proximity– this "method" of constructing LTCs was initiated by Ben-Sasson et al. [11].

1.4 Contributions of This Thesis

The contributions of this thesis are in two parts. The first part deals with list decoding while the second part deals with property testing of codes.

1.4.1 List Decoding

This thesis advances our understanding of list decoding. Our results can be roughly divided into three parts: (i) List decodable codes of optimal rate over large alphabets, (ii) List decodable codes over small alphabets, and (iii) Limits to list decodability. We now look at each of these in more detail.

List Decodable Codes over Large Alphabets

Recall that for codes of rate R, it is information theoretically not possible to correct beyond $1-R$ fraction of errors. Further, using random coding argument one can show the existence of codes that can correct up to $1-R-\varepsilon$ fraction of errors for any $\varepsilon > 0$ (using list decoding). Since the first non-trivial algorithm of Sudan [97], there has been a lot of effort in designing explicit codes along with efficient list-decoding algorithms that can correct errors close to the information theoretic limit. In Chapter 3, we present the culmination of this line of work by presenting explicit codes (which are in turn extensions of Reed-Solomon codes) along with polynomial time list-decoding algorithm that can correct $1 - R - \varepsilon$ fraction of errors in polynomial time (for every rate $0 < R < 1$ and any $\varepsilon > 0$). This answers a question that has been open for close to 50 years and meets one of the central challenges in coding theory.

This work was done jointly with Venkatesan Guruswami and was published in the proceedings of the 38th Symposium on Theory of Computing (STOC), 2006 [58] and is under review for the journal IEEE Transactions on Information Theory.

List Decodable Codes over Small Alphabets

The codes mentioned in the last subsection are defined over alphabets whose size increases with the block length of the code. As discussed in Section 1.1.1, this is not a desirable feature. In Chapter 4, we show how to use our codes from Chapter 3 along with known

techniques of *code concatenation* and *expander graphs* to design codes over alphabets of size $2^{O(\varepsilon^{-4})}$ that can still correct up to $1 - R - \varepsilon$ fraction of errors for any $\varepsilon > 0$. To get to within ε of the information theoretic limit of $n - k$, it is known that one needs an alphabet of size $2^{\Omega(\varepsilon^{-1})}$ (see Chapter 2 for more details).

However, if one were interested in codes over alphabets of fixed size, the situation is different. First, it is known that for fixed size alphabets, the information theoretic limit is much smaller than $n - k$ (see Chapter 2 for more details). Again, one can show that random codes meet this limit. In Chapter 4, we present explicit codes along with efficient list-decoding algorithms that correct errors up to the so called Blokh-Zyablov bound. These results are the currently best known via explicit codes, though the number of errors that can be corrected is much smaller than the limit achievable by random codes.

This work was done jointly with Venkatesan Guruswami and appears in two different papers. The first was published in in the proceedings of the 38th Symposium on Theory of Computing (STOC), 2006 [58] and is under review for the journal IEEE Transactions on Information Theory. The second paper will appear in the proceedings of the 11th International Workshop on Randomization and Computation (RANDOM) [60].

Explicit codes over fixed alphabets, considered in Chapter 4, are constructed using *code concatenation*. However, as mentioned earlier, the fraction of errors that such codes can tolerate via list decoding is far from the information theoretic limit. A natural question to ask is whether one can use concatenated codes to achieve the information theoretic limit? In Chapter 5 we give a positive answer to this question in following sense. We present a random ensemble of concatenated codes that with high probability meet the information theoretic limit: That is, they can potentially list decode as large a fraction of errors as general random codes, though with larger lists.

This work was done jointly with Venkatesan Guruswami and is an unpublished manuscript [61].

Limits to List Decoding Reed-Solomon Codes

The results discussed in the previous two subsections are of the following flavor. We know that random codes allow us to list decode up to a certain number of errors, and that is optimal. Can we design more explicit codes (maybe with efficient list-decoding algorithms) that can correct close to the number of errors that can be corrected by random codes? However, consider the scenario where one is constrained to work with a certain family of codes, say Reed-Solomon codes. Under this restriction what is the most number of errors from which one can hope to list decode?

The result of Guruswami and Sudan [63] says that one can efficiently correct up to $n - \sqrt{nk}$ many errors for Reed-Solomon codes. However, is this the best possible? In Chapter 6, we give some evidence that the Guruswami-Sudan algorithm might indeed be the best possible. Along the way we also give some explicit constructions of "bad list-decoding configurations." A bad list-decoding configuration refers to a received word $\mathbf{y}$ along with an

error bound ρ such that there are super-polynomial (in n) many Reed-Solomon codewords within a distance of ρn from $\mathbf{y}$.

This work was done jointly with Venkatesan Guruswami and appears in two different papers. The first was published in in the proceedings of the 37th Symposium on Theory of Computing (STOC), 2005 [56] as well as in the IEEE Transactions on Information Theory [59]. The second paper is an unpublished manuscript [62].

1.4.2 Property Testing

We now discuss our results on property testing of error correcting codes.

Testing Reed-Muller Codes

Reed-Muller codes are generalizations of Reed-Solomon codes. Reed-Muller codes are based on multivariate polynomials while Reed-Solomon codes are based on univariate polynomials. Local testing of Reed-Muller codes was instrumental in many constructions of PCPs. However, the testers were only designed for Reed-Muller codes over large alphabets. In fact, the size of the alphabet of such codes depends on the block length of the codes. In Chapter 7, we present near-optimal local testers for Reed-Muller codes defined over (a class of) alphabets of fixed size.

This work was done jointly with Charanjit Jutla, Anindya Patthak and David Zuckerman and was published in the proceedings of the 45th Symposium on Foundation of Computer Science (FOCS), 2005 [72] and is currently under review for the journal Random Structures and Algorithms.

Tolerant Locally Testable Codes

Recall that the notion of spot checkers that we were interested in had to accept the received word if it is far from all codewords and reject when it is close to some codeword (as opposed to LTCs, which only require to accept when the received word is a codeword). Surprisingly, such testers were not considered in literature before. In Chapter 8, we define such testers, which we call tolerant testers. Our results show that in general LTCs do not imply tolerant testability, though most LTCs that achieve the best parameters also have tolerant testers.

As a slight aside, we look at certain strong form of local testability (called *robust testability*) of certain *product of codes*. Product of codes are also special cases of certain concatenated codes considered in Chapter 4. We show that in general, certain product of codes cannot be robustly testable.

This work on tolerant testing was done jointly with Venkatesan Guruswami and was published in the proceedings of the 9th International Workshop on Randomization and Computation (RANDOM) [57]. The work on robust testability of product of codes is joint work with Don Coppersmith and is an unpublished manuscript [26].

1.4.3 Organization of the Thesis

We start with some preliminaries in Chapter 2. In Chapter 3, we present the main result of this thesis: codes with optimal rate over large alphabets. This result is then used to design new codes in Chapters 4 and 5. We present codes over small alphabets in Chapter 4, which are constructed by a combination of the codes from Chapter 3 and *code concatenation*. In Chapter 5, we show that certain random codes constructed by code concatenation also achieve the list-decoding capacity. In Chapter 6, we present some limitations to list decoding Reed-Solomon codes. We switch gears in Chapter 7 and present new local testers for Reed-Muller codes. We present our results on tolerant testability in Chapter 8. We conclude with the major open questions in Chapter 9.

Chapter 2

PRELIMINARIES

In this chapter we will define some basic concepts and notations that will be used throughout this thesis. We will also review some basic results in list decoding that will set the stage for our results. Finally, we will look at some specific families of codes that will crop up frequently in the thesis.

2.1 The Basics

We first fix some notation that will be used frequently in this work, most of which is standard.

For any integer $m \geqslant 1$, we will use $[m]$ to denote the set $\{1, \ldots, m\}$. Given positive integers n and m, we will denote the set of all length n vectors over $[m]$ by $[m]^n$. Unless mentioned otherwise, all vectors in this thesis will be row vectors. $\log x$ will denote the logarithm of x in base 2. $\ln x$ will denote the natural logarithm of x. For bases other than 2 and e, we will specify the base of the logarithm explicitly: for example logarithm of x in base q will be denoted by $\log_q x$.

A finite field with q elements will be denoted by $\mathbb{F}_q$ or $GF(q)$. For any real value x in the range $0 \leqslant x \leqslant 1$, we will use $H_q(x) = x \log_q(q-1) - x \log_q x - (1-x) \log_q(1-x)$ to denote the *q-ary entropy function*. For the special case of $q = 2$, we will simply use $H(x)$ for $H_2(x)$. For more details on the q-ary entropy function, see Section 2.2.2.

For any finite set S, we will use $|S|$ to denote the size of the set.

We now move on to the definitions of the basic notions of error correcting codes.

2.1.1 Basic Definitions for Codes

Let $q \geqslant 2$ be an integer.

Code, Blocklength, Alphabet size :

- An *error correcting code* (or simply a *code*) C is a subset of $[q]^n$ for positive integers q and n. The elements of C are called *codewords*.
- The parameter q is called the *alphabet size* of C. In this case, we will also refer to C as a *q-ary code*. When $q = 2$, we will refer to C as a *binary code*.
- The parameter n is called the *block length* of the code.

Dimension and Rate :

- For a q-ary code C, the quantity $k = \log_q |C|$ is called the *dimension* of the code (this terminology makes more sense for certain classes of codes called linear codes, which we will discuss shortly).

- For a q-ary code C with block length n, its *rate* is defined as the ratio $R = \frac{\log_q |C|}{n}$.

Often it will be useful to use the following alternate way of looking at a code. We will think of a q-ary code C with block length n and $|C| = M$ as a function $[M] \to [q]^n$. Every element x in $[M]$ is called a *message* and $C(x)$ is its *associated codeword*. If M is a power of q, then we will think of the message as length k-vector in $[q]^k$. Viewed this way, C provides a systematic way to add redundancy such that messages of length k over $[q]$ are mapped to n symbols over $[q]$.

(Minimum) Distance and Relative distance : Given any two vectors $\mathbf{v} = \langle v_1, \ldots, v_n \rangle$ and $\mathbf{u} = \langle u_1, \ldots, u_n \rangle$ in $[q]^n$, their *Hamming distance* (or simply distance), denoted by $\Delta(\mathbf{v}, \mathbf{u})$, is the number of positions that they differ in. In other words, $\Delta(\mathbf{v}, \mathbf{u}) = |\{i | u_i \neq v_i\}$.

- The *(minimum) distance* of a code C is the minimum Hamming distance between any two codewords in the code. More formally

$$\mathrm{dist}(C) = \min_{\substack{c_1, c_2 \in C, \\ c_1 \neq c_2}} \Delta(c_1, c_2).$$

- The *relative distance* of a code C of block length n is defined as $\delta = \frac{\mathrm{dist}(C)}{n}$.

2.1.2 Code Families

The focus of this thesis will be on the asymptotic performance of decoding algorithms. For such analysis to make sense, we need to work with an infinite family of codes instead of a single code. In particular, an infinite family of q-ary codes $\mathcal{C}$ is a collection $\{C_i | i \in \mathbb{Z}\}$, where for every i, C_i is a q-ary code of length n_i and $n_i > n_{i-1}$. The rate of the family $\mathcal{C}$ is defined as

$$R(\mathcal{C}) = \liminf_i \left\{ \frac{\log_q |C_i|}{n_i} \right\}.$$

The relative distance of such a family is defined as

$$\delta(\mathcal{C}) = \liminf_i \left\{ \frac{\mathrm{dist}(C_i)}{n_i} \right\}.$$

From this point on, we will overload notation by referring to an infinite family of codes simply as a code. In particular, from now on, whenever we talk a code C of length n, rate

R and relative distance δ, we will implicitly assume the following. We will think of n as large enough so that its rate R and relative distance δ are (essentially) same as the rate and the relative distance of the corresponding infinite family of codes.

Given this implicit understanding, we can talk about the asymptotics of different algorithms. In particular, we will say that an algorithm that works with a code of block length n is *efficient* if its running time is $O(n^c)$ for some fixed constant c.

2.1.3 Linear Codes

We will now consider an important sub-class of codes called linear codes.

Definition 2.1. *Let q be a prime power. A q-ary code C of block length n is said to be* linear *if it is a linear subspace (over some field $\mathbb{F}_q$) of the vector space $\mathbb{F}_q^n$.*

The size of a q-ary linear code is obviously q^k for some integer k. In fact, it is the dimension of the corresponding subspace in $\mathbb{F}_q^n$. Thus, the dimension of the subspace is same as the dimension of the code. (This is the reason behind the terminology of dimension of a code.)

We will denote a q-ary linear code of dimension k, length n and distance d as an $[n, k, d]_q$ code. (For a general code with the same parameters, we will refer to it as an $(n, k, d)_q$ code.) Most of the time, we will drop the distance part and just refer to the code as an $[n, k]_q$ code. Finally, we will drop the dependence on q if the alphabet size is clear from the context.

We now make some easy observations about q-ary linear codes. First, the zero vector is always a codeword. Second, the minimum distance of a linear code is equal to the minimum *Hamming weight* of the non-zero codewords, where the Hamming weight of a vector is the number of positions with non-zero values.

Any $[n, k]_q$ code C can be defined in the following two ways.

- C can be defined as a set $\{\mathbf{x}\mathbf{G} | \mathbf{x} \in \mathbb{F}_q^k\}$, where $\mathbf{G}$ is an $k \times n$ matrix over $\mathbb{F}_q$. $\mathbf{G}$ is called a *generator matrix* of C.

- C can also be characterized by the following subspace $\{\mathbf{c} | \mathbf{c} \in \mathbb{F}_q^n \text{ and } H\mathbf{c}^T = \mathbf{0}\}$, where H is an $(n - k) \times n$ matrix over $\mathbb{F}_q$. H is called the *parity check matrix* of C. The code with H as its generator matrix is called the *dual* of C and is generally denoted by $C^\perp$.

The above two representations imply the following two things for an $[n, k]_q$ code C. First, given the generator matrix $\mathbf{G}$ and a message $\mathbf{x} \in \mathbb{F}_q^n$, one can compute $C(x)$ using $O(nk)$ field operations (by multiplying $\mathbf{x}^T$ with $\mathbf{G}$). Second, given a received word $\mathbf{y} \in \mathbb{F}_q^n$ and the parity check matrix H for C, one can check if $\mathbf{y} \in C$ using $O(n(n-k))$ operations (by computing $H\mathbf{y}$ and checking if it is the all zeroes vector).

Finally, given a q-ary linear code C, we can define the following equivalence relation. $\mathbf{x} \equiv_C \mathbf{y}$ if and only if $\mathbf{x} - \mathbf{y} \in C$. It is easy to check that since C is linear this indeed is an equivalence relation. In particular, $\equiv_C$ partitions $\mathbb{F}_q^n$ into equivalence classes. These are called *cosets* of C (note that one of the cosets is the code C itself). In particular, every coset is of the form $\mathbf{y} + C$, where either $\mathbf{y} = \mathbf{0}$ or $\mathbf{y} \notin C$ and $\mathbf{y} + C$ is shorthand for $\{\mathbf{y} + \mathbf{c} | \mathbf{c} \in C\}$.

We are now ready to talk about definitions and preliminaries for list decoding and property testing of codes.

2.2 Preliminaries and Definitions Related to List Decoding

Recall that list decoding is a relaxation of the decoding problem, where given a received word, the idea is to output all "close-by" codewords. More precisely, given an error bound, we want to output all codewords that lie within the given error bound from the received word. Note that this introduces a new parameter into the mix: the worst case list size. We will shortly define the notion of list decoding that we will be working with in this thesis.

Given integers $q \geqslant 2$, $n \geqslant 1$, $0 \leqslant e \leqslant n$ and a vector $\mathbf{x} \in [q]^n$, we define the *Hamming ball* around $\mathbf{x}$ of *radius* e to be the set of all vectors in $[q]^n$ that are at Hamming distance at most e from $\mathbf{x}$. That is,

$$B_q(\mathbf{x}, e) = \{\mathbf{y} | \mathbf{y} \in [q]^n \textit{ and } \Delta(\mathbf{y}, \mathbf{x}) \leqslant e\}.$$

We will need the following well known result.

Proposition 2.1 ([80]). *Let $q \geqslant 2$ and $e, n \geqslant 1$ be integers such that $e \leqslant (1 - 1/q)n$. Define $\rho = e/n$. Then the following relations are satisfied.*

$$|B_q(\mathbf{0}, e)| = \sum_{i=0}^{e} \binom{n}{i} (q-1)^i \leqslant q^{H_q(e/n)n} = q^{H_q(\rho)n}. \tag{2.1}$$

$$|B_q(\mathbf{0}, e)| \geqslant q^{H_q(\rho)n - o(n)}. \tag{2.2}$$

We will be using the following definition quite frequently.

Definition 2.2 (List-Decodable Codes). *Let C be a q-ary code of block length n. Let $L \geqslant 1$ be an integer and $0 < \rho < 1$ be a real. Then C is called (ρ, L)-list decodable if every Hamming ball of radius ρn has at most L codewords in it. That is, for every $\mathbf{y} \in \mathbb{F}_q^n$, $|B(\mathbf{y}, \rho n) \cap C| \leqslant L$.*

In the definitions above, the parameter L can depend on the block length of the code. In such cases, we will explicitly denote the list size by $L(n)$, where n is the block length.

We will also frequently use the notion of list-decoding radius, which is defined next.

Definition 2.3 (List-Decoding Radius). *Let C be a q-ary code of block length n. Let $0 < \rho < 1$ be a real and define $e = \rho n$. C is said to have a list-decoding radius of ρ (or e) with list size L if ρ (or e) is the* maximum *value for which C is (ρ, L)-list decodable.*

We will frequently use the term *list-decoding radius* without explicitly mentioning the list size in which case the list size is assumed to be at most some fixed polynomial in the block length. Note that one way to show that a code C has a list-decoding radius of at least ρ is to present a polynomial time list-decoding algorithm that can list decode C up to a ρ fraction of errors. Thus, by abuse of notation, given an efficient list-decoding algorithm for a code that can list decode a ρ fraction (or e number) of errors, we will say that the list-decoding algorithm has a list-decoding radius of ρ (or e). In most places, we will be exclusively talking about list-decoding algorithms in which case we will refer to their list-decoding radius as *decoding radius* or just *radius*. In such a case, the code under consideration is said to be list decodable up to the corresponding decoding radius (or just radius). Whenever we are talking about a different notion of decoding (say unique decoding), we will refer to the maximum fraction of errors that a decoder can correct by qualifying the decoding radius with the specific notion of decoding (for example *unique decoding radius*).

We will also use the following generalization of list decoding.

Definition 2.4 (List-Recoverable Codes). *Let C be a q-ary code of block length n. Let $\ell, L \geqslant 1$ be integers and $0 < \rho < 1$ be a real. Then C is called (ρ, ℓ, L)-list recoverable if the following is true. For every sequence of sets $S_1, \ldots, S_n$, where $S_i \subseteq [q]$ and $|S_i| \leqslant \ell$ for every $1 \leqslant i \leqslant n$, there are at most L codewords $\mathbf{c} = \langle c_1, \ldots, c_n \rangle \in C$ such that $c_i \in S_i$ for at least $(1 - \rho)n$ positions i.*

Further, code C is said to (ρ, ℓ)-list recoverable in polynomial time *if it is $(\rho, \ell, L(n))$-list recoverable for some polynomially bounded function $L(\cdot)$, and moreover there is a polynomial time algorithm to find the at most $L(n)$ codewords that are solutions to any $(\rho, \ell, L(n))$-list recovery instance.*

List recovery has been implicitly studied in several works; the name itself was coined in [52]. Note that a $(\rho, 1, L)$-list recoverable code is a (ρ, L)-list decodable code and hence, list recovery is indeed a generalization of list decoding. List recovery is useful in list decoding codes obtained by a certain code composition procedure. The natural list decoder for such a code is a two stage algorithm, where in the first stage the "inner" codes are list decoded to get a sequence of lists, from which one needs to recover codewords from the "outer" code(s). For such an algorithm to be efficient, the outer codes need to be list recoverable.

We next look at the most fundamental tradeoff that we would be interested in for list decoding.

2.2.1 Rate vs. List decodability

In this subsection, we will consider the following question. Given limits $L \geqslant 1$ and $0 < \rho < 1$ on the worst case list size and the fraction of errors that we want to tolerate, what

is the maximum rate that a (ρ, L)-list decodable code can achieve? The following results were implicit in [110] but were formally stated and proved in [35]. We present the proofs for the sake of completeness.

We first start with a positive result.

Theorem 2.1 ([110, 35]). *Let $q \geqslant 2$ be an integer and $0 < \delta \leqslant 1$ be a real. For any integer $L \geqslant 1$ and any real $0 < \rho < 1 - 1/q$, there exists a (ρ, L)-list decodable q-ary code with rate at least $1 - H_q(\rho) - \frac{1}{L+1} - \frac{1}{n^\delta}$.*

Proof. We will prove the result by using the probabilistic method [2]. Choose a code C of block length n and dimension $k = \lceil (1 - H_q(\rho) - \frac{1}{L+1})n - n^{1-\delta} \rceil$ at random. That is, pick each of the q^k codewords in C uniformly (and independently) at random from $[q]^n$. We will show that with high probability, C is (ρ, L)-list decodable.

Let $|C| = M = q^k$. We first fix the received word $\mathbf{y} \in [q]^n$. Consider an $(L+1)$-tuple of codewords $(\mathbf{c}^1, \ldots, \mathbf{c}^{L+1})$ in C. Now if all these codewords fall in a Hamming ball of radius ρn around $\mathbf{y}$, then C is not (ρ, L)-list decodable. In other words, this $(L+1)$-tuple forms a counter-example for C having the required list decodable properties. What is the probability that such an event happens ? For any fixed codeword $\mathbf{c} \in C$, the probability that it lies in $B(\mathbf{y}, \rho n)$ is exactly

$$\frac{|B(\mathbf{y}, \rho n)|}{q^n}.$$

Now since every codeword is picked independently, the probability that the tuple $(\mathbf{c}^1, \ldots, \mathbf{c}^{L+1})$ forms a counter example is

$$\left(\frac{|B(\mathbf{y}, \rho n)|}{q^n} \right)^{L+1} \leqslant q^{-(L+1)n(1-H_q(\rho))},$$

where the inequality follows from Proposition 2.1 (and the fact that the volume of a Hamming ball is translation invariant). Since there are $\binom{M}{L+1} \leqslant M^{L+1}$ different choices of $L+1$ tuples of codewords from C, the probability that there exists at least one $L+1$-tuple that lies in $B(\mathbf{y}, \rho n)$ is at most (by the union bound):

$$M^{L+1} \cdot q^{-(L+1)n(1-H_q(\rho))} = q^{-(L+1)n(1-H_q(\rho)-R)},$$

where $R = k/n$ is the rate of C. Finally, since there are at most q^n choices for $\mathbf{y}$, the probability that there exists some Hamming ball with $L+1$ codewords from C is at most

$$q^n \cdot q^{-(L+1)n(1-H_q(\rho)-R)} = q^{(L+1)n(1-H_q(\rho)-R-1/(L+1)} \leqslant q^{-n^{1-\delta}},$$

where the last inequality follows as $k/n \geqslant 1 - H_q(\rho) - 1/(L+1) - 1/n^\delta$. Thus, with probability $1 - q^{-n^{1-\delta}} > 0$ (for large enough n), C is a (ρ, L)-list decodable code, as desired. □

The following is an immediate consequence of the above theorem.

Corollary 2.2. *Let $q \geqslant 2$ be an integer and $0 < \rho < 1 - 1/q$. For every $\varepsilon > 0$, there exists a q-ary code with rate at least $1 - H_q(\rho) - \varepsilon$ that is $(\rho, O(1/\varepsilon))$-list decodable.*

We now move to an upper bound on the rate of good list decodable codes.

Theorem 2.3 ([110, 35]). *Let $q \geqslant 2$ be an integer and $0 < \rho \leqslant 1 - 1/q$. For every $\varepsilon > 0$, there do not exist any q-ary code with rate $1 - H_q(\rho) + \varepsilon$ that is $(\rho, L(n))$-list decodable for any function $L(n)$ that is polynomially bounded in n.*

Proof. The proof like that of Theorem 2.1 uses the probabilistic method. Let C be any q-ary code of block length n with rate $R = 1 - H_q(\rho) + \varepsilon$. Pick a received word $\mathbf{y}$ uniformly at random from $[q]^n$. Now, the probability that for some fixed $\mathbf{c} \in C$, $\Delta(\mathbf{y}, \mathbf{c}) \leqslant \rho n$ is

$$\frac{|B(\mathbf{0}, \rho n)|}{q^n} \geqslant q^{n(H_q(\rho)-1)-o(n)},$$

where the inequality follows from Proposition 2.1. Thus, the expected number of codewords within a Hamming ball of radius ρn around $\mathbf{y}$ is at least

$$|C| \cdot q^{n(H_q(\rho)-1)-o(n)} = q^{n(R-(1-H_q(\rho)))-o(n)},$$

which by the value of R is $q^{\Omega(n)}$. Since the expected number of codewords is exponential, this implies that there exists a received word $\mathbf{y}$ that has exponentially many codewords from C within a distance ρn from it. Thus, C cannot be $(\rho, L(n))$-list decodable for any polynomially bounded (in fact any subexponential) function $L(\cdot)$. □

List decoding capacity

Theorems 2.1 and 2.3 say that to correct a ρ fraction of errors using list decoding with small list sizes the best rate that one can hope for and can achieve is $1 - H_q(\rho)$. We will call this quantity the *list-decoding capacity*.

The terminology is inspired by the connection of the results above to Shannon's theorem for the special case of the q-symmetric channel (which we will denote by qSC_ρ). In this channel, every symbol (from $[q]$) remains untouched with probability $1 - \rho$ while it is changed to each of the other symbols in $[q]$ with probability $\frac{\rho}{q-1}$. Shannon's theorem states that one can have reliable communication with code of rate less than $1 - H_q(\rho)$ but not with rates larger than $1 - H_q(\rho)$. Thus, Shannon's capacity for qSC_ρ is $1 - H_q(\rho)$, which matches the expression for the list-decoding capacity.

Note that in qSC_ρ, the expected fraction of errors when a codeword is transmitted is ρ. Further, as the errors on each symbol occur independently, the Chernoff bound implies that with high probability the fraction of errors is concentrated around ρ. However, Shannon's proof crucially uses the fact that these (roughly) ρ fraction of errors occur randomly. What Theorems 2.1 and 2.3 say is that even with a ρ fraction of *adversarial* errors[1] one can

[1] Where both the *location* and the *nature* of errors are arbitrary.

have reliable communication via codes of rate $1 - H_q(\rho)$ with list decoding using lists of sufficiently large constant size.

We now consider the list-decoding capacity in some more detail. First we note the following special case of the expression for list-decoding capacity for large enough alphabets. When q is $2^{\Theta(1/\varepsilon)}$, $1 - \rho - \varepsilon$ is a good approximation of $H_q(\rho)$ (see Proposition 2.2). Recall that in Section 1.2, we saw that $1 - \rho$ is the information theoretic limit for codes over any alphabet. The discussion above states that we match this bound for large alphabets.

The proof of Theorem 2.1 uses a general random code. A natural question to ask is if one can prove Theorem 2.1 for special classes of codes: for example, linear codes. For $q = 2$ it is known that Theorem 2.1 is true for linear codes [51]. However, unlike general codes, where Theorem 2.1 (with $\delta < 1$) holds for random codes with high probability, the result in [51] does *not* hold with high probability. For $q > 2$, it is only known that random linear codes (with high probability) are (ρ, L)-list decodable with rate at least $1 - H_q(\rho) - \frac{1}{\log_q(L+1)} - o(1)$.

Achieving List-Decoding Capacity with Explicit Codes

There are two unsatisfactory aspects of Theorem 2.1: (i) The codes are not explicit and (ii) There is no efficient list-decoding algorithm. In light of Theorem 2.1, we can formalize the challenge of list decoding that was posed in Section 1.2.3 as follows:

Grand Challenge. *Let $q \geqslant 2$ and let $0 < \rho < 1 - 1/q$ and $\varepsilon > 0$ be reals. Give an explicit construction[2] of a q-ary code C with rate $1 - H_q(\rho) - \varepsilon$ that is $(\rho, O(1/\varepsilon))$-list decodable. Further, design a polynomial time list-decoding algorithm that can correct ρ fraction of errors while using lists of size $O(1/\varepsilon)$.*

We still do not know how to meet the above grand challenge in its entirety. In Chapter 3, we will show how to meet the challenge above for large enough alphabets (with lists of larger size).

2.2.2 Results Related to the q-ary Entropy Function

We conclude our discussion on list decoding by recording few properties of the q-ary entropy function that will be useful later.

We first start with a calculation where the q-ary entropy function naturally pops up. This hopefully will give the reader a feel for the function (and as a bonus will pretty much prove the lower bound in Proposition 2.1). Let $0 \leqslant x \leqslant 1$ and $q \geqslant 2$. We claim that the quantity $\binom{n}{nx}(q-1)^{nx}$ is approximated very well by $q^{H_q(x)n}$ for large enough n. To see this, let us

[2]By explicit construction, we mean an algorithm that in time polynomial in the block length of the code can output some succinct description of the code. For a linear code, such a description could be the generator matrix of the code.

first use Stirling's approximation of $m!$ by $(m/e)^m$ (for large enough m)[3] to approximate $\binom{n}{nx}$:

$$\binom{n}{nx} = \frac{n!}{(nx)!(n-nx)!} \approx \frac{n^n e^{nx} e^{n-nx}}{(nx)^{nx}(n-nx)^{n-nx}e^n} = \frac{q^{n\log_q n}}{q^{nx\log_q(nx)} q^{n(1-x)\log_q(n(1-x))}}$$
$$= q^{-n(x\log_q x + (1-x)\log_q(1-x))}.$$

Thus, we have

$$\binom{n}{nx}(q-1)^{nx} \approx q^{-n(x\log_q x+(1-x)\log_q(1-x))} \cdot q^{nx\log_q(q-1)} = q^{H_q(x)n},$$

as desired.

Figure 2.1 gives a pictorial view of the q-ary function for the first few values of q.

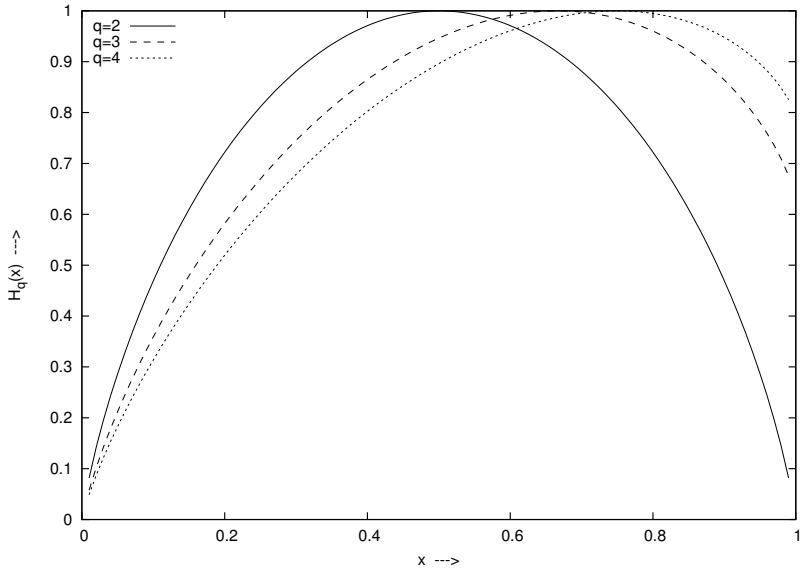


Figure 2.1: A plot of $H_q(x)$ for $q = 2, 3$ and 4. The maximum value of 1 is achieved at $x = 1 - 1/q$.

We now look at the q-ary entropy function for large q.

Proposition 2.2. *For small enough ε, $1 - H_q(\rho) \geqslant 1 - \rho - \varepsilon$ for every $0 < \rho \leqslant 1 - 1/q$ if and only if q is $2^{\Theta(1/\varepsilon)}$.*

[3]There is a $\sqrt{2\pi n}$ factor that we are ignoring.

Proof. We first note that $H_q(\rho) = \rho \log_q(q-1) - \rho \log_q \rho - (1-\rho)\log_q(1-\rho) = \rho \log_q(q-1) + H(\rho)/\log_2 q$. Now if $q = 2^{1/\varepsilon}$, we get that $H_q(\rho) \leqslant \rho + \varepsilon$ as $\log_q(q-1) \leqslant 1$ and $H(\rho) \leqslant 1$. Next we claim that for small enough ε, if $q \geqslant 1/\varepsilon^2$ then $\log_q(q-1) \geqslant 1-\varepsilon$. Indeed, $\log_q(q-1) = 1 + (1/\ln q)\ln(1-1/q) = 1 - O\left(\frac{1}{q \ln q}\right)$, which is at least $1-\varepsilon$ for $q \geqslant 1/\varepsilon^2$. Finally, if $q = 2^{o\left(\frac{1}{\varepsilon}\right)}$ (but $q \geqslant 1/\varepsilon^2$), then for fixed ρ, $H(\rho)/\log q = \varepsilon \cdot \omega(1)$. Then $\rho \log_q(q-1) + H(\rho)/\log q \geqslant \rho - \varepsilon + \varepsilon \cdot \omega(1) > \rho + \varepsilon$, which implies that $1 - H_q(\rho) < 1 - \rho - \varepsilon$, as desired. □

Next, we look at the entropy function when its value is very close to 1.

Proposition 2.3. *For small enough $\varepsilon > 0$,*

$$H_q\left(1 - \frac{1}{q} - \varepsilon\right) \leqslant 1 - c_q \varepsilon^2,$$

c_q is constant that only depends on q.

Proof. The intuition behind the proof is the following. Since the derivate of $H_q(x)$ is zero at $x = 1 - 1/q$, in the Taylor expansion of $H_q(1 - 1/q - \varepsilon)$ the ε term will vanish. We will now make this intuition more concrete. We will think of q as fixed and $1/\varepsilon$ as growing. In particular, we will assume that $\varepsilon < 1/q$. Consider the following equalities:

$$\begin{aligned}
H_q(1 - 1/q - \varepsilon) = & -\left(1 - \frac{1}{q} - \varepsilon\right)\log_q\left(\frac{1 - 1/q - \varepsilon}{q-1}\right) - \left(\frac{1}{q} + \varepsilon\right)\log_q\left(\frac{1}{q} + \varepsilon\right) \\
= & -\log_q\left(\frac{1}{q}\left(1 - \frac{\varepsilon q}{q-1}\right)\right) + \left(\frac{1}{q} + \varepsilon\right)\log_q\left(\frac{1 - (\varepsilon q)/(q-1)}{1 + \varepsilon q}\right) \\
= & \ 1 - \frac{1}{\ln q}\left[\ln\left(1 - \frac{\varepsilon q}{q-1}\right) - \left(\frac{1}{q} + \varepsilon\right)\ln\left(\frac{1 - (\varepsilon q)/(q-1)}{1 + \varepsilon q}\right)\right] \\
= & \ 1 + o(\varepsilon^2) - \frac{1}{\ln q}\left[-\frac{\varepsilon q}{q-1} - \frac{\varepsilon^2 q^2}{2(q-1)^2} - \left(\frac{1}{q} + \varepsilon\right)\left(-\frac{\varepsilon q}{q-1}\right.\right. \\
& \qquad \left.\left. - \frac{\varepsilon^2 q^2}{2(q-1)^2} - \varepsilon q + \frac{\varepsilon^2 q^2}{2}\right)\right] \qquad (2.3) \\
= & \ 1 + o(\varepsilon^2) - \frac{1}{\ln q}\left[-\frac{\varepsilon q}{q-1} - \frac{\varepsilon^2 q^2}{2(q-1)^2}\right. \\
& \qquad \left. - \left(\frac{1}{q} + \varepsilon\right)\left(-\frac{\varepsilon q^2}{q-1} + \frac{\varepsilon^2 q^3 (q-2)}{2(q-1)^2}\right)\right] \\
= & \ 1 + o(\varepsilon^2) - \frac{1}{\ln q}\left[-\frac{\varepsilon^2 q^2}{2(q-1)^2} + \frac{\varepsilon^2 q^2}{q-1} - \frac{\varepsilon^2 q^2 (q-2)}{2(q-1)^2}\right] \qquad (2.4) \\
= & \ 1 - \frac{\varepsilon^2 q^2}{2 \ln q(q-1)} + o(\varepsilon^2)
\end{aligned}$$

$$\leqslant 1 - \frac{\varepsilon^2 q^2}{4 \ln q(q-1)} \tag{2.5}$$

(2.3) follows from the fact that for $|x| < 1$, $\ln(1+x) = x - x^2/2 + x^3/3 - \ldots$ and by collecting the ε^3 and smaller terms in $o(\varepsilon^2)$. (2.4) follows by rearranging the terms and by absorbing the ε^3 term in $o(\varepsilon^2)$. The last step is true assuming ε is small enough. □

We will also work with the inverse of the q-ary entropy function. Note that $H_q(\cdot)$ on the domain $[0, 1-1/q]$ is an bijective map into $[0,1]$. Thus, we define $H_q^{-1}(y) = x$ such that $H_q(x) = y$ and $0 \leqslant x \leqslant 1 - 1/q$. Finally, we will need the following lower bound.

Lemma 2.4. *For every $0 \leqslant y \leqslant 1 - 1/q$ and for every small enough $\varepsilon > 0$,*

$$H_q^{-1}(y - \varepsilon^2/c_q') \geqslant H_q^{-1}(y) - \varepsilon,$$

where $c_q' \geqslant 1$ is a constant that depends only on q.

Proof. It is easy to check that $H_q^{-1}(y)$ is a strictly increasing convex function in the range $y \in [0,1]$. This implies that the derivate of $H_q^{-1}(y)$ increases with y. In particular, $(H_q^{-1})'(1) \geqslant (H_q^{-1})'(y)$ for every $0 \leqslant y \leqslant 1$. In other words, for every $0 < y \leqslant 1$, and (small enough) $\delta > 0$, $\frac{H_q^{-1}(y) - H_q^{-1}(y-\delta)}{\delta} \leqslant \frac{H_q^{-1}(1) - H_q^{-1}(1-\delta)}{\delta}$. Proposition 2.3 along with the facts that $H_q^{-1}(1) = 1 - 1/q$ and H_q^{-1} is increasing completes the proof if one picks $c_q' = \max(1, 1/c_q)$ and $\delta = \varepsilon^2/c_q'$. □

2.3 Definitions Related to Property Testing of Codes

We first start with some generic definitions. Let $q \geqslant 2$, $n \geqslant 1$ be integers and let $0 < \varepsilon < 1$ be a real. Given a vector $\mathbf{x} \in [q]^n$ and a subset $S \subseteq [q]^n$, we say that $\mathbf{x}$ is *ε-close* to S if there exist a $\mathbf{y} \in S$ such that $\delta(\mathbf{x}, \mathbf{y}) \leqslant \varepsilon$, where $\delta(\mathbf{x}, \mathbf{y}) = \Delta(\mathbf{x}, \mathbf{y})/n$ is the *relative Hamming distance* between $\mathbf{x}$ and $\mathbf{y}$. Otherwise, $\mathbf{x}$ is *ε-far* from S.

Given a q-ary code C of block length n, an integer $r \geqslant 1$ and real $0 < \varepsilon < 1$, we say that a randomized algorithm T_C is an (r, ε)-tester for C if the following conditions hold:

- (Completeness) For every codeword $\mathbf{y} \in C$, $\Pr[T_C(\mathbf{y}) = 1] = 1$, that is, T_C always *accepts* a codeword.
- (Soundness) For every $\mathbf{y} \in [q]^n$ that is ε-far from C, $\Pr[T_C(\mathbf{y}) = 1] \leqslant 1/3$, that is, with probability at least $2/3$, T_C *rejects* $\mathbf{y}$.
- (Query Complexity) For every random choice made by T_C, the tester only probes at most r positions in $\mathbf{y}$.

We remark that the above definition only makes sense when C has large distance. Otherwise we could choose $C = [q]^n$ and the trivial tester that accepts all received words is a $(0, \varepsilon)$-tester. For this thesis, we will adopt the convention that whenever we are taking about testers for a code C, C will have some non trivial distance (in most cases C will have linear distance).

The above kind of tester is also called a *one-sided tester* as it never makes a mistake in the completeness case. Also, the choice of $2/3$ in the soundness case is arbitrary in the following sense. The probability of rejection can be made $1 - \delta$ for any $\delta > 0$, as long as we are happy with $O(r)$ many queries, which is fine for this thesis as we will be interested in the asymptotics of the query complexity. The number of *queries* (or r) can depend on n. Note that there is a gap in the definition of the completeness and soundness of a tester. In particular, the tester can have arbitrary output when the received word $\mathbf{y}$ is not a codeword but is still ε-close to C. In particular, the tester can still reject (very) close-by codewords. We will revisit this in Chapter 8.

We say that a (r, ε) tester is a *local tester* if it makes sub-linear number of queries[4], that is, $r = o(n)$ and ε is some small enough constant. A code is called a *Locally Testable Code* (or *LTC*), if it has a local tester. We also say that a local tester for a code C allows for locally testing C.

2.4 Common Families of Codes

In this section, we will review some code families that will be used frequently in this thesis.

2.4.1 Reed-Solomon Codes

Reed-Solomon codes (named after their inventors [90]) is a linear code that is based on univariate polynomials over finite fields. More formally, an $[n, k+1]_q$ Reed-Solomon code with $k < n$ and $q \geqslant n$ is defined as follows. Let $\alpha_1, \ldots, \alpha_n$ be distinct elements from $\mathbb{F}_q$ (which is why we needed $q \geqslant n$). Every message $\mathbf{m} = \langle m_0, \ldots, m_k \rangle \in \mathbb{F}_q^{k+1}$ is thought of as a degree k polynomial over $\mathbb{F}_q$ by assigning the $k+1$ symbols to the $k+1$ coefficients of a degree k polynomial. In other words, $P_{\mathbf{m}}(X) = m_0 + m_1 X + \cdots + m_k X^k$. The codeword corresponding to $\mathbf{m}$ is defined as follows

$$RS(\mathbf{m}) = \langle P_{\mathbf{m}}(\alpha_1), \ldots, P_{\mathbf{m}}(\alpha_n) \rangle.$$

Now a degree k polynomial can have at most k roots in any field. This implies that any two distinct degree k polynomials can agree in at most k places. In other words,

Proposition 2.5. *An $[n, k+1]_q$ Reed-Solomon code is an $[n, k+1, d = n-k]_q$ code.*

[4]Recall that in this thesis we are implicitly dealing with code families.

By the Singleton bound (see for example [80]), the distance of any code of dimension $k+1$ and length n is at most $n-k$. Thus, Reed-Solomon codes have the optimal distance: such codes are called *Maximum Distance Separable* (or *MDS*) codes. The MDS property along with its nice algebraic structure has made Reed-Solomon code the center of a lot of research in coding theory. In particular, the algebraic properties of these codes have been instrumental in the algorithmic progress in list decoding [97, 63, 85]. In addition to their nice theoretical applications, Reed-Solomon codes have found widespread use in practical applications. In particular, these codes are used in CDs, DVDs and other storage media, deep space communications, DSL and paper bar codes. We refer the reader to [105] for more details on some of these applications of Reed-Solomon codes.

2.4.2 Reed-Muller Codes

Reed-Muller codes are generalization of Reed-Solomon codes. For integers $\ell \geqslant 1$ and $m \geqslant 1$, the message space is the set of all polynomials over $\mathbb{F}_q$ in ℓ variables that have total degree at most m. The codeword corresponding to a message is the evaluation of the corresponding ℓ-variate polynomial over n distinct points in $\mathbb{F}_q^\ell$ (note that this requires $q^\ell \geqslant n$). Finally, note that when $\ell = 1$ and $m = k$, we get an $[n, k+1]_q$ Reed-Solomon code. Interestingly, Reed-Muller codes [82, 89] were discovered before Reed-Solomon codes.

2.5 Basic Finite Field Algebra

We will be using a fair amount of finite field algebra in the thesis. In this section, we recap some basic notions and facts about finite fields.

A field consists of a set of elements that is closed under addition, multiplication and (both additive and multiplicative) inversion. It also has two special elements 0 and 1, which are the additive and multiplicative identities respectively. A field is called a *finite field* if its set of elements is finite. The set of integers modulo some prime p, form the finite field $\mathbb{F}_p$.

The ring of univariate polynomials with coefficients from $\mathbb{F}$ will be denoted by $\mathbb{F}[X]$. A polynomial $E(X)$ is said to be irreducible if for every way of writing $E(X) = A(X) \cdot B(X)$, either $A(X)$ or $B(X)$ is a constant polynomial. A polynomial is called *monic*, if the coefficient of its leading term is 1.

If $E(X)$ is an irreducible polynomial of degree d over a field $\mathbb{F}$, then the quotient ring $\mathbb{F}[X]/(E(X))$, consisting of all polynomials in $\mathbb{F}[X]$ modulo $E(X)$ is itself a finite field and is called *field extension* of $\mathbb{F}$. The extension field also forms a vector space of dimension d over $\mathbb{F}$.

All finite fields are either $\mathbb{F}_p$ for prime p or is an extension of a prime field. Thus, the number of elements in a finite field is a prime power. Further, for any prime power q there exists only one finite field (up to isomorphism). For any q that is a power of prime p, the field $\mathbb{F}_q$ has *characteristic* of p. The multiplicative groups of non-zero elements of a field

$\mathbb{F}_q$, denoted by $\mathbb{F}_q^*$, is known to be cyclic. In other words, $\mathbb{F}_q^* = \{1, \gamma, \gamma^2, \ldots, \gamma^{q-2}\}$ for some element $\gamma \in \mathbb{F}_q \setminus \{0\}$. γ is also called the *primitive element* or *generator* of $\mathbb{F}_q^*$.

The following property of finite fields will be crucial. Any polynomial $f(X)$ of degree at most d in $\mathbb{F}[X]$ has at most d roots, where $\alpha \in \mathbb{F}$ is a root of $f(X)$ if $f(\alpha) = 0$. We would be also interested in finding roots of univariate polynomials (over extension fields) for which we will use a classical algorithm due to Berlekamp [16].

Theorem 2.4 ([16]). *Let p be a prime. There exists a deterministic algorithm that on input a polynomial in $\mathbb{F}_{p^t}[X]$ of degree d, can find all the irreducible factors (and hence the roots) in time polynomial in d, p and t.*

Chapter 3

LIST DECODING OF FOLDED REED-SOLOMON CODES

3.1 Introduction

Even though list decoding was defined in the late 1950s, there was essentially no algorithmic progress that could harness the potential of list decoding for nearly forty years. The work of Sudan [97] and improvements to it by Guruswami and Sudan in [63], achieved efficient list decoding up to a $\rho_{\mathrm{GS}}(R) = 1 - \sqrt{R}$ fraction of errors for Reed-Solomon codes of rate R. Note that $1 - \sqrt{R} > \rho_U(R) = (1-R)/2$ for every rate R, $0 < R < 1$, so this result showed that list decoding can be effectively used to go beyond the unique decoding radius for every rate (see Figure 3.1). The ratio $\rho_{\mathrm{GS}}(R)/\rho_U(R)$ approaches 2 for rates $R \to 0$, enabling error-correction when the fraction of errors approaches 100%, a feature that has found numerous applications outside coding theory, see for example [98], [49, Chap. 12].

Unfortunately, the improvement provided by [63] over unique decoding diminishes for larger rates, which is actually the regime of greater practical interest. For rates $R \to 1$, the ratio $\frac{\rho_{\mathrm{GS}}(R)}{\rho_U(R)}$ approaches 1, and already for rate $R = 1/2$ the ratio is at most 1.18. Thus, while the results of [97, 63] demonstrated that list decoding always, for every rate, enables correcting more errors than unique decoding, they fell short of realizing the full quantitative potential of list decoding (recall that the list-decoding capacity promises error correction up to a $1 - R = 2\rho_U(R)$ fraction of errors).

The bound $\rho_{\mathrm{GS}}(R)$ stood as the best known decoding radius for efficient list decoding (for any code) for several years. In fact constructing (ρ, L)-list decodable codes of rate R for $\rho > \rho_{\mathrm{GS}}(R)$ and polynomially bounded L, regardless of the complexity of actually performing list decoding to radius ρ, itself was elusive. Some of this difficulty was due to the fact that $1 - \sqrt{R}$ is the largest radius for which small list size can be shown generically, via the so-called Johnson bound which argues about the number of codewords in Hamming balls using only information on the relative distance of the code, cf. [48].

In a recent breakthrough paper [85], Parvaresh and Vardy presented codes that are list-decodable beyond the $1-\sqrt{R}$ radius for low rates R. The codes they suggest are variants of Reed-Solomon (or simply RS) codes obtained by evaluating $m \geqslant 1$ correlated polynomials at elements of the underlying field (with $m = 1$ giving RS codes). For any $m \geqslant 1$, they achieve the list-decoding radius $\rho_{\mathrm{PV}}^{(m)}(R) = 1 - \sqrt[m+1]{m^m R^m}$. For rates $R \to 0$, choosing m large enough, they can list decode up to radius $1 - O(R \log(1/R))$, which approaches the capacity $1 - R$. However, for $R \geqslant 1/16$, the best choice of m (the one that maximizes $\rho_{\mathrm{PV}}^{(m)}(R)$) is in fact $m = 1$, which reverts back to RS codes and the list-decoding radius $1 - \sqrt{R}$. (See Figure 3.1 where the bound $1 - \sqrt[3]{4R^2}$ for the case $m = 2$ is plotted

— except for very low rates, it gives a small improvement over $\rho_{\mathrm{GS}}(R)$.) Thus, getting arbitrarily close to capacity for some rate, as well as beating the $1 - \sqrt{R}$ bound for every rate, both remained open before our work[1].

In this chapter, we describe codes that get arbitrarily close to the list-decoding capacity for every rate (for large alphabets). In other words, we give explicit codes of rate R together with polynomial time list decoding up to a fraction $1 - R - \varepsilon$ of errors for every rate R and arbitrary $\varepsilon > 0$. As mentioned in Section 2.2.1, this attains the best possible trade-off one can hope for between the rate and list-decoding radius. This is the first result that approaches the list-decoding capacity for *any* rate (and over any alphabet).

Our codes are simple to describe: they are *folded Reed-Solomon codes*, which are in fact *exactly* Reed-Solomon codes, but viewed as codes over a larger alphabet by careful bundling of codeword symbols. Given the ubiquity of RS codes, this is an appealing feature of our result, and in fact our methods directly yield better decoding algorithms for RS codes when errors occur in *phased bursts* (a model considered in [75]).

Our result extends easily to the problem of *list recovery* (recall Definition 2.4). The biggest advantage here is that we are able to achieve a rate that is independent of the size of the input lists. This is an extremely useful feature that will be used in Chapters 4 and 5 to design codes over smaller alphabets. In particular, we will construct new codes from folded Reed-Solomon codes that achieve list-decoding capacity over constant sized alphabets (the folded Reed-Solomon codes are defined over alphabets whose size increases with the block length of the code).

Our work builds on existing work of Guruswami and Sudan [63] and Parvaresh and Vardy [85]. See Figure 3.1 for a comparison of our work with previous known list-decoding algorithms (for various codes).

We start with the description of our code in Section 3.2 and give some intuition why these codes might have good list decodable properties. We present the main ideas in our list-decoding algorithms for the folded Reed-Solomon codes in Section 3.3. In Section 3.4, we present and analyze a polynomial time list-decoding algorithm for folded RS codes of rate R that can correct roughly $1 - \sqrt[3]{R^2}$ fraction of errors . In Section 3.5, we extend the results in Section 3.4 to present codes that can be efficiently list decoded up to the list-decoding capacity. Finally, we extend our results to list recovery in Section 3.6.

3.2 Folded Reed-Solomon Codes

In this section, we will define a simple variant of Reed-Solomon codes called folded Reed-Solomon codes. By choosing parameters suitably, we will design a list-decoding algorithm that can decode close to the optimal fraction $1 - R$ of errors with rate R.

[1] Independent of our work, Alex Vardy (personal communication) constructed a variant of the code defined in [85] which could be list decoded with fraction of errors more than $1 - \sqrt{R}$ for all rates R. However, his construction gives only a small improvement over the $1 - \sqrt{R}$ bound and does not achieve the list-decoding capacity.

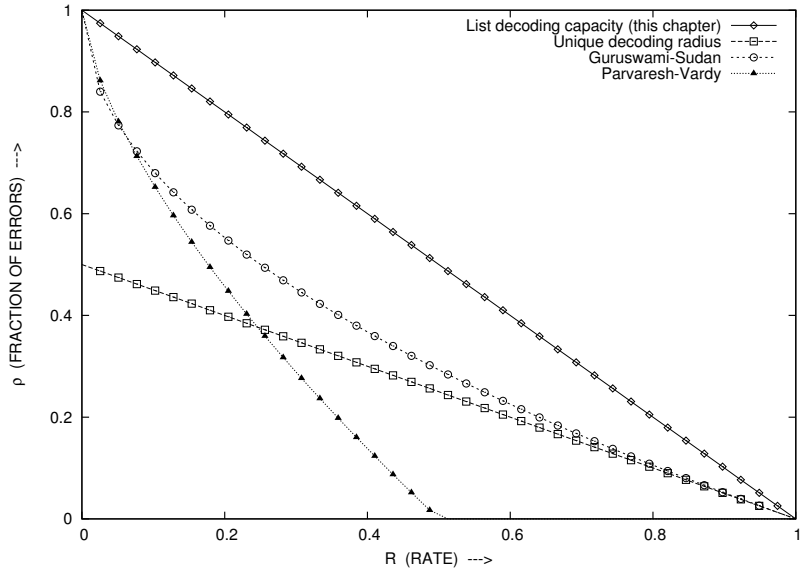


Figure 3.1: List-decoding radius ρ plotted against the rate R of the code for known algorithms. The best possible trade-off, i.e., list-decoding capacity, is $\rho = 1 - R$, and our work achieves this.

3.2.1 Description of Folded Reed-Solomon Codes

Consider a $[n, k+1]_q$ Reed-Solomon code C consisting of evaluations of degree k polynomials over $\mathbb{F}_q$ at the set $\mathbb{F}_q^*$. Note that $q = n + 1$. Let γ be a generator of the multiplicative group $\mathbb{F}_q^*$, and let the evaluation points be ordered as $1, \gamma, \gamma^2, \ldots, \gamma^{n-1}$. Using all nonzero field elements as evaluation points is one of the most commonly used instantiations of Reed-Solomon codes.

Let $m \geqslant 1$ be an integer parameter called the *folding parameter*. For ease of presentation, we will assume that m divides $n = q - 1$.

Definition 3.1 (Folded Reed-Solomon Code). *The m-folded version of the RS code C, denoted $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$, is a code of block length $N = n/m$ over $\mathbb{F}_q^m$, where $n = q - 1$. The encoding of a message $f(X)$, a polynomial over $\mathbb{F}_q$ of degree at most k, has as its j'th symbol, for $0 \leqslant j < n/m$, the m-tuple $(f(\gamma^{jm}), f(\gamma^{jm+1}), \cdots, f(\gamma^{jm+m-1}))$. In other words, the codewords of $C' = \mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ are in one-one correspondence with those of the RS code C and are obtained by bundling together consecutive m-tuple of symbols in codewords of C.*

The way the above definition is stated the message alphabet is $\mathbb{F}_q$ while the codeword alphabet is $\mathbb{F}_q^m$ whereas in our definition of codes, both the alphabets were the same. This

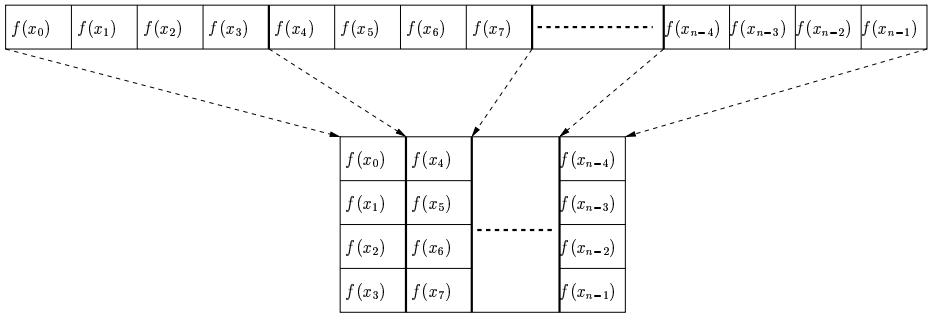


Figure 3.2: Folding of the Reed-Solomon Code with Parameter $m = 4$.

can be easily taken care of by bundling m consecutive message symbols from $\mathbb{F}_q$ to make the message alphabet to be $\mathbb{F}_q^m$. We will however, state our results with the message symbols as coming from $\mathbb{F}_q$ as this simplifies our presentation.

We illustrate the above construction for the choice $m = 4$ in Figure 3.2. The polynomial $f(X)$ is the message, whose Reed-Solomon encoding consists of the values of f at $x_0, x_1, \ldots, x_{n-1}$ where $x_i = \gamma^i$. Then, we perform a folding operation by bundling together tuples of 4 symbols to give a codeword of length $n/4$ over the alphabet $\mathbb{F}_q^4$.

Note that the folding operation does not change the rate R of the original Reed-Solomon code. The relative distance of the folded RS code also meets the Singleton bound and is at least $1 - R$.

Remark 3.1 (Origins of term "folded RS codes"). *The terminology of folded RS codes was coined in [75], where an algorithm to correct random errors in such codes was presented (for a noise model similar to the one used in [27, 18]: see Section 3.7 for more details). The motivation was to decode RS codes from many random "phased burst" errors. Our decoding algorithm for folded RS codes can also be likewise viewed as an algorithm to correct beyond the $1 - \sqrt{R}$ bound for RS codes if errors occur in large, phased bursts (the actual errors can be adversarial).*

3.2.2 Why Might Folding Help?

Since folding seems like such a simplistic operation, and the resulting code is essentially just a RS code but viewed as a code over a large alphabet, let us now understand why it can possibly give hope to correct more errors compared to the bound for RS codes.

Consider the folded RS code with folding parameter $m = 4$. First of all, decoding the folded RS code up to a fraction ρ of errors is certainly not harder than decoding the RS code up to the same fraction ρ of errors. Indeed, we can "unfold" the received word of the folded RS code and treat it as a received word of the original RS code and run the RS list-decoding algorithm on it. The resulting list will certainly include all folded RS codewords

within distance ρ of the received word, and it may include some extra codewords which we can, of course, easily prune.

In fact, decoding the folded RS code is a strictly easier task. To see why, say we want to correct a fraction $1/4$ of errors. Then, if we use the RS code, our decoding algorithm ought to be able to correct an error pattern that corrupts every 4'th symbol in the RS encoding of $f(X)$ (i.e., corrupts $f(x_{4i})$ for $0 \leqslant i < n/4$). However, after the folding operation, this error pattern corrupts every one of the symbols over the larger alphabet $\mathbb{F}_q^4$, and thus need not be corrected. In other words, for the same fraction of errors, the folding operation reduces the total number of error patterns that need to be corrected, since the channel has less flexibility in how it may distribute the errors.

It is of course far from clear how one may exploit this to actually correct more errors. To this end, algebraic ideas that exploit the specific nature of the folding and the relationship between a polynomial $f(X)$ and its shifted counterpart $f(\gamma X)$ will be used. These will become clear once we describe our algorithms later in the chapter.

We note that the above simplification of the channel is not attained for free since the alphabet size increases after the folding operation. For folding parameter m that is an absolute constant, the increase in alphabet size is moderate and the alphabet remains polynomially large in the block length. (Recall that the RS code has an alphabet size that is linear in the block length.) Still, having an alphabet size that is a large polynomial is somewhat unsatisfactory. Fortunately, existing alphabet reduction techniques, which are used in Chapter 4, can handle polynomially large alphabets, so this does not pose a big problem.

3.2.3 Relation to Parvaresh Vardy Codes

In this subsection, we relate folded RS codes to the Parvaresh-Vardy (PV) codes [85], which among other things will help make the ideas presented in the previous subsection more concrete.

The basic idea in the PV codes is to encode a polynomial f of degree k by the evaluations of $s \geqslant 2$ polynomials $f_0 = f, f_1, \ldots, f_{s-1}$ where $f_i(X) = f_{i-1}(X)^d \mod E(X)$ for an appropriate power d (and some irreducible polynomial $E(X)$ of some appropriate degree) — let us call s the *order* of such a code. Our first main idea is to pick the irreducible polynomial $E(X)$ (and the parameter d) in such a manner that every polynomial f of degree at most k satisfies the following identity: $f(\gamma X) = f(X)^d \mod E(X)$, where γ is the generator of the underlying field. Thus, a folded RS code with bundling using an γ as above is in fact exactly the PV code of order $s = m$ for the set of evaluation points $\{1, \gamma^m, \gamma^{2m}, \ldots, \gamma^{(n/m-1)m}\}$. This is nice as it shows that PV codes can meet the Singleton bound (since folded RS codes do), but as such does not lead to any better codes for list decoding.

We now introduce our second main idea. Let us compare the folded RS code to a PV code of order 2 (instead of order m where m divides n) for the set of evaluation points $\{1, \gamma, \ldots \gamma^{m-2}, \gamma^m, \gamma^{m+1}, \ldots, \gamma^{2m-2}, \ldots, \gamma^{n-m}, \gamma^{n-m+1} \ldots, \gamma^{n-2}\}$. We find that in the PV encoding of f, for every $0 \leqslant i \leqslant n/m - 1$ and every $0 < j < m - 1$, $f(\gamma^{mi+j})$

appears exactly twice (once as $f(\gamma^{mi+j})$ and another time as $f_1(\gamma^{-1}\gamma^{mi+j})$), whereas it appears only once in the folded RS encoding. (See Figure 3.3 for an example when $m = 4$ and $s = 2$.) In other words, the PV and folded RS codes have the same information, but the

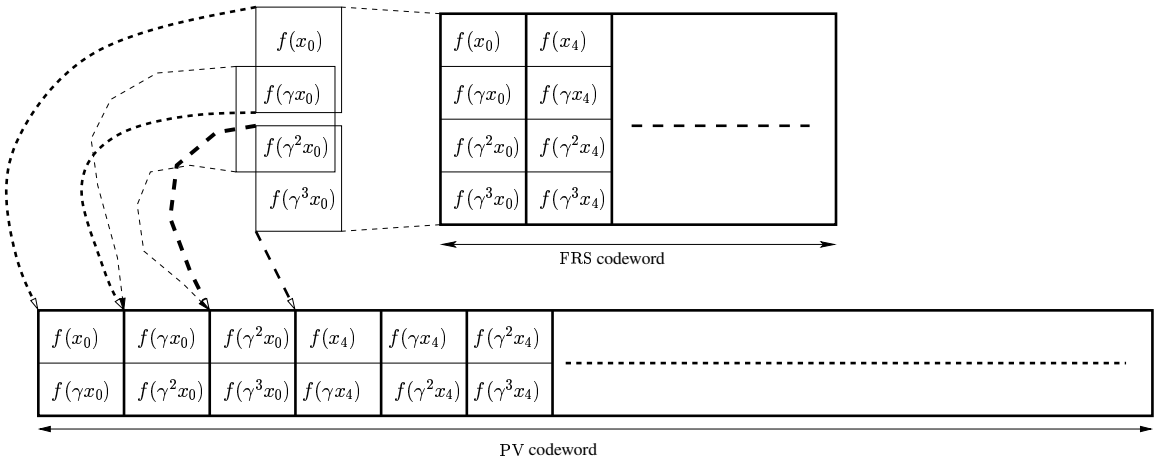


Figure 3.3: The correspondence between a folded Reed-Solomon code (with $m = 4$ and $x_i = \gamma^i$) and the Parvaresh Vardy code (of order $s = 2$) evaluated over $\{1, \gamma, \gamma^2, \gamma^4, \ldots, \gamma^{n-4}, \ldots, \gamma^{n-2}\}$. The correspondence for the first block in the folded RS codeword and the first three blocks in the PV codeword is shown explicitly in the left corner of the figure.

rate of the folded RS codes is bigger by a factor of $\frac{2m-2}{m} = 2 - \frac{2}{m}$. Decoding the folded RS codes from a fraction ρ of errors reduces to correcting the same fraction ρ of errors for the PV code. But the rate vs. list-decoding radius trade-off is better for the folded RS code since it has (for large enough m, almost) twice the rate of the PV code.

In other words, our folded RS codes are chosen such that they are compressed forms of suitable PV codes, and thus have better rate than the corresponding PV code for a similar error-correction performance. This is where our gain is, and using this idea we are able to construct folded RS codes of rate R that are list decodable up to radius roughly $1 - \sqrt[s+1]{R^s}$ for any $s \geqslant 1$. Picking s large enough lets us get within any desired ε of list-decoding capacity.

3.3 Problem Statement and Informal Description of the Algorithms

We first start by stating more precisely the problem we will solve in the rest of the chapter. We will give list-decoding algorithms for the folded Reed-Solomon code $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ of rate R. More precisely, for every $1 \leqslant s \leqslant m$ and $\delta > 0$, given a received word $\mathbf{y} =$

$\langle (y_0, \ldots, y_{m-1}), \ldots, (y_{n-m}, \ldots, y_{n-1}) \rangle$ (where recall $n = q - 1$), we want to output all codewords in $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ that disagree with $\mathbf{y}$ in at most $1 - (1+\delta)\left(\frac{m}{m-s+1}\right) R^{s/(s+1)}$ fraction of positions in polynomial time. In other words, we need to output all degree k polynomials $f(X)$ such that for at least $(1+\delta)\left(\frac{m}{m-s+1}\right) R^{s/(s+1)}$ fraction of $0 \leqslant i \leqslant n/m - 1$, $f(\gamma^{im+j}) = y_{im+j}$ (for every $0 \leqslant j \leqslant m-1$). By picking the parameters m, s and δ carefully, we will get folded Reed-Solomon codes of rate R that can be list decoded up to a $1 - R - \varepsilon$ fraction of errors (for any $\varepsilon > 0$).

We will now present the main ideas need to design our list-decoding algorithm. Readers familiar with list-decoding algorithms of [97, 63, 85] can skip the rest of this section.

For the ease of presentation we will start with the case when $s = m$. As a warm up, let us consider the case when $s = m = 1$. Note that for $m = 1$, we are interested in list decoding Reed-Solomon codes. More precisely, given the received word $\mathbf{y} = \langle y_0, \ldots, y_{n-1} \rangle$, we are interested in all degree k polynomials $f(X)$ such that for at least $(1+\delta)\sqrt{R}$ fraction of positions $0 \leqslant i \leqslant n-1$, $f(\gamma^i) = y_i$. We now sketch the main ideas of the algorithms in [97, 63]. The algorithms have two main steps: the first is an *interpolation* step and the second one is a *root finding* step. In the interpolation step, the list-decoding algorithm finds a bivariate polynomial $Q(X, Y)$ that *fits* the input. That is,

> for every position i, $Q(\gamma^i, y_i) = 0$.

Such a polynomial $Q(\cdot, \cdot)$ can be found in polynomial time if we search for one with large enough total degree (this amounts to solving a system of linear equations). After the interpolation step, the root finding step finds all factors of $Q(X, Y)$ of the form $Y - f(X)$. The crux of the analysis is to show that

> for every degree k polynomial $f(X)$ that satisfies $f(\gamma^i) = y_i$ for at least $(1 + \delta)\sqrt{R}$ fraction of positions i, $Y - f(X)$ is indeed a factor of $Q(X, Y)$.

However, the above is not true for every bivariate polynomial $Q(X, Y)$ that satisfies $Q(\gamma^i, y_i) = 0$ for all positions i. The main ideas in [97, 63] were to introduce more constraints on $Q(X, Y)$. In particular, the work of Sudan [97] added the constraint that a certain weighted degree of $Q(X, Y)$ is below a fixed upper bound. Specifically, $Q(X, Y)$ was restricted to have a non-trivially bounded $(1, k)$-weighted degree. The $(1, k)$-weighted degree of a monomial $X^i Y^j$ is $i + jk$ and the $(1, k)$-weighted degree of a bivariate polynomial $Q(X, Y)$ is the maximum $(1, k)$-weighted degree among its monomials. The intuition behind defining such a weighted degree is that given $Q(X, Y)$ with weighted $(1, k)$ of D, the *univariate* polynomial $Q(X, f(X))$, where $f(X)$ is some degree k polynomial, has total degree at most D. The upper bound D is chosen carefully such that if $f(X)$ is a codeword that needs to be output, then $Q(X, f(X))$ has more than D zeroes and thus $Q(X, f(X)) \equiv 0$, which in turn implies that $Y - f(X)$ divides $Q(X, Y)$. To get to the bound of $1 - (1+\delta)\sqrt{R}$, Guruswami and Sudan in [63], added a further constraint on $Q(X, Y)$ that required it to have r roots at (γ^i, y_i), where r is some parameter (in [97] $r = 1$ while in [63], r is roughly $1/\delta$).

We now consider the next non-trivial case of $m = s = 2$ (the ideas for this case can be easily generalized for the general $m = s$ case). Note that now given the received word $\langle (y_0, y_1), (y_2, y_3), \ldots, (y_{n-2}, y_{n-1}) \rangle$ we want to find all degree k polynomials $f(X)$ such that for at least $2(1+\delta)\sqrt[3]{R^2}$ fraction of positions $0 \leqslant i \leqslant n/2 - 1$, $f(\gamma^{2i}) = y_{2i}$ and $f(\gamma^{2i+1}) = y_{2i+1}$. As in the previous case, we will have an interpolation and a root finding step. The interpolation step is a straightforward generalization of $m = 1$ case: we find a trivariate polynomial $Q(X, Y, Z)$ that fits the received word, that is, for every $0 \leqslant i \leqslant n/2 - 1$, $Q(\gamma^{2i}, y_{2i}, y_{2i+1}) = 0$. Further, $Q(X, Y, Z)$ has an upper bound on its $(1, k, k)$-weighted degree (which is a straightforward generalization of the $(1, k)$-weighted degree for the bivariate case) and has a multiplicity of r at every point. These straightforward generalization and their various properties are recorded in Section 3.4.1. For the root finding step, it suffices to show that for every degree k polynomial $f(X)$ that needs to be output $Q(X, f(X), f(\gamma X)) \equiv 0$. This, however does not follow from weighted degree and multiple root properties of $Q(X, Y, Z)$. Here we will need two new ideas, the first of which is to show that for some irreducible polynomial $E(X)$ of degree $q - 1$, $f(X)^q \equiv f(\gamma X) \mod (E(X))$ (this is Lemma 3.4). The second idea, due to Parvaresh and Vardy [85], is the following. We first obtain the bivariate polynomial (over an appropriate extension field) $T(Y, Z) \equiv Q(X, Y, Z) \mod (E(X))$. Note that by our first idea, we are looking for solutions on the curve $Z = Y^q$ (Y corresponds to $f(X)$ and Z corresponds to $f(\gamma X)$ in the extension field). The crux of the argument is to show that all the polynomials $f(X)$ that need to be output correspond to (in the extension field) some root of the equation $T(Y, Y^q) = 0$. See Section 3.4.3 for the details.

As was mentioned earlier, the extension of the $m = s = 2$ case to the general $m = s > 2$ case is fairly straightforward (and is presented in part as Lemma 3.6). To go from $s = m$ to any $s \leqslant m$ requires another simple idea: We will reduce the problem of list decoding folded Reed-Solomon code with folding parameter m to the problem of list decoding folded Reed-Solomon code with folding parameter s. We then use the algorithm outlined in the previous paragraph for the folded Reed-Solomon code with folding parameter s. A careful tracking of the agreement parameter in the reduction, brings down the final agreement fraction (that is required for the original folded Reed-Solomon code with folding parameter m) from $m(1+\delta) \sqrt[m+1]{R^m}$ (which can be obtained without the reduction) to $(1+\delta)\left(\frac{m}{m-s+1}\right) \sqrt[s+1]{R^s}$. This reduction is presented in detail in Section 3.4 for the $s = 2$ case. The generalization to any $s \leqslant m$ is presented in Section 3.5.

3.4 Trivariate Interpolation Based Decoding

As mentioned in the previous section, the list-decoding algorithm for RS codes from [97, 63] is based on bivariate interpolation. The key factor driving the agreement parameter t needed for the decoding to be successful was the ($(1, k)$-weighted) degree D of the interpolated bivariate polynomial. Our quest for an improved algorithm for folded RS codes will be based on trying to lower this degree D by using more degrees of freedom in the interpo-

lation. Specifically, we will try to use *trivariate interpolation* of a polynomial $Q(X, Y_1, Y_2)$ through n points in $\mathbb{F}_q^3$. This enables us to perform the interpolation with D in $O((k^2 n)^{1/3})$, which is much smaller than the $\Theta(\sqrt{kn})$ bound for bivariate interpolation. In principle, this could lead to an algorithm that works for agreement fraction $R^{2/3}$ instead of $R^{1/2}$. Of course, this is a somewhat simplistic hope and additional ideas are needed to make this approach work. We now turn to the task of developing a trivariate interpolation based decoder and proving that it can indeed decode up to a $1 - R^{2/3}$ fraction of errors.

3.4.1 Facts about Trivariate Interpolation

We begin with some basic definitions and facts concerning trivariate polynomials.

Definition 3.2. *For a polynomial $Q(X, Y_1, Y_2) \in \mathbb{F}_q[X, Y_1, Y_2]$, its $(1, k, k)$-weighted degree is defined to be the maximum value of $\ell + kj_1 + kj_2$ taken over all monomials $X^\ell Y_1^{j_1} Y_2^{j_2}$ that occur with a nonzero coefficient in $Q(X, Y_1, Y_2)$. If $Q(X, Y_1, Y_2) \equiv 0$ then its $(1, k, k)$-weighted degree is 0.*

Definition 3.3 (Multiplicity of zeroes). *A polynomial $Q(X, Y_1, Y_2)$ over $\mathbb{F}_q$ is said to have a zero of multiplicity $r \geqslant 1$ at a point $(\alpha, \beta_1, \beta_2) \in \mathbb{F}_q^3$ if $Q(X + \alpha, Y_1 + \beta_1, Y_2 + \beta_2)$ has no monomial of degree less than r with a nonzero coefficient. (The degree of the monomial $X^i Y_1^{j_1} Y_2^{j_2}$ equals $i + j_1 + j_2$.)*

Lemma 3.1. *Let $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^n$ be an arbitrary set of n triples from $\mathbb{F}_q^3$. Let $Q(X, Y_1, Y_2) \in \mathbb{F}_q[X, Y_1, Y_2]$ be a nonzero polynomial of $(1, k, k)$-weighted degree at most D that has a zero of multiplicity r at $(\alpha_i, y_{i1}, y_{i2})$ for every $i \in [n]$. Let $f(X), g(X)$ be polynomials of degree at most k such that for at least $t > D/r$ values of $i \in [n]$, we have $f(\alpha_i) = y_{i1}$ and $g(\alpha_i) = y_{i2}$. Then, $Q(X, f(X), g(X)) \equiv 0$.*

Proof. If we define $R(X) = Q(X, f(X), g(X))$, then $R(X)$ is a univariate polynomial of degree at most D, and for every $i \in [n]$ for which $f(\alpha_i) = y_{i1}$ *and* $g(\alpha_i) = y_{i2}$, $(X - \alpha_i)^r$ divides $R(X)$. Therefore if $rt > D$, then $R(X)$ has more roots (counting multiplicities) than its degree, and so it must be the zero polynomial. □

Lemma 3.2. *Given an arbitrary set of n triples $\{(\alpha_i, y_{i1}, y_{i2})\}_{i=1}^n$ from $\mathbb{F}_q^3$ and an integer parameter $r \geqslant 1$, there exists a nonzero polynomial $Q(X, Y_1, Y_2)$ over $\mathbb{F}_q$ of $(1, k, k)$-weighted degree at most D such that $Q(X, Y_1, Y_2)$ has a zero of multiplicity r at $(\alpha_i, y_{i1}, y_{i2})$ for all $i \in [n]$, provided $\frac{D^3}{6k^2} > n\binom{r+2}{3}$. Moreover, we can find such a $Q(X, Y_1, Y_2)$ in time polynomial in n, r by solving a system of homogeneous linear equations over $\mathbb{F}_q$.*

Proof. We begin with the following claims. (i) The condition that $Q(X, Y_1, Y_2)$ has a zero of multiplicity r at $(\alpha_i, y_{i1}, y_{i2})$ for all $i \in [n]$ amounts to $n\binom{r+2}{3}$ homogeneous linear conditions in the coefficients of Q; and (ii) The number of monomials in $Q(X, Y_1, Y_2)$ equals the number, say $N_3(k, D)$, of triples (i, j_1, j_2) of nonnegative integers that obey

$i + kj_1 + kj_2 \leqslant D$ is at least $\frac{D^3}{6k^2}$. Hence, if $\frac{D^3}{6k^2} > n\binom{r+2}{3}$, then the number of unknowns exceeds the number of equations, and we are guaranteed a nonzero solution.

To complete the proof, we prove the two claims. To prove the first claim, it suffices to show that for any arbitrary tuple (α, β, γ), the condition that $Q(X, Y, Z)$ has multiplicity r at point (α, β, γ) amounts to $\binom{r+2}{3}$ many homogeneous linear constraints. By the definition of multiplicities of roots, this amounts to setting the coefficients of all monomials of total degree r in $Q(X+\alpha, Y+\beta, Z+\gamma)$ to be zero. In particular, the coefficient of the monomial $X^{i_1}Y^{i_2}Z^{i_3}$ is given by $\sum_{i'_1 \geqslant i_1} \sum_{i'_2 \geqslant i_2} \sum_{i'_3 \geqslant i_3} \binom{i'_1}{i_1}\binom{i'_2}{i_2}\binom{i'_3}{i_3} q_{i'_1,i'_2,i'_3} \alpha^{i'_1 - i_1} \beta^{i'_2 - i_2} \gamma^{i'_3 - i_3}$, where $q_{i'_1,i'_2,i'_3}$ is the coefficient of $X^{i'_1}Y^{i'_2}Z^{i'_3}$ in $Q(X, Y, Z)$. Thus, the condition on multiplicities on roots of $Q(X, Y, Z)$ at (α, β, γ) follows if the following is satisfied by for every triple (i_1, i_2, i_3) such that $i_1 + i_2 + i_3 \leqslant r$:

$$\sum_{i'_1 \geqslant i_1} \sum_{i'_2 \geqslant i_2} \sum_{i'_3 \geqslant i_3} \binom{i'_1}{i_1}\binom{i'_2}{i_2}\binom{i'_3}{i_3} q_{i'_1,i'_2,i'_3} \alpha^{i'_1 - i_1} \beta^{i'_2 - i_2} \gamma^{i'_3 - i_3} = 0.$$

The claim follows by noting that the number of integral solutions to $i_1 + i_2 + i_3 \leqslant r$ is $\binom{r+2}{3}$.

To prove the second claim, following [85], we will first show that the number $N_3(k, D)$ is at least as large as the volume of the 3-dimensional region $\mathcal{P} = \{x + ky_1 + ky_2 \leqslant D \mid x, y_1, y_2 \geqslant 0\} \subset \mathbb{R}^3$. Consider the correspondence between monomials in $\mathbb{F}_q[X, Y, Z]$ and unit cubes in $\mathbb{R}^3$: $X^{i_1}Y^{i_2}Z^{i_3} \to \mathcal{C}(i_1, i_2, i_3)$, where $\mathcal{C}(i_1, i_2, i_3) = [i_1, i_1+1) \times [i_2, i_2+1) \times [i_3, i_3+1)$. Note that the volume of each such cube is 1. Thus, $N_3(k, D)$ is the volume of the union of cubes $\mathcal{C}(i_1, i_2, i_3)$ for positive integers i_1, i_2, i_3 such that $i_1 + ki_2 + ki_3 \leqslant D$: let $\mathcal{U}$ denote this union. It is easy to see that $\mathcal{P} \subset \mathcal{U}$. To complete the claim we will show that the volume of $\mathcal{P}$ equals $\frac{D^3}{6k^2}$. Indeed the volume of $\mathcal{P}$ is

$$\begin{aligned}
\int_0^D \int_0^{(D-x)/k} \int_0^{(D-x)/k-y_1} dy_2\, dy_1\, dx &= \int_0^D \int_0^{(D-x)/k} \left(\frac{D-x}{k} - y_1\right) dy_1\, dx \\
&= \int_0^D \frac{(D-x)^2}{2k^2}\, dx \\
&= \frac{1}{2k^2} \int_0^D z^2\, dz \\
&= \frac{D^3}{6k^2},
\end{aligned}$$

where the third equality follows by substituting $z = D - x$. □

3.4.2 Using Trivariate Interpolation for Folded RS Codes

Let us now see how trivariate interpolation can be used in the context of decoding the folded RS code $C' = \mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ of block length $N = (q-1)/m$. (Throughout this section, we

denote $n = q - 1$.) Given a received word $\mathbf{z} \in (\mathbb{F}_q^m)^N$ for C' that needs to be list decoded, we define $\mathbf{y} \in \mathbb{F}_q^n$ to be the corresponding "unfolded" received word. (Formally, let the j'th symbol of $\mathbf{z}$ be $(z_{j,0}, \ldots, z_{j,m-1})$ for $0 \leqslant j < N$. Then $\mathbf{y}$ is defined by $y_{jm+l} = z_{j,l}$ for $0 \leqslant j < N$ and $0 \leqslant l < m$.)

Suppose that $f(X)$ is a polynomial whose encoding agrees with $\mathbf{z}$ on at least t locations (note that the agreement is on symbols from $\mathbb{F}_q^m$). Then, here is an obvious but important observation:

> For at least $t(m-1)$ values of i, $0 \leqslant i < n$, *both* the equalities $f(\gamma^i) = y_i$ and $f(\gamma^{i+1}) = y_{i+1}$ hold.

Define the notation $g(X) = f(\gamma X)$. Therefore, if we consider the n triples $(\gamma^i, y_i, y_{i+1}) \in \mathbb{F}_q^3$ for $i = 0, 1, \ldots, n-1$ (with the convention $y_n = y_0$), then for at least $t(m-1)$ triples, we have $f(\gamma^i) = y_i$ *and* $g(\gamma^i) = y_{i+1}$. This suggests that interpolating a polynomial $Q(X, Y_1, Y_2)$ through these n triples and employing Lemma 3.1, we can hope that $f(X)$ will satisfy $Q(X, f(X), f(\gamma X)) = 0$, and then somehow use this to find $f(X)$. We formalize this in the following lemma. The proof follows immediately from the preceding discussion and Lemma 3.1.

Lemma 3.3. *Let $\mathbf{z} \in (\mathbb{F}_q^m)^N$ and let $\mathbf{y} \in \mathbb{F}_q^n$ be the unfolded version of $\mathbf{z}$. Let $Q(X, Y_1, Y_2)$ be any nonzero polynomial over $\mathbb{F}_q$ of $(1, k, k)$-weighted degree at D that has a zero of multiplicity r at (γ^i, y_i, y_{i+1}) for $i = 0, 1, \ldots, n-1$. Let t be an integer such that $t > \frac{D}{(m-1)r}$. Then every polynomial $f(X) \in \mathbb{F}_q[X]$ of degree at most k whose encoding according to $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ agrees with $\mathbf{z}$ on at least t locations satisfies $Q(X, f(X), f(\gamma X)) \equiv 0$.*

Lemmas 3.2 and 3.3 motivate the following approach to list decoding the folded RS code $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$. Here $\mathbf{z} \in (\mathbb{F}_q^m)^N$ is the received word and $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in \mathbb{F}_q^n$ is its unfolded version. The algorithm uses an integer multiplicity parameter $r \geqslant 1$, and is intended to work for an agreement parameter $1 \leqslant t \leqslant N$.

Algorithm Trivariate-FRS-decoder:

Step 1 (Trivariate Interpolation) Define the degree parameter

$$D = \lfloor \sqrt[3]{k^2 n r(r+1)(r+2)} \rfloor + 1 \; . \tag{3.1}$$

> Interpolate a nonzero polynomial $Q(X, Y_1, Y_2)$ with coefficients from $\mathbb{F}_q$ with the following two properties: (i) Q has $(1, k, k)$-weighted degree at most D, and (ii) Q has a zero of multiplicity r at (γ^i, y_i, y_{i+1}) for $i = 0, 1, \ldots, n-1$ (where $y_n = y_0$). (Lemma 3.2 guarantees the feasibility of this step as well as its computability in time polynomial in n, r.)

Step 2 (Trivariate "Root-finding") Find a list of all degree $\leqslant k$ polynomials $f(X) \in \mathbb{F}_q[X]$ such that $Q(X, f(X), f(\gamma X)) = 0$. Output those whose encoding agrees with $\mathbf{z}$ on at least t locations.

Ignoring the time complexity of Step 2 for now, we can already claim the following result concerning the error-correction performance of this algorithm.

Theorem 3.1. *The algorithm* Trivariate-FRS-decoder *successfully list decodes the folded Reed-Solomon code* $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ *up to a radius of* $N - \left\lfloor N\frac{m}{m-1}\sqrt[3]{\frac{k^2}{n^2}\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)}\right\rfloor - 2$.

Proof. By Lemma 3.3, we know that any $f(X)$ whose encoding agrees with $\mathbf{z}$ on t or more locations will be output in Step 2, provided $t > \frac{D}{(m-1)r}$. For the choice of D in (3.1), this condition is met for the choice $t = 1 + \left\lfloor \sqrt[3]{\frac{k^2 n}{(m-1)^3}\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r}\right\rfloor$. Indeed, we have

$$\begin{aligned}
\frac{D}{(m-1)r} &\leqslant \frac{1}{(m-1)r}\left(\sqrt[3]{k^2 nr(r+1)(r+2)}+1\right)\\
&= \frac{1}{m-1}\sqrt[3]{k^2 n\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r}\\
&< 1 + \left\lfloor \frac{1}{m-1}\sqrt[3]{k^2 n\left(1+\frac{1}{r}\right)\left(1+\frac{2}{r}\right)} + \frac{1}{(m-1)r}\right\rfloor\\
&= t,
\end{aligned}$$

where the first inequality follows from (3.1) and the fact that for any real $x \geqslant 0$, $\lfloor x \rfloor \leqslant x$ while the second inequality follows from the fact that for any real $x \geqslant 0$, $x < \lfloor x \rfloor + 1$. The decoding radius is equal to $N - t$, and recalling that $n = mN$, we get bound claimed in the lemma. □

The rate of the folded Reed-Solomon code is $R = (k+1)/n > k/n$, and so the fraction of errors corrected (for large enough r) is $1 - \frac{m}{m-1}R^{2/3}$. Letting the parameter m grow, we can approach a decoding radius of $1 - R^{2/3}$.

3.4.3 Root-finding Step

In light of the above discussion, the only missing piece in our decoding algorithm is an efficient way to solve the following trivariate "root-finding" problem:

> Given a nonzero polynomial $Q(X, Y_1, Y_2)$ with coefficients from a finite field $\mathbb{F}_q$, a primitive element γ of the field $\mathbb{F}_q$, and an integer parameter $k < q - 1$, find the list of all polynomials $f(X)$ of degree at most k such that $Q(X, f(X), f(\gamma X)) \equiv 0$.

The following simple algebraic lemma is at the heart of our solution to this problem.

Lemma 3.4. *Let* γ *be a primitive element that generates* $\mathbb{F}_q^*$. *Then we have the following two facts:*

1. *The polynomial* $E(X) \stackrel{\text{def}}{=} X^{q-1} - \gamma$ *is irreducible over* $\mathbb{F}_q$.

2. *Every polynomial* $f(X) \in \mathbb{F}_q[X]$ *of degree less than* $q-1$ *satisfies* $f(\gamma X) = f(X)^q \mod E(X)$.

Proof. The fact that $E(X) = X^{q-1} - \gamma$ is irreducible over $\mathbb{F}_q$ follows from a known, precise characterization of all irreducible binomials, i.e., polynomials of the form $X^a - c$, see for instance [77, Chap. 3, Sec. 5]. For completeness, and since this is an easy special case, we now prove this fact. Suppose $E(X)$ is not irreducible and some irreducible polynomial $f(X) \in \mathbb{F}_q[X]$ of degree b, $1 \leqslant b < q-1$, divides it. Let ζ be a root of $f(X)$ in the extension field $\mathbb{F}_{q^b}$. We then have $\zeta^{q^b-1} = 1$. Also, $f(\zeta) = 0$ implies $E(\zeta) = 0$, which implies $\zeta^{q-1} = \gamma$. These equations together imply $\gamma^{\frac{q^b-1}{q-1}} = 1$. Now, γ is primitive in $\mathbb{F}_q$, so that $\gamma^a = 1$ iff a is divisible by $(q-1)$. We conclude that $q-1$ must divide $\frac{q^b-1}{q-1} = 1+q+q^2+\cdots+q^{b-1}$. This is, however, impossible since $1+q+q^2+\cdots+q^{b-1} \equiv b \pmod{(q-1)}$ and $0 < b < q-1$. This contradiction proves that $E(X)$ has no such factor of degree less than $q-1$, and is therefore irreducible.

For the second part, we have the simple but useful identity $f(X)^q = f(X^q)$ that holds for all polynomials in $\mathbb{F}_q[X]$. Therefore, $f(X)^q - f(\gamma X) = f(X^q) - f(\gamma X)$. Since $X^q = \gamma X$ implies $f(X^q) = f(\gamma X)$, $f(X^q) - f(\gamma X)$ is divisible by $X^q - \gamma X$, and thus also by $X^{q-1} - \gamma$. Hence $f(X)^q \equiv f(\gamma X) \pmod{E(X)}$ which implies that $f(X)^q \mod E(X) = f(\gamma X)$ since the degree of $f(\gamma X)$ is less than $q-1$. □

Armed with this lemma, we are ready to tackle the trivariate root-finding problem.

Lemma 3.5. *There is a deterministic algorithm that on input a finite field* $\mathbb{F}_q$*, a primitive element* γ *of the field* $\mathbb{F}_q$*, a nonzero polynomial* $Q(X, Y_1, Y_2) \in \mathbb{F}_q[X, Y_1, Y_2]$ *of degree less than* q *in* Y_1*, and an integer parameter* $k < q-1$*, outputs a list of all polynomials* $f(X)$ *of degree at most* k *satisfying the condition* $Q(X, f(X), f(\gamma X)) \equiv 0$*. The algorithm has run time polynomial in* q*.*

Proof. Let $E(X) = X^{q-1} - \gamma$. We know by Lemma 3.4 that $E(X)$ is irreducible. We first divide out the largest power of $E(X)$ that divides $Q(X, Y_1, Y_2)$ to obtain $Q_0(X, Y_1, Y_2)$ where $Q(X, Y_1, Y_2) = E(X)^b Q_0(X, Y_1, Y_2)$ for some $b \geqslant 0$ and $E(X)$ does not divide $Q_0(X, Y_1, Y_2)$. Note that as $E(X)$ is irreducible, $f(X)$ does not divide $E(X)$. Thus, if $f(X)$ satisfies $Q(X, f(X), f(\gamma X)) \equiv 0$, then $Q_0(X, f(X), f(\gamma X)) \equiv 0$ as well, so we will work with Q_0 instead of Q. Let us view $Q_0(X, Y_1, Y_2)$ as a polynomial $T_0(Y_1, Y_2)$ with coefficients from $\mathbb{F}_q[X]$. Further, reduce each of the coefficients modulo $E(X)$ to get a polynomial $T(Y_1, Y_2)$ with coefficients from the extension field $\mathbb{F}_{q^{q-1}}$ (which is isomorphic to $\mathbb{F}_q[X]/(E(X))$ as $E(X)$ is irreducible over $\mathbb{F}_q$). We note that $T(Y_1, Y_2)$ is a nonzero polynomial since $Q_0(X, Y_1, Y_2)$ is not divisible by $E(X)$.

In view of Lemma 3.4, it suffices to find degree $\leqslant k$ polynomials $f(X)$ satisfying $Q_0(X, f(X), f(X)^q) \pmod{E(X)} \equiv 0$. In turn, this means it suffices to find elements

$\Gamma \in \mathbb{F}_{q^{q-1}}$ satisfying $T(\Gamma, \Gamma^q) = 0$. If we define the univariate polynomial $R(Y_1) \stackrel{\text{def}}{=} T(Y_1, Y_1^q)$, this is equivalent to finding all $\Gamma \in \mathbb{F}_{q^{q-1}}$ such that $R(\Gamma) = 0$, or in other words the roots in $\mathbb{F}_{q^{q-1}}$ of $R(Y_1)$.

Now $R(Y_1)$ is a nonzero polynomial since $R(Y_1) = 0$ iff $Y_2 - Y_1^q$ divides $T(Y_1, Y_2)$, and this cannot happen as $T(Y_1, Y_2)$ has degree less than q in Y_1. The degree of $R(Y_1)$ is at most dq where d is the total degree of $Q(X, Y_1, Y_2)$. The characteristic of $\mathbb{F}_{q^{q-1}}$ is at most q, and its degree over the base field is at most $q \lg q$. Therefore, by Theorem 2.4 we can find all roots of $R(Y_1)$ by a deterministic algorithm running in time polynomial in d, q. Each of the roots will be a polynomial in $\mathbb{F}_q[X]$ of degree less than $q-1$. Once we find all the roots, we prune the list and only output those roots $f(X)$ that have degree at most k and satisfy $Q_0(X, f(X), f(\gamma X)) = 0$. □

With this, we have a polynomial time implementation of the algorithm Trivariate-FRS-decoder. There is the technicality that the degree of $Q(X, Y_1, Y_2)$ in Y_1 should be less than q. This degree is at most D/k, which by the choice of D in (3.1) is at most $(r+3)\sqrt[3]{n/k} < (r+3)q^{1/3}$. For a fixed r and growing q, the degree is much smaller than q. (In fact, for constant rate codes, the degree is a constant independent of n.) By letting m, r grow in Theorem 3.1, and recalling that the running time is polynomial in n, r, we can conclude the following main result of this section.

Theorem 3.2. *For every $\delta > 0$ and R, $0 < R < 1$, there is a family of m-folded Reed-Solomon codes for m in $O(1/\delta)$ that have rate at least R and that can be list decoded up to a fraction $1 - (1+\delta)R^{2/3}$ of errors in time polynomial in the block length and $1/\delta$.*

Remark 3.2 (Optimality of degree q of relation between $f(X)$ and $f(\gamma X)$). *Let $\mathbb{F}_{q^{q-1}}$ be the extension field $\mathbb{F}_q[X]/(E(X))$ — its elements are in one-one correspondence with polynomials of degree less than $q-1$ over $\mathbb{F}_q$. Let $\Gamma : \mathbb{F}_{q^{q-1}} \to \mathbb{F}_{q^{q-1}}$ be such that for every $f(X) \in \mathbb{F}_{q^{q-1}}$, $\Gamma(f(X)) = f(G(X))$ for some polynomial G over $\mathbb{F}_q$. (In the above, we had $\Gamma(f(X)) = f(X)^q \mod (E(X))$ and $G(X) = \gamma X$; as a polynomial over $\mathbb{F}_{q^{q-1}}$, $\Gamma(Z) = Z^q$, and hence had degree q.) Any such map Γ is an $\mathbb{F}_q$-linear function on $\mathbb{F}_{q^{q-1}}$, and is therefore a* linearized *polynomial, cf. [77, Chap. 3, Sec. 4], which has only terms with exponents that are powers of q (including $q^0 = 1$). It turns out that for our purposes Γ cannot have degree 1, and so it must have degree at least q.*

3.5 Codes Approaching List Decoding Capacity

Given that trivariate interpolation improved the decoding radius achievable with rate R from $1 - R^{1/2}$ to $1 - R^{2/3}$, it is natural to attempt to use higher order interpolation to improve the decoding radius further. In this section, we discuss the quite straightforward technical changes needed for such a generalization.

Consider again the m-folded RS code $C' = \mathrm{FRS}_{\mathbb{F}_q, \gamma, m, k}$. Let s be an integer in the range $1 \leqslant s \leqslant m$. We will develop a decoding algorithm based on interpolating an $(s+1)$-variate

polynomial $Q(X, Y_1, Y_2, \ldots, Y_s)$. The definitions of the $(1, k, k, \ldots, k)$-weighted degree (with k repeated s times) of Q and the multiplicity at a point $(\alpha, \beta_1, \beta_2, \ldots, \beta_s) \in \mathbb{F}_q^{s+1}$ are straightforward extensions of Definitions 3.2 and 3.3.

As before let $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$ be the unfolded version of the received word $\mathbf{z} \in (\mathbb{F}_q^m)^N$ of the folded RS code that needs to be decoded. For convenience, define $y_j = y_{j \mod n}$ for $j \geqslant n$. Following algorithm Trivariate-FRS-decoder, for suitable integer parameters D, r, the interpolation phase of the $(s+1)$-variate FRS decoder will fit a nonzero polynomial $Q(X, Y_1, \ldots, Y_s)$ with the following properties:

1. It has $(1, k, k, \ldots, k)$-weighted degree at most D

2. It has a zero of multiplicity r at $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$ for $i = 0, 1, \ldots, n-1$.

The following is a straightforward generalization of Lemmas 3.2 and 3.3.

Lemma 3.6. *(a) Provided $\frac{D^{s+1}}{(s+1)!k^s} > n\binom{r+s}{s+1}$, a nonzero polynomial $Q(X, Y_1, \ldots, Y_s)$ with the following properties exists. $Q(X, Y_1, \ldots, Y_s)$ has $(1, k, \ldots, k)$ weighted degree at most D and has roots with multiplicity r at $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$ for every $i \in \{0, \ldots, n-1\}$. Moreover such a $Q(X, Y_1, \ldots, Y_s)$ can be found in time polynomial in n, r^s and D^{s+1}/k^s.*

(b) Let t be an integer such that $t > \frac{D}{(m-s+1)r}$. Then every polynomial $f(X) \in \mathbb{F}_q[X]$ of degree at most k whose encoding according to $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ agrees with the received word $\mathbf{z}$ on at least t locations satisfies $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1}X)) \equiv 0$.

Proof. The first part follows from (i) a simple lower bound on the number of monomials $X^a Y_1^{b_1} \cdots Y_s^{b_s}$ with $a + k(b_1 + b_2 + \cdots + b_s) \leqslant D$, which gives the number of coefficients of $Q(X, Y_1, \ldots, Y_s)$, and (ii) an estimation of the number of $(s+1)$-variate monomials of total degree less than r, which gives the number of interpolation conditions per $(s+1)$-tuple. We now briefly justify these claims. By a generalization of the argument in Lemma 3.2, one can lower bound the number of monomials $X^a Y_1^{b_1} \cdots Y_s^{b_s}$ such that $a + k(b_1 + \cdots b_s) \leqslant D$ by the volume of $\mathcal{P}_{s,D} = \{x + ky_1 + ky_2 + \cdots + ky_s \leqslant D | x, y_1, y_2, \ldots y_s \geqslant 0\}$. We will use induction on s to prove that the volume of $\mathcal{P}_{s,D}$ is $\frac{D^{s+1}}{(s+1)!k^s}$. The proof of Lemma 3.2 shows this for $s = 2$. Now assume that the volume of $\mathcal{P}_{s-1,D}$ is exactly $\frac{D^s}{s!k^{s-1}}$. Note that the subset of $\mathcal{P}_{s,D}$ where the value of $y_s = \alpha$ is fixed is exactly $\mathcal{P}_{s-1,D-k\alpha}$ Thus, the volume of $\mathcal{P}_{s,D}$ is exactly

$$\int_0^{D/k} \frac{(D - ky_s)^s}{s!k^{s-1}}\, dy_s = \frac{1}{s!k^s} \int_0^D z^s\, dz = \frac{D^{s+1}}{(s+1)!k^s},$$

where the second equality follows by substituting $z = D - ky_s$. Further, a straightforward generalization of the argument in the proof of Lemma 3.2, shows that the condition on the

multiplicity of the polynomial $Q(X, Y_1, \ldots, Y_s)$ is satisfied if for every $i \in \{0, \ldots, n-1\}$ and every tuple $(l, j_1, \ldots, j_s)$ such that $l + j_1 + j_2 \cdots + j_s \leqslant r$ the following is 0

$$\sum_{l' \geqslant l} \sum_{j'_1 \geqslant j_1} \sum_{j'_2 \geqslant j_2} \cdots \sum_{j'_s \geqslant j_s} \binom{l'}{l} \binom{j'_1}{j_1} \binom{j'_2}{j_2} \cdots \binom{j'_s}{j_s} q_{l', j'_1, i'_2, \ldots, j'_s} \gamma^{i(l'-l)} y_i^{j'_1 - g_1} y_{i+1}^{j'_2 - j_2} \cdots y_{i+s-1}^{j'_s - j_s},$$

where $q_{l', j'_1, j'_2, \ldots, j'_s}$ is the coefficient of the monomial $X^{l'} Y_1^{j'_1} \cdots Y_s^{j'_s}$ in $Q(X, Y_1, \ldots, Y_s)$. The number of positive integral solutions for $i + j_1 + j_2 \cdots + j_s \leqslant r$ is exactly $\binom{r+s}{s+1}$. Thus, the total number of constraints is $n\binom{r+s}{s+1}$. Thus, the condition in part (a) of the lemma, implies that the set of homogeneous linear equations have more variables than constraints. Hence, a solution can be found in time polynomial in the number of variables ($\leqslant D^{s+1}/k^s$) and constraints (at most $nr^{O(s)}$).

The second part is similar to the proof of Lemma 3.3. If $f(X)$ has agreement on at least t locations of $\mathbf{z}$, then for at least $t(m-s+1)$ of the $(s+1)$-tuples $(\gamma^i, y_i, y_{i+1}, \ldots, y_{i+s-1})$, we have $f(\gamma^{i+j}) = y_{i+j}$ for $j = 0, 1, \ldots, s-1$. As in Lemma 3.1, we conclude that $R(X) \stackrel{\text{def}}{=} Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1} X))$ has a zero of multiplicity r at γ^i for each such $(s+1)$-tuple. Also, by design $R(X)$ has degree at most D. Hence if $t(m-s+1)r > D$, then $R(X)$ has more zeroes (counting multiplicities) than its degree, and thus $R(X) \equiv 0$. $\square$

Note the lower bound condition on D above is met with the choice

$$D = \left\lfloor (k^s n r(r+1) \cdots (r+s))^{1/(s+1)} \right\rfloor + 1 \,. \tag{3.2}$$

The task of finding the list of all degree k polynomials $f(X) \in \mathbb{F}_q[X]$ satisfying $Q(X, f(X), f(\gamma X), \ldots, f(\gamma^{s-1} X)) = 0$ can be solved using ideas similar to the proof of Lemma 3.5. First, by dividing out by $E(X)$ enough times, we can assume that not all coefficients of $Q(X, Y_1, \ldots, Y_s)$, viewed as a polynomial in $Y_1, \ldots, Y_s$ with coefficients in $\mathbb{F}_q[X]$, are divisible by $E(X)$. We can then go modulo $E(X)$ to get a nonzero polynomial $T(Y_1, Y_2, \ldots, Y_s)$ over the extension field $\mathbb{F}_{q^{q-1}} = \mathbb{F}_q[X]/(E(X))$. Now, by Lemma 3.4, we have $f(\gamma^j X) = f(X)^{q^j} \mod E(X)$ for every $j \geqslant 1$. Therefore, the task at hand reduces to the problem of finding all roots $\Gamma \in \mathbb{F}_{q^{q-1}}$ of the polynomial $R(Y_1)$ where $R(Y_1) = T(Y_1, Y_1^q, \ldots, Y_1^{q^{s-1}})$. There is the risk that $R(Y_1)$ is the zero polynomial, but it is easily seen that this cannot happen if the total degree of T is less than q. This will be the case since the total degree is at most D/k, which is at most $(r+s)(n/k)^{1/(s+1)} \ll q$.

The degree of the polynomial $R(Y_1)$ is at most q^s, and therefore all its roots in $\mathbb{F}_{q^{q-1}}$ can be found in $q^{O(s)}$ time (by Theorem 2.4). We conclude that the "root-finding" step can be accomplished in polynomial time.

The algorithm works for agreement $t > \frac{D}{(m-s+1)r}$, which for the choice of D in (3.2) is satisfied if

$$t \geqslant \left(1 + \frac{s}{r}\right) \frac{(k^s n)^{1/(s+1)}}{m-s+1} + 2 \,.$$

Indeed,

$$\begin{aligned}
\frac{D}{(m-s+1)r} &\leqslant \frac{1}{(m-s+1)r} \cdot \left(\sqrt[s+1]{k^s n r(r+1)\cdots(r+s)} + 1 \right) \\
&\leqslant \frac{1}{(m-s+1)r} \cdot \left((r+s)\sqrt[s+1]{k^s n} + 1 \right) \\
&= \left(1 + \frac{s}{r}\right) \frac{(k^s n)^{1/(s+1)}}{m-s+1} + \frac{1}{r(m-s+1)} \\
&< \left(1 + \frac{s}{r}\right) \frac{(k^s n)^{1/(s+1)}}{m-s+1} + 2 \\
&\leqslant t,
\end{aligned}$$

where the first inequality follows from (3.2) along with the fact that for any real $x \geqslant 0$, $\lfloor x \rfloor \leqslant x$ while the second inequality follows by upper bounding $r+i$ by $r+s$ for every $0 \leqslant i \leqslant s$. We record these observations in the following, which is a multivariate generalization of Theorem 3.1.

Theorem 3.3. *For every integer $m \geqslant 1$ and every s, $1 \leqslant s \leqslant m$, the $(s+1)$-variate FRS decoder successfully list decodes the m-folded Reed-Solomon code $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ up to a radius $n/m - t$ as long as the agreement parameter t satisfies*

$$t \geqslant \left(1 + \frac{s}{r}\right) \frac{(k^s n)^{1/(s+1)}}{m-s+1} + 2\,. \tag{3.3}$$

The algorithm runs in $n^{O(s)}$ time and outputs a list of size at most $|F|^s = (n+1)^s$.

Recalling that the block length of $\mathrm{FRS}_{\mathbb{F}_q,\gamma,m,k}$ is $N = n/m$ and the rate is $(k+1)/n$, the above algorithm can decode a fraction of errors approaching

$$1 - \left(1 + \frac{s}{r}\right) \frac{m}{m-s+1} R^{s/(s+1)} \tag{3.4}$$

using lists of size at most q^s. By picking r, m large enough compared to s, the decoding radius can be made larger than $1 - (1+\delta)R^{s/(s+1)}$ for any desired $\delta > 0$. We state this result formally below.

Theorem 3.4. *For every $0 < \delta \leqslant 1$, integer $s \geqslant 1$ and $0 < R < 1$, there is a family of m-folded Reed-Solomon codes for $m \leqslant 4s/\delta$ that have rate at least R and which can be list decoded up to a $1 - (1+\delta)R^{s/(s+1)}$ fraction of errors in time $(Nm)^{O(s)}$ and outputs a list of size at most $(Nm)^{O(s)}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(Nm)^{O(m)}$.*

Proof. We first instantiate the parameters r and m in terms of s and δ:

$$r = \frac{3s}{\delta} \qquad m = \frac{(s-1)(3+\delta)}{\delta}\,.$$

Note that as $\delta \leqslant 1$, $m \leqslant 4s/\delta$. With the above choice, we have

$$\left(1 + \frac{s}{r}\right) \frac{m}{m - s + 1} = \left(1 + \frac{\delta}{3}\right)^2 < 1 + \delta \, .$$

Together with the bound (3.4) on the decoding radius, we conclude that the $(s+1)$-variate decoding algorithm certainly list decodes up to a fraction $1 - (1+\delta)R^{s/(s+1)}$ of errors.

The worst case list size is q^s and the claim on the list size follows by recalling that $q = n + 1$ and $N = n/m$. The alphabet size is $q^m = (Nm)^{O(m)}$. The running time has two major components: (1) Interpolating the $s+1$-variate polynomial $Q(\cdot)$, which by Lemma 3.6 is $(nr^s)^{O(1)}$; and (2) Finding all the roots of the interpolated polynomial, which takes $q^{O(s)}$ time. Of the two, the time complexity of the root finding step dominates, which is $(Nm)^{O(s)}$. □

In the limit of large s, the decoding radius approaches the list-decoding capacity $1 - R$, leading to our main result.

Theorem 3.5 (Explicit capacity-approaching codes). *For every $0 < R < 1$ and $0 < \varepsilon \leqslant R$, there is a family of folded Reed-Solomon codes that have rate at least R and which can be list decoded up to a $1 - R - \varepsilon$ fraction of errors in time (and outputs a list of size at most) $(N/\varepsilon^2)^{O(\varepsilon^{-1}\log(1/R))}$ where N is the block length of the code. The alphabet size of the code as a function of the block length N is $(N/\varepsilon^2)^{O(1/\varepsilon^2)}$.*

Proof. Given ε, R, we will apply Theorem 3.4 with the choice

$$s = \left\lceil \frac{\log(1/R)}{\log(1+\varepsilon)} \right\rceil \quad \text{and} \quad \delta = \frac{\varepsilon(1-R)}{R(1+\varepsilon)} \, . \tag{3.5}$$

Note that as $\varepsilon \leqslant R$, $\delta \leqslant 1$. Thus, the list-decoding radius guaranteed by Theorem 3.4 is at least

$$\begin{aligned} 1 - (1+\delta)R^{s/(s+1)} &= 1 - R(1+\delta)(1/R)^{1/(s+1)} \\ &\geqslant 1 - R(1+\delta)(1+\varepsilon) \ \text{(by the choice of } s \text{ in (3.5))} \\ &= 1 - (R + \varepsilon) \ \text{(using the value of } \delta) \, . \end{aligned}$$

We now turn our attention to the time complexity of the decoding algorithm and the alphabet size of the code. To this end we first claim that m is $O(1/\varepsilon^2)$. First we note that by the choice of s,

$$s \leqslant \frac{2\ln(1/R)}{\ln(1+\varepsilon)} \leqslant \frac{4\ln(1/R)}{\varepsilon},$$

where the second inequality follows from the fact that for $0 < x \leqslant 1$, $\ln(1+x) \geqslant x/2$. Thus, we have

$$m \leqslant \frac{4s}{\delta} = 4s \cdot \frac{R(1+\varepsilon)}{\varepsilon(1-R)} \leqslant 8s \cdot \frac{R}{\varepsilon(1-R)} \leqslant \frac{32}{\varepsilon^2} \cdot \frac{R\ln(1/R)}{1-R} \leqslant \frac{32}{\varepsilon^2} \, ,$$

where for the last step we used $\ln(1/R) \leqslant \frac{1}{R} - 1$ for $0 < R \leqslant 1$. The claims on the running time, worst case list size and the alphabet size of the code follow from Theorem 3.4 and the facts that m is $O(1/\varepsilon^2)$ and s is $O(\varepsilon^{-1}\log(1/R))$. □

Remark 3.3 (Upper bound on ε in Theorem 3.5). *A version of Theorem 3.5 can also be proven for $\varepsilon > R$. The reason it is stated for $\varepsilon \leqslant R$, is that we generally think of ε as much smaller than R (this is certainly true when we apply the generalization of Theorem 3.5 (Theorem 3.6) in Chapters 4 and 5). However, if one wants $\varepsilon \geqslant R$, first note that the theorem is trivial for $\varepsilon \geqslant 1 - R$. Then if $R < \varepsilon < 1 - R$, can do a proof similar to the one above. However, in this range $\delta = \frac{\varepsilon(1-R)}{R(1+\varepsilon)}$ can be strictly greater than 1. In such a case we apply Theorem 3.4 with $\delta = 1$ (note that applying Theorem 3.4 with a smaller δ than what we want only increases the decoding radius). This implies that we have $m \leqslant 4s$, in which case both the worst case list and the alphabet size become $(N\log(1/R)/\varepsilon)^{\varepsilon^{-1}\log(1/R)}$.*

Remark 3.4 (Minor improvement to decoding radius). *It is possible to slightly improve the bound of (3.4) to $1 - \left(1 + \frac{s}{r}\right)\left(\frac{mR}{m-s+1}\right)^{s/(s+1)}$ with essentially no effort. The idea is to not use only a fraction $(m-s+1)/m$ of the n $(s+1)$-tuples for interpolation. Specifically, we omit tuples with γ^i for $i \mod m > m - s$. This does not affect the number of $(s+1)$-tuples for which we have agreement (this remains at least $t(m-s+1)$), but the number of interpolation conditions is reduced to $N(m-s+1) = n(m-s+1)/m$. This translates into the stated improvement in list-decoding radius. For clarity of presentation, we simply chose to use all n tuples for interpolation.*

Remark 3.5 (Average list size). *Theorem 3.5 states that the worst case list size (over all possible received words) is polynomial in the block length of the codeword (for fixed R and ε). One might also be interested in what is the* average *list size (over all the possible received words within a distance ρn from some codeword). It is known that for Reed-Solomon codes of rate R the average list size is $\ll 1$ even for ρ close to $1 - R$ [81]. Since folded Reed-Solomon codes are just Reed-Solomon codewords with symbols bundled together, the arguments in [81] extend easily to show that even for folded Reed-Solomon codes, the average list size is $\ll 1$.*

3.6 Extension to List Recovery

We now present a very useful generalization of the list decoding result of Theorem 3.5 to the setting of list recovery. Recall that under the list recovery problem, one is given as input for each codeword position, not just one but a set of several, say ℓ, alphabet symbols. The goal is to find and output all codewords which agree with some element of the input sets for several positions. Codes for which this more general problem can be solved turn out to be extremely valuable as outer codes in concatenated code constructions. In short, this is because one can pass a set of possibilities from decodings of the inner codes and then list recover the outer code with those sets as the input. If we only had a list-decodable code at

the outer level, we will be forced to make a unique choice in decoding the inner codes thus losing valuable information.

This is a good time to recall the definition of list recoverable codes (Definition 2.4).

Theorem 3.5 can be generalized to list recover the folded RS codes. Specifically, for a FRS code with parameters as in Section 3.5, for an arbitrary constant $\ell \geqslant 1$, we can (ζ, ℓ)-list recover in polynomial time provided

$$(1-\zeta)N \geqslant \left(1+\frac{s}{r}\right)\frac{\sqrt[s+1]{n\ell k^s}}{m-s+1}, \tag{3.6}$$

where $N = n/m$. We briefly justify this claim. The generalization of the list-decoding algorithm of Section 3.5 is straightforward: instead of one interpolation condition for each symbol of the received word, we just impose $|S_i| \leqslant \ell$ many interpolation conditions for each position $i \in \{1, 2, \ldots, n\}$ (where S_i is the i'th input set in the list recovery instance). The number of interpolation conditions is at most $n\ell$, and so replacing n by $n\ell$ in the bound of Lemma 3.6 guarantees successful decoding[2]. This in turn implies that the condition on the number of agreement of (3.3) generalizes to the one in (3.6).[3] This simple generalization to list recovery is a positive feature of all interpolation based decoding algorithms [97, 63, 85] beginning with the one due to Sudan [97].

Picking $r \gg s$ and $m \gg s$ in (3.6), we get (ζ, ℓ)-list recoverable codes with rate R for $\zeta \leqslant 1 - \left(\ell R^s\right)^{1/(s+1)}$. Now comes the remarkable fact: we can pick a suitable $s \gg \ell$ and perform (ζ, ℓ)-list recovery with $\zeta \leqslant 1 - R - \varepsilon$ which is independent of ℓ ! We state the formal result below (Theorem 3.5 is a special case when $\ell = 1$).

Theorem 3.6. *For every integer $\ell \geqslant 1$, for all R, $0 < R < 1$ and $0 < \varepsilon \leqslant R$, and for every prime p, there is an* explicit *family of folded Reed-Solomon codes over fields of characteristic p that have rate at least R and which are $(1-R-\varepsilon, \ell, L(n))$-list recoverable in polynomial time, where $L(n) = (N/\varepsilon^2)^{O(\varepsilon^{-1}\log(\ell/R))}$. The alphabet size of a code of block length N in the family is $(N/\varepsilon^2)^{O(\varepsilon^{-2}\log \ell/(1-R))}$.*

Proof. (Sketch) Using the exact same arguments as in the proof of Theorem 3.4 to the agreement condition of (3.6), we get that one can list recover in polynomial time as long as $\zeta \leqslant 1-(1+\delta)(\ell R^s)^{1/(s+1)}$, for any $0 < \delta \leqslant 1$. The arguments to obtain an upper bound of $1 - R - \varepsilon$ are similar to the ones employed in the proof of theorem 3.5. However, s needs to be defined in a slightly different manner:

$$s = \left\lceil \frac{\log(\ell/R)}{\log(1+\varepsilon)} \right\rceil.$$

[2]In fact, this extension also works when the average size of the size is at most ℓ, that is $\sum_{i=1}^{n} |S_i| \leqslant \ell n$.

[3]We will also need the condition that $(r+s)(n\ell/k)^{1/(s+1)} < q$. This condition is required to argue that in the "root finding" step, the "final" polynomial $R(Y_1)$ is not the zero polynomial. The condition is met for constant rate codes if $\ell \ll q^s$ (recall that we think of q as growing while r and s are fixed). In all our applications of list recovery for folded Reed-Solomon codes, the parameter ℓ will be a constant, so this is not a concern.

Also this implies that m is $O\left(\frac{\log \ell}{(1-R)\varepsilon^2}\right)$, which implies the claimed bound on the alphabet size of the code as well as $L(n)$. □

We also note that increasing the folding parameter m only helps improve the result (at the cost of a larger alphabet). In particular, we have the following corollary of the theorem above.

Corollary 3.7. *For every integer $\ell \geqslant 1$, for all constants $0 < \varepsilon \leqslant R$, for all R, R'; $0 < R \leqslant R' < 1$, and for every prime p, there is an* explicit *family of folded Reed-Solomon codes, over fields of characteristic p that have rate at least R and which can be $(1 - R - \varepsilon, \ell, L(N))$-list recovered in polynomial time, where for codes of block length N, $L(N) = (N/\varepsilon^2)^{O(\varepsilon^{-1}\log(\ell/R))}$ and the code is defined over alphabet of size $(N/\varepsilon^2)^{O(\varepsilon^{-2}\log \ell/(1-R'))}$.*

Note that one can trivially increase the alphabet of a code by thinking of every symbol as coming from a larger alphabet. However, this trivial transformation decreases the rate of the code. Corollary 3.7 states that for folded Reed-Solomon codes, we can increase the alphabet while retaining the rate and the list recoverability properties. At this point this extra feature is an odd one to state explicitly, but we will need this result in Chapter 4.

Remark 3.6 (Soft Decoding). *The decoding algorithm for folded RS codes from Theorem 3.5 can be further generalized to handle soft information, where for each codeword position i the decoder is given as input a non-negative weight $w_{i,z}$ for each possible alphabet symbol z. The weights $w_{i,z}$ can be used to encode the confidence information concerning the likelihood of the the i'th symbol of the codeword being z. For any $\varepsilon > 0$, for suitable choice of parameters, our codes of rate R over alphabet Σ have a soft decoding algorithm that outputs all codewords $c = \langle c_1, c_2, \ldots, c_N \rangle$ that satisfy*

$$\sum_{i=1}^{N} w_{i,c_i} \geqslant \left((1+\varepsilon)(RN)^s \Big(\sum_{i=1}^{N}\sum_{z\in\Sigma} w_{i,z}^{s+1}\Big)\right)^{1/(s+1)}.$$

For $s = 1$, this soft decoding condition is identical to the one for Reed-Solomon codes in [63].

3.7 Bibliographic Notes and Open Questions

We have solved the qualitative problem of achieving list-decoding capacity over large alphabets. Our work could be improved with some respect to some parameters. The size of the list needed to perform list decoding to a radius that is within ε of capacity grows as $n^{O(1/\varepsilon)}$ where n is the block length of the code. It remains an open question to bring this list size down to a constant independent of n, or even to $f(\varepsilon)n^c$ with an exponent c independent of ε (we recall that the existential random coding arguments work with a list size of $O(1/\varepsilon)$).

These results in this chapter were first reported in [58]. We would like to point out that the presentation in this chapter is somewhat different from the original papers [85, 58] in terms of technical details, organization, as well as chronology. Our description closely follows that of a survey by Guruswami [50]. With the benefit of hindsight, we believe this alternate presentation to be simpler and more self-contained than the description in [58], which used the results of Parvaresh-Vardy as a black-box. Below, we discuss some technical aspects of the original development of this material, in order to shed light on the origins of our work.

Two independent works by Coppersmith and Sudan [27] and Bleichenbacher, Kiayias and Yung [18] considered the variant of RS codes where the message consists of two (or more) independent polynomials over some field $\mathbb{F}_q$, and the encoding consists of the joint evaluation of these polynomials at elements of $\mathbb{F}_q$ (so this defines a code over $\mathbb{F}_q^2$).[4] A naive way to decode these codes, which are also called "interleaved Reed-Solomon codes," would be to recover the two polynomials individually, by running separate instances of the RS decoder. Of course, this gives no gain over the performance of RS codes. The hope in these works was that something can possibly be gained by exploiting that errors in the two polynomials happen at "synchronized" locations. However, these works could not give any improvement over the $1 - \sqrt{R}$ bound known for RS codes for worst-case errors. Nevertheless, for *random errors*, where each error replaces the correct symbol by a uniform random field element, they were able to correct well beyond a fraction $1 - \sqrt{R}$ of errors. In fact, as the order of interleaving (i.e., number of independent polynomials) grows, the radius approaches the optimal value $1 - R$. This model of random errors is not very practical or interesting in a coding-theoretic setting, though the algorithms are interesting from an algebraic viewpoint.

The algorithm of Coppersmith and Sudan bears an intriguing relation to multivariate interpolation. Multivariate interpolation essentially amounts to finding a non-trivial linear dependence among the rows of a certain matrix (that consists of the evaluations of appropriate monomials at the interpolation points). The algorithm in [27], instead finds a non-trivial linear dependence among the *columns* of this same matrix! The positions corresponding to columns not involved in this dependence are erased (they correspond to error locations) and the codeword is recovered from the remaining symbols using erasure decoding.

In [84], Parvaresh and Vardy gave a heuristic decoding algorithm for these interleaved RS codes based on multivariate interpolation. However, the provable performance of these codes coincided with the $1 - \sqrt{R}$ bound for Reed-Solomon codes. The key obstacle in improving this bound was the following: for the case when the messages are pairs $(f(X), g(X))$ of degree k polynomials, two algebraically independent relations were needed to identify both $f(X)$ and $g(X)$. The interpolation method could only provide one such relation in general (of the form $Q(X, f(X), g(X)) = 0$ for a trivariate polynomial $Q(X, Y, Z)$). This still left too much ambiguity in the possible values of $(f(X), g(X))$. (The approach

[4] The resulting code is in fact just a Reed-Solomon code where the evaluation points belong to the subfield $\mathbb{F}_q$ of the extension field over $\mathbb{F}_q$ of degree two.

in [84] was to find several interpolation polynomials, but there was no guarantee that they were not all algebraically dependent.)

Then, in [85], Parvaresh and Vardy put forth the ingenious idea of obtaining the extra algebraic relation essentially "for free" by enforcing it as an *a priori* condition satisfied at the encoder. Specifically, instead of letting the second polynomial $g(X)$ to be an independent degree k polynomial, their insight was to make it correlated with $f(X)$ by a specific algebraic condition, such as $g(X) = f(X)^d \mod E(X)$ for some integer d and an irreducible polynomial $E(X)$ of degree $k+1$.

Then, once we have the interpolation polynomial $Q(X, Y, Z)$, $f(X)$ can be obtained as follows: Reduce the coefficients of $Q(X, Y, Z)$ modulo $E(X)$ to get a polynomial $T(Y, Z)$ with coefficients from $\mathbb{F}_q[X]/(E(X))$ and then find roots of the univariate polynomial $T(Y, Y^d)$. This was the key idea in [85] to improve the $1 - \sqrt{R}$ decoding radius for rates less than $1/16$. For rates $R \to 0$, their decoding radius approached $1 - O(R \log(1/R))$.

The modification to using independent polynomials, however, does not come for free. In particular, since one sends at least twice as much information as in the original RS code, there is no way to construct codes with rate more than $1/2$ in the PV scheme. If we use $s \geqslant 2$ correlated polynomials for the encoding, we incur a factor $1/s$ loss in the rate. This proves quite expensive, and as a result the improvements over RS codes offered by these codes are only manifest at very low rates.

The central idea behind our work is to avoid this rate loss by making the correlated polynomial $g(X)$ essentially identical to the first (say $g(X) = f(\gamma X)$). Then the evaluations of $g(X)$ can be inferred as a simple cyclic shift of the evaluations of $f(X)$, so intuitively there is no need to explicitly include those too in the encoding.

Chapter 4

RESULTS VIA CODE CONCATENATION

4.1 Introduction

In Chapter 3, we presented efficient list-decoding algorithms for folded Reed-Solomon codes that can correct $1 - R - \varepsilon$ fraction of errors with rate R (for any $\varepsilon > 0$). One drawback of folded Reed-Solomon codes is that they are defined over alphabets whose size is polynomial in the blocklength of the code. This is an undesirable feature of the code and we address this issue in this chapter.

First, we show how to convert folded Reed-Solomon codes to a related code that can still be list decoded up to $1-R-\varepsilon$ fraction of errors with rate R (for any $\varepsilon > 0$). However, unlike folded Reed-Solomon codes these codes are defined over alphabets of size $2^{O(\varepsilon^{-4}\log(1/\varepsilon))}$. Recall that codes that can be list decoded up to $1 - R - \varepsilon$ fraction of errors need alphabets of size $2^{\Omega(\varepsilon^{-1})}$ (see section 2.2.1).

Next, we will show how to use folded Reed-Solomon codes to obtain codes over *fixed* alphabets (for example, binary codes). We will present explicit linear codes over fixed alphabets that achieve tradeoffs between rate and fraction of errors that satisfy the so called Zyablov and Blokh-Zyablov bounds (along with efficient list-decoding algorithms that achieve these tradeoffs). The codes list decodable up to the Blokh-Zyablov bound tradeoff are the best known to date for explicit codes over fixed alphabets. However, unlike Chapter 3, these results do not get close to the list-decoding capacity (see Figure 4.1). In particular, for binary codes, if $1/2 - \gamma$ fraction of errors are targeted, our codes have rate $\Omega(\gamma^3)$. By contrast, codes on list-decoding capacity will have rate $\Omega(\gamma^2)$. Unfortunately (as has been mentioned before), the only codes that are known to achieve list-decoding capacity are random codes for which no efficient list-decoding algorithms are known. Previous to our work, the best known explicit codes had rate $\Theta(\gamma^4)$ [51] (these codes also had efficient list-decoding algorithms). We choose to present the codes that are list decodable up to the Zyablov bound (even though the code that are list decodable up to the Blokh Zyablov have better rate vs. list decodability tradeoff) because of the following reasons (i) The construction is much simpler and these codes give the same asymptotic rate for the high error regime and (ii) The worst case list sizes and the code construction time are asymptotically smaller.

All our codes are based on code concatenation (and their generalizations called multi-level code concatenation). We next turn to an informal description of code concatenation.

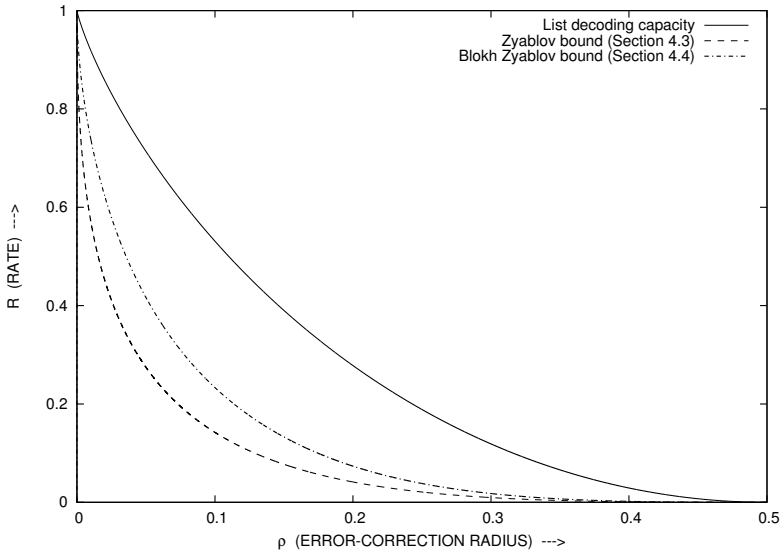


Figure 4.1: Rate R of our binary codes plotted against the list-decoding radius ρ of our algorithms. The best possible trade-off, i.e., list-decoding capacity, $\rho = H^{-1}(1-R)$ is also plotted.

4.1.1 Code Concatenation and List Recovery

Concatenated codes were defined in the seminal thesis of Forney [40]. Concatenated codes are constructed from two different codes that are defined over alphabets of different sizes. Say we are interested in a code over $[q]$ (in this chapter, we will always think of $q \geqslant 2$ as being a fixed constant). Then the *outer code* C_{out} is defined over $[Q]$, where $Q = q^k$ for some positive integer k. The second code, called the *inner code* is defined over $[q]$ and is of dimension k (Note that the message space of C_{in} and the alphabet of C_{out} have the same size). The concatenated code, denoted by $C = C_{out} \circ C_{in}$, is defined as follows. Let the rate of C_{out} be R and let the blocklengths of C_{out} and C_{in} be N and n respectively. Define $K = RN$ and $r = k/n$. The input to C is a vector $\mathbf{m} = \langle m_1, \ldots, m_K \rangle \in ([q]^k)^K$. Let $C_{out}(\mathbf{m}) = \langle x_1, \ldots, x_N \rangle$. The codeword in C corresponding to $\mathbf{m}$ is defined as follows

$$C(\mathbf{m}) = \langle C_{in}(x_1), C_{in}(x_2), \ldots, C_{in}(x_N) \rangle.$$

It is easy to check that C has rate rR, dimension kK and blocklength nN.

Notice that to construct a q-ary code C we use another q-ary code C_{in}. However, the nice thing about C_{in} is that it has small blocklength. In particular, since R and r are constants (and typically Q and N are polynomially related), $n = O(\log N)$. This implies that we can use up exponential time (in n) to search for a "good" inner code. Further, one can use the brute force algorithm to (list) decode C_{in}.

Table 4.1: Values of rate at different decoding radius for List decoding capacity (R_{Cap}), Zyablov bound (R_Z) and Blokh Zyablov bound (R_{BZ}) in the binary case. For rates above 0.4, the Blokh Zyablov bound is 0 up to 3 decimal places, hence we have not shown this.

ρ	0.01	0.02	0.03	0.05	0.10	0.15	0.20	0.25	0.30	0.35
R_{Cap}	0.919	0.858	0.805	0.713	0.531	0.390	0.278	0.188	0.118	0.065
R_Z	0.572	0.452	0.375	0.273	0.141	0.076	0.041	0.020	0.009	0.002
R_{BZ}	0.739	0.624	0.539	0.415	0.233	0.132	0.073	0.037	0.017	0.006

Finally, we motivate why we are interested in list recovery. Consider the following natural decoding algorithm for the concatenated code $C_{out} \circ C_{in}$. Given a received word in $([q]^n)^N$, we divide it into N blocks from $[q]^n$. Then we use a decoding algorithm for C_{in} to get an intermediate received word to feed into a decoding algorithm for C_{out}. Now one can use unique decoding for C_{in} and list decoding for C_{out}. However, this loses information in the first step. Instead, one can use the brute force list-decoding algorithm for C_{in} to get a sequence of lists (each of which is a subset of $[Q]$). Now we use a list-recovery algorithm for C_{out} to get the final list of codewords.

The natural algorithm above is used to design codes over fixed alphabets that are list decodable up to the Zyablov bound in Section 4.3 and (along with expanders) to design codes that achieve list-decoding capacity (but have much smaller alphabet size as compared to those for folded Reed-Solomon codes) in Section 4.2.

4.2 Capacity-Achieving Codes over Smaller Alphabets

Theorem 3.5 has two undesirable aspects: both the alphabet size and worst-case list size output by the list-decoding algorithm are a polynomial of large degree in the block length. We now show that the alphabet size can be reduced to a constant that depends only on the distance ε to capacity.

Theorem 4.1. *For every R, $0 < R < 1$, every $\varepsilon > 0$, there is a polynomial time constructible family of codes over an alphabet of size $2^{O(\varepsilon^{-4}\log(1/\varepsilon))}$ that have rate at least R and which can be list decoded up to a fraction $(1 - R - \varepsilon)$ of errors in polynomial time.*

Proof. The theorem is proved using the code construction scheme used by Guruswami and Indyk in [54] for linear time unique decodable codes with optimal rate, with different components appropriate for list decoding plugged in. We briefly describe the main ideas behind the construction and proof below. The high level approach is to concatenate two codes C_{out} and C_{in}, and then redistribute the symbols of the resulting codeword using an expander graph. Assume that $\varepsilon < (1 - R)/7$ and let $\delta = \varepsilon^2$.

The outer code C_{out} will be a code of rate $(1 - 2\varepsilon)$ over an alphabet Σ of size $n^{(1/\delta)^{O(1)}}$ that can be $(\varepsilon, O(1/\varepsilon))$-list recovered in polynomial time (to recall definitions pertaining to

list recovery, see Definition 2.4), as guaranteed by Theorem 3.6. That is, the rate of C_{out} will be close to 1, and it can be (ζ, l)-list recovered for large l and $\zeta \to 0$.

The inner code C_{in} will be a $((1 - R - 4\varepsilon), O(1/\varepsilon))$-list decodable code with near-optimal rate, say rate at least $(R + 3\varepsilon)$. Such a code is guaranteed to exist over an alphabet of size $O(1/\varepsilon^2)$ using random coding arguments. A naive brute-force for such a code, however, is too expensive, since we need a code with $|\Sigma| = n^{\Omega(1)}$ codewords. Guruswami and Indyk [52], see also [49, Sec. 9.3], prove that there is a small (quasi-polynomial sized) sample space of *pseudolinear codes* in which most codes have the needed property. Furthermore, they also present a deterministic polynomial time construction of such a code (using derandomization techniques), see [49, Sec. 9.3.3].

The concatenation of C_{out} and C_{in} gives a code C_{concat} of rate at least $(1 - 2\varepsilon)(R + 3\varepsilon) \geqslant R$ over an alphabet Σ of size $|\Sigma| = O(1/\varepsilon^2)$. Moreover, given a received word of the concatenated code, one can find all codewords that agree with the received word on a fraction $R + 4\varepsilon$ of locations in at least $(1 - \varepsilon)$ fraction of the inner blocks. Indeed, we can do this by running the natural list-decoding algorithm, call it $\mathcal{A}$, for C_{concat} that decodes each of the inner blocks to a radius of $(1 - R - 4\varepsilon)$ returning up to $l = O(1/\varepsilon)$ possibilities for each block, and then (ε, l)-list recovering C_{out} in polynomial time.

The last component in this construction is a $D = O(1/\varepsilon^4)$-regular bipartite expander graph which is used to redistribute symbols of the concatenated code in a manner so that an overall agreement on a fraction $R + 7\varepsilon$ of the redistributed symbols implies a fractional agreement of at least $R + 4\varepsilon$ on most (specifically a fraction $(1 - \varepsilon)$) of the inner blocks of the concatenated code. In other words, the expander redistributes symbols in a manner that "smoothens" the distributions of errors evenly among the various inner blocks (except for possibly a ε fraction of the blocks). This expander based redistribution incurs no loss in rate, but increases the alphabet size to $O(1/\varepsilon^2)^{O(1/\varepsilon^4)} = 2^{O(\varepsilon^{-4} \log(1/\varepsilon))}$.

We now discuss some details of how the expander is used. Suppose that the block length of the folded Reed-Solomon code C_{out} is N_1 and that of C_{in} is N_2. Let us assume that N_2 is a multiple of D, say $N_2 = n_2 D$ (if this is not the case, we can make it so by padding at most $D - 1$ dummy symbols at a negligible loss in rate). Therefore codewords of C_{in}, and therefore also of C_{concat}, can be thought of as being composed of blocks of D symbols each. Let $N = N_1 n_2$, so that codewords of C_{concat} can be viewed as elements in $(\Sigma^D)^N$.

Let $G = (L, R, E)$ be a D-regular bipartite graph with N vertices on each side (i.e., $|L| = |R| = N$), with the property that for every subset $Y \subseteq R$ of size at least $(R + 7\varepsilon)N$, the number of vertices belonging to L that have at most $(R + 6\varepsilon)D$ of their neighbors in Y is at most δN (for $\delta = \varepsilon^2$). It is a well-known fact (used also in [54]) that if G is picked to be the double cover of a Ramanujan expander of degree $D \geqslant 4/(\delta\varepsilon^2)$, then G will have such a property.

We now define our final code $C^* = G(C_{\text{concat}}) \subseteq (\Sigma^D)^N$ formally. The codewords in C^* are in one-one correspondence with those of C_{concat}. Given a codeword $c \in C_{\text{concat}}$, its ND symbols (each belonging to Σ) are placed on the ND edges of G, with the D symbols in its i'th block (belonging to Σ^D, as defined above) being placed on the D edges incident

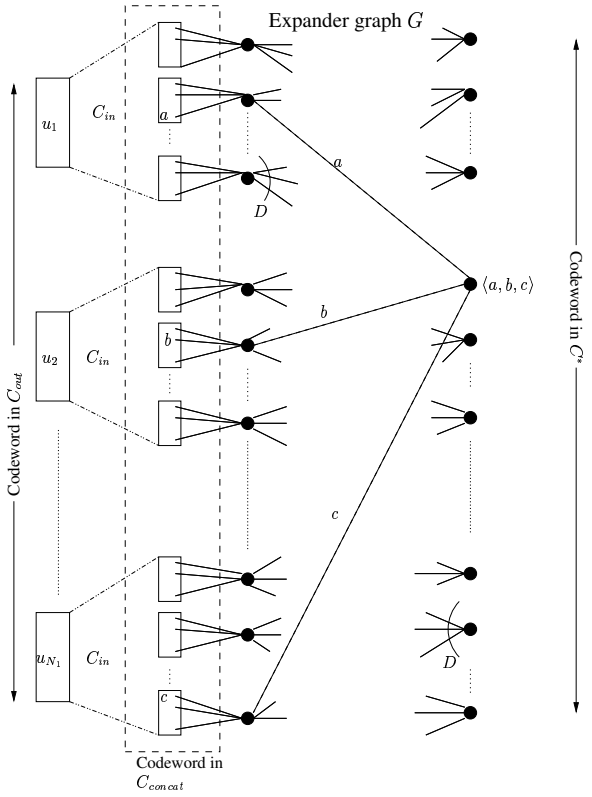


Figure 4.2: The code C^* used in the proof of Theorem 4.1. We start with a codeword $\langle u_1, \ldots, u_{N_1} \rangle$ in C_{out}. Then every symbol is encoded by C_{in} to form a codeword in C_{concat} (this intermediate codeword is marked by the dotted box). The symbols in the codeword for C_{concat} are divided into chunks of D symbols and then redistributed along the edges of an expander G of degree D. In the figure, we use $D = 3$ for clarity. Also the distribution of three symbols a, b and c (that form a symbol in the final codeword in C^*) is shown.

on the i'th vertex of L (in some fixed order). The codeword in C^* corresponding to c has as its i'th symbol the collection of D symbols (in some fixed order) on the D edges incident on the i'th vertex of R. See Figure 4.2 for a pictorial view of the construction.

Note that the rate of C^* is identical to that C_{concat}, and is thus at least R. Its alphabet size is $|\Sigma|^D = O(1/\varepsilon^2)^{O(1/\varepsilon^4)} = 2^{O(\varepsilon^{-4}\log(1/\varepsilon))}$, as claimed. We will now argue how C^* can be list decoded up to a fraction $(1 - R - 7\varepsilon)$ of errors.

Given a received word $\mathbf{r} \in (\Sigma^D)^N$, the following is the natural algorithm to find all codewords of C^* with agreement at least $(R+7\varepsilon)N$ with $\mathbf{r}$. Redistribute symbols according to the expander backwards to compute the received word $\mathbf{r}'$ for C_{concat} which would result in $\mathbf{r}$. Then run the earlier-mentioned decoding algorithm $\mathcal{A}$ on $\mathbf{r}'$.

We now briefly argue the correctness of this algorithm. Let $\mathbf{c} \in C^*$ be a codeword with

agreement at least $(R + 7\varepsilon)N$ with $\mathbf{r}$. Let $\mathbf{c}'$ denote the codeword of C_{concat} that leads to $\mathbf{c}$ after symbol redistribution by G, and finally suppose $\mathbf{c}''$ is the codeword of C_{out} that yields $\mathbf{c}'$ upon concatenation by C_{in}. By the expansion properties of G, it follows that all but a δ fraction of N D-long blocks of $\mathbf{r}'$ have agreement at least $(R+6\varepsilon)D$ with the corresponding blocks of $\mathbf{c}'$. By an averaging argument, this implies that at least a fraction $(1 - \sqrt{\delta})$ of the N_1 blocks of $\mathbf{c}'$ that correspond to codewords of C_{in} encoding the N_1 symbols of $\mathbf{c}''$, agree with at least a fraction $(1 - \sqrt{\delta})(R + 6\varepsilon) = (1 - \varepsilon)(R + 6\varepsilon) \geqslant R + 4\varepsilon$ of the symbols of the corresponding block of $\mathbf{r}'$. As argued earlier, this in turn implies that the decoding algorithm $\mathcal{A}$ for C_{concat} when run on input $\mathbf{r}'$ will output a polynomial size list that will include $\mathbf{c}'$. □

4.3 Binary Codes List Decodable up to the Zyablov Bound

Concatenating the folded Reed-Solomon codes with suitable inner codes also gives us polytime constructible binary codes that can be efficiently list decoded up to the Zyablov bound, i.e., up to twice the radius achieved by the standard GMD decoding of concatenated codes [41]. The optimal list recoverability of the folded Reed-Solomon codes plays a crucial role in establishing such a result.

Theorem 4.2. *For all $0 < R, r < 1$ and all $\varepsilon > 0$, there is a polynomial time constructible family of binary linear codes of rate at least $R \cdot r$ which can be list decoded in polynomial time up to a fraction $(1 - R)H^{-1}(1 - r) - \varepsilon$ of errors.*

Proof. Let $\gamma > 0$ be a small constant that will be fixed later. We will construct binary codes with the claimed property by concatenating two codes C_1 and C_2. For C_1, we will use a folded Reed-Solomon code over a field of characteristic 2 with block length n_1, rate at least R, and which can be $(1-R-\gamma, l)$-list recovered in polynomial time for $l = \lceil 10/\gamma \rceil$. Let the alphabet size of C_1 be 2^M where M is $O(\gamma^{-2}\log(1/\gamma)(1 - R)^{-1}\log n_1)$ (by Theorem 3.6, such a C_1 exists). For C_2, we will use a binary linear code of dimension M and rate at least r which is (ρ, l)-list decodable for $\rho = H^{-1}(1 - r - \gamma)$. Such a code is known to exist via a random coding argument that employs the semi-random method [51]. Also, a greedy construction of such a code by constructing its M basis elements in turn is presented in [51] and this process takes $2^{O(M)}$ time. We conclude that the necessary inner code can be constructed in $n_1^{O(\gamma^{-2}(1-R)^{-1}\log(1/\gamma))}$ time. The code C_1, being a folded Reed-Solomon code over a field of characteristic 2, is $\mathbb{F}_2$-linear, and therefore when concatenated with a binary linear inner code such as C_2, results in a binary linear code. The rate of the concatenated code is at least $R \cdot r$.

The decoding algorithm proceeds in a natural way. Given a received word, we break it up into blocks corresponding to the various inner encodings by C_1. Each of these blocks is list decoded up to a radius ρ, returning a set of at most l possible candidates for each outer codeword symbol. The outer code is then $(1 - R - \gamma, l)$-list recovered using these sets, each of which has size at most l, as input. To argue about the fraction of errors this

algorithm corrects, we note that the algorithm fails to recover a codeword only if on more than a fraction $(1-R-\gamma)$ of the inner blocks the codeword differs from the received word on more than a fraction ρ of symbols. It follows that the algorithm correctly list decodes up to a radius $(1-R-\gamma)\rho = (1-R-\gamma)H^{-1}(1-r-\gamma)$. If we pick an appropriate γ in $\Theta(\varepsilon^2)$, then by Lemma 2.4, $H^{-1}(1-r-\gamma) \geqslant H^{-1}(1-r) - \varepsilon/3$ (and $(1-R-\gamma) \geqslant 1-R-\varepsilon/3$), which implies $(1-R-\gamma)H^{-1}(1-r-\gamma) \geqslant (1-R)H^{-1}(1-r) - \varepsilon$ as desired. □

Optimizing over the choice of inner and outer codes rates r, R in the above results, we can decode up to the Zyablov bound, see Figure 4.1. For an analytic expression, see (4.2) with $s = 1$.

Remark 4.1. *In particular, decoding up to the Zyablov bound implies that we can correct a fraction $(1/2-\varepsilon)$ of errors with rate $\Omega(\varepsilon^3)$ for small $\varepsilon \to 0$, which is better than the rate of $\Omega(\varepsilon^3/\log(1/\varepsilon))$ achieved in [55]. However, our construction and decoding complexity are $n^{O(\varepsilon^{-2}\log(1/\varepsilon))}$ whereas these are at most $f(\varepsilon)n^c$ for an absolute constant c in [55]. Also, we bound the list size needed in the worst-case by $n^{O(\varepsilon^{-1}\log(1/\varepsilon))}$, while the list size needed in the construction in [55] is $(1/\varepsilon)^{O(\log\log(1/\varepsilon))}$.*

4.4 Unique Decoding of a Random Ensemble of Binary Codes

We will digress a bit to talk about a consequence of (the proof of) Theorem 4.2.

One of the biggest open questions in coding theory is to come up with explicit binary codes that are on the Gilbert Varshamov (or GV) bound. In particular, these are codes that achieve relative distance δ with rate $1 - H(\delta)$. There exist ensembles of binary codes for which if one picks a code at random then with high probability it lies on the GV bound. Coming up with an explicit construction of such a code, however, has turned out to be an elusive task.

Given the bleak state of affairs, some attention has been paid to the following problem. Give a probabilistic construction of binary codes that meet the GV bound (with high probability) together with efficient (encoding and) *decoding up to half the distance* of the code. Zyablov and Pinsker [110] give such a construction for binary codes of rate about 0.02 with *subexponential* time decoding algorithms. Guruswami and Indyk [53] give such a construction for binary linear codes up to rates about 10^{-4} with *polynomial* time encoding and decoding algorithms. Next we briefly argue that Theorem 4.2 can be used to extend the result of [53] to work till rates of about 0.02. In other words, we get the rate achieved by the construction of [110] but (like [53]) we get *polynomial* time encoding and decoding (up to half the GV bound).

We start with a brief overview of the construction of [53], which is based on code concatenation. The outer code is chosen to be the Reed-Solomon code (of say length N and rate R) while there are N linear binary inner codes of rate r (recall that in the "usual" code concatenation only one inner code is used) that are chosen uniformly (and independently) at random. A result of Thommesen [102] states that with high probability such a code

lies on the GV bound provided the rates of the codes satisfy $R \leqslant \alpha(r)/r$, where $\alpha(r) = 1 - H(1 - 2^{r-1})$. Guruswami and Indyk then give list-decoding algorithms for such codes such that for (overall) rate $rR \leqslant 10^{-4}$, the fraction of errors they can correct is at least $\frac{1}{2} \cdot H^{-1}(1 - rR)$ (that is, more than half the distance on the GV bound) as well as satisfy the constraint in Thommesen's result.

Given Theorem 4.2, here is the natural way to extend the result of [53]. We pick the outer code of rate R to be a folded Reed-Solomon code (with the list recoverable properties as required in Theorem 4.2) and the pick N independent binary linear codes of rate r as the inner codes. It is not hard to check that the proof of Thommesen also works when the outer code is folded Reed-Solomon. That is, the construction just mentioned lies on the GV bound with high probability. It is also easy to check that the proof of Theorem 4.2 also works when all the inner codes are different (basically the list decoding for the inner code in Theorem 4.2 is done by brute-force, which of course one can do even if all the N inner codes are different). Thus, if $rR \leqslant \alpha(r)$, we can list decode up to $(1 - R)H^{-1}(1 - r)$ fraction of errors and at the same time have the property that with high probability, the constructed code lies on the GV bound. Thus, all we now need to do is to check what is the maximum rate rR one can achieve while at the same time satisfying $rR \leqslant \alpha(r)$ and $(1-R)H^{-1}(1-r) \geqslant \frac{1}{2}H^{-1}(1-rR)$. This rate turns out to be around 0.02 (see Figure 4.3).

Thus, we have argued the following.

Theorem 4.3. *There is a probabilistic polynomial time procedure to construct codes whose rate vs. distance tradeoff meets the Gilbert-Varshamov bound with high probability for all rates up to* 0.02*. Furthermore, these codes can be decoded in polynomial time up to half the relative distance.*

One might hope that this method along with ideas of multilevel concatenated codes (about which we will talk next) can be used to push the overall rate significantly up from 0.02 that we achieve here. However, the following simple argument shows that one cannot go beyond a rate of 0.04. If we are targeting list decoding up to ρ fraction of errors (and use code concatenation), then the inner rate r must be at most $1 - H(\rho)$ (see for example (4.2)). Now by the Thommesen condition the overall rate is at most $\alpha(r)$. It is easy to check that $\alpha(\cdot)$ is an increasing function. Thus, the maximum overall rate that we can achieve is $\alpha(1 - H(\rho))$— this is the curve titled "Limit of the method" in Figure 4.3. One can see from Figure 4.3, the maximum rate for which this curve still beats half the GV bound is at most 0.04.

4.5 List Decoding up to the Blokh Zyablov Bound

We now present linear codes over any fixed alphabet that can be constructed in polynomial time and can be efficiently list decoded up to the so called Blokh-Zyablov bound (Figure 4.1). This achieves a sizable improvement over the tradeoff achieved by codes from Section 4.3 (see Figure 4.1 and Table 4.1).

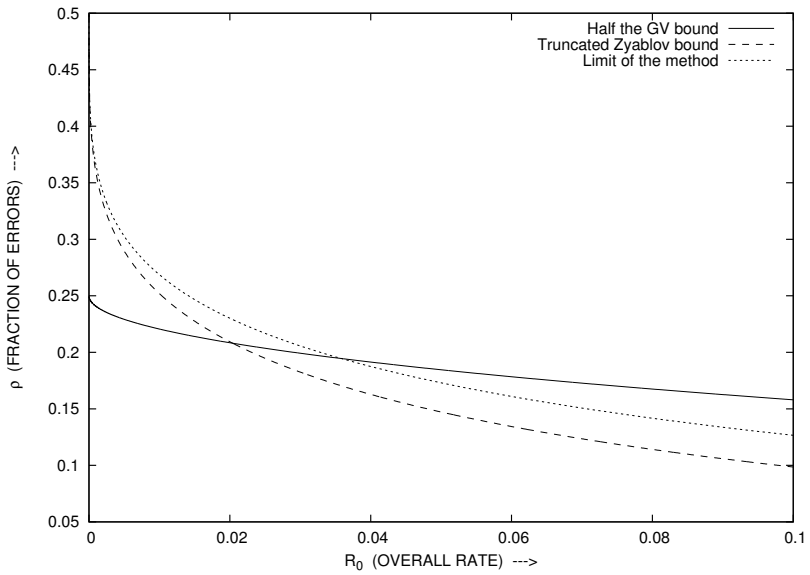


Figure 4.3: Tradeoffs for decoding certain random ensemble of concatenated codes. "Half the GV bound" is the curve $\frac{1}{2} \cdot H^{-1}(1 - R_0)$ while "Truncated Zyablov bound" is the limit till which we can list decode the concatenated codes (and still satisfy the Thommesen condition, that is for inner and outer rates r and R, $rR = R_0 \leqslant \alpha(r)$). "Limit of the method" is the best tradeoff one can hope for using list decoding of code concatenation along with the Thommesen result.

Our codes are constructed via multilevel concatenated codes. We will provide a formal definition later on — we just sketch the basic idea here. For an integer $s \geqslant 1$, a multilevel concatenated code C over $\mathbb{F}_q$ is obtained by combining s "outer" codes $C^0_{out}, C^1_{out}, \ldots, C^{s-1}_{out}$ of the same block length , say N, over large alphabets of size say $q^{a_0}, q^{a_1}, \ldots, q^{a_{s-1}}$, respectively, with a suitable "inner" code. The inner code is of dimension $a_0 + a_1 \cdots + a_{s-1}$. Given messages $m^0, m^1, \ldots, m^{s-1}$ for the s outer codes, the encoding as per the multilevel generalized concatenation codes proceeds by first encoding each m^j as per C^j_{out}. Then for every $1 \leqslant i \leqslant N$, the collection of the ith symbols of $C^j_{out}(m^j)$ for $0 \leqslant j \leqslant s-1$, which can be viewed as a string over $\mathbb{F}_q$ of length $a_0 + a_1 + \cdots + a_{s-1}$, is encoded by the inner code. For $s = 1$ this reduces to the usual definition of code concatenation.

We present a list-decoding algorithm for C, given list-recovery algorithms for the outer codes and list-decoding algorithms for the inner code and some of its subcodes. What makes this part more interesting than the usual code concatenation (like those in Section 4.3), is that the inner code in addition to having good list-decodable properties, also needs to have good list-decodable properties for certain subcodes. Specifically, the subcodes of dimension $a_j + a_{j+1} + \cdots + a_{s-1}$ of the inner code obtained by arbitrarily fixing the first

$a_0 + \cdots + a_{j-1}$ symbols of the message, must have better list-decodability properties for increasing j (which is intuitively possible since they have lower rate). In turn, this allows the outer codes C_{out}^j to have rates increasing with j, leading to an overall improvement in the rate for a certain list-decoding radius.

To make effective use of the above approach, we also prove, via an application of the probabilistic method, that a random linear code over $\mathbb{F}_q$ has the required stronger condition on list decodability. By applying the method of conditional expectation ([2]), we can construct such a code deterministically in time singly exponential in the block length of the code (which is polynomial if the inner code encodes messages of length $O(\log N)$). Note that constructing such an inner code, given the existence of such codes, is easy in quasi-polynomial time by trying all possible generator matrices. The lower time complexity is essential for constructing the final code C in polynomial time.

4.5.1 Multilevel Concatenated Codes

We will be working with multilevel concatenation coding schemes [33]. We start this section with the definition of multilevel concatenated codes. As the name suggests, these are generalizations of concatenated codes. Recall that for a concatenated code, we start with a code C_{out} over a large alphabet (called the outer code). Then we need a code C_{in} that maps all symbols of the larger alphabet to strings over a smaller alphabet (called the inner code). The encoding for the concatenated code (denoted by $C_{out} \circ C_{in}$) is done as follows. We think of the message as being a string over the large alphabet and then encode it using C_{out}. Now we use C_{in} to encode each of the symbols in the codeword of C_{out} to get our codeword (in $C_{out} \circ C_{in}$) over the smaller alphabet.

Multilevel code concatenation generalizes the usual code concatenation in the following manner. Instead of there being one outer code, there are multiple outer codes. In particular, we "stack" codewords from these multiple outer codes and construct a matrix. The inner codes then act on the columns of these intermediate matrix. We now formally define multilevel concatenated codes.

There are $s \geqslant 1$ outer codes, denoted by $C_{out}^0, C_{out}^1, \ldots, C_{out}^{s-1}$. For every $0 \leqslant i \leqslant s-1$, C_{out}^i is a code of block length N and rate R_i and defined over a field $\mathbb{F}_{Q_i}$. The inner code C_{in} is code of block length n and rate r that maps tuples from $\mathbb{F}_{Q_0} \times \mathbb{F}_{Q_1} \times \cdots \times \mathbb{F}_{Q_{s-1}}$ to symbols in $\mathbb{F}_q$. In other words,

$$C_{out}^i : (\mathbb{F}_{Q_i})^{R_i N} \to (\mathbb{F}_{Q_i})^N,$$

$$C_{in} : \mathbb{F}_{Q_0} \times \mathbb{F}_{Q_1} \times \cdots \times \mathbb{F}_{Q_{s-1}} \to (\mathbb{F}_q)^n.$$

The multilevel concatenated code, denoted by $(C_{out}^0 \times C_{out}^1 \times \ldots C_{out}^{s-1}) \circ C_{in}$ is a map of the following form:

$$(C_{out}^0 \times C_{out}^1 \times \ldots C_{out}^{s-1}) \circ C_{in} : (\mathbb{F}_{Q_0})^{R_0 N} \times (\mathbb{F}_{Q_1})^{R_1 N} \times \cdots \times (\mathbb{F}_{Q_{S-1}})^{R_{s-1} N} \to (\mathbb{F}_q)^{nN}.$$

We now describe the encoding scheme. Given a message $(m^0, m^1, \ldots, m^{s-1}) \in (\mathbb{F}_{Q_0})^{R_0 N} \times (\mathbb{F}_{Q_1})^{R_1 N} \times \cdots \times (\mathbb{F}_{Q_{S-1}})^{R_{s-1} N}$, we first construct an $s \times N$ matrix M, whose i^{th} row is the codeword $C^i_{out}(m^i)$. Note that every column of M is an element from the set $\mathbb{F}_{Q_0} \times \mathbb{F}_{Q_1} \times \cdots \times \mathbb{F}_{Q_{s-1}}$. Let the j^{th} column (for $1 \leqslant j \leqslant N$) be denoted by M_j. The codeword corresponding to the multilevel concatenated code ($C \stackrel{def}{=} (C^0_{out} \times C^1_{out} \times \ldots C^{s-1}_{out}) \circ C_{in}$) is defined as follows

$$C(m^0, m^1, \ldots, m^{s-1}) = (C_{in}(M_1), C_{in}(M_2), \cdots, C_{in}(M_N)) .$$

(The codeword can be naturally be thought of as an $n \times N$ matrix, whose i'th column corresponds to the inner codeword encoding the i'th symbols of the s outer codewords.)

For the rest of the chapter, we will only consider outer codes over the same alphabet, that is, $Q_0 = Q_1 = \cdots = Q_{s-1} = Q$. Further, $Q = q^a$ for some integer $a \geqslant 1$. Note that if $C^0_{out}, \ldots, C^{s-1}_{out}$ and C_{in} are all $\mathbb{F}_q$ linear, then so is $(C^0_{out} \times C^1_{out} \times \cdots \times C^{s-1}_{out}) \circ C_{in}$.

The gain from using multilevel concatenated codes comes from looking at the inner code C_{in} along with its subcodes. For the rest of the section, we will consider the case when C_{in} is linear (though the ideas can easily be generalized for general codes). Let $G \in \mathbb{F}_q^{as \times n}$ be the generator matrix for C_{in}. Let $r_0 = as/n$ denote the rate of C_{in}. For $0 \leqslant j \leqslant s-1$, define $r_j = r_0(1 - j/s)$, and let G_j denote $r_j n \times n$ submatrix of G containing the last $r_j n$ rows of G. Denote the code generated by G_j by C^j_{in}; the rate of C^j_{in} is r_j. For our purposes we will actually look at the subcode of C_{in} where one fixes the first $0 \leqslant j \leqslant s-1$ message symbols. Note that for every j these are just cosets of C^j_{in}. We will be looking at C_{in}, which in addition to having good list decoding properties as a "whole," also has good list-decoding properties for each of its subcode C^j_{in}.

The multilevel concatenated code C ($= (C^0_{out} \times \cdots \times C^{s-1}_{out}) \circ C_{in}$) has rate $R(C)$ that satisfies

$$R(C) = \frac{r_0}{s} \sum_{i=0}^{s-1} R_i \, . \tag{4.1}$$

The Blokh-Zyablov bound is the trade-off between rate and relative distance obtained when the outer codes meet the Singleton bound (i.e., C^j_{out} has relative distance $1 - R_j$), and the various subcodes C^j_{in} of the inner code, including the whole inner code $C_{in} = C^0_{in}$, lie on the Gilbert-Varshamov bound (i.e., have relative distance $\delta_j \geqslant H_q^{-1}(1 - r_j)$). The multilevel concatenated code then has relative distance at least $\min_{0 \leqslant j \leqslant s-1}(1 - R_j) H_q^{-1}(1 - r_j)$. Expressing the rate in terms of distance, the Blokh-Zyablov bound says that there exist multilevel concatenated code C with relative distance at least δ with the following rate:

$$R^s_{BZ}(C) = \max_{0 < r < 1 - H_q(\delta)} r - \frac{r}{s} \sum_{i=0}^{s-1} \frac{\delta}{H_q^{-1}(1 - r + ri/s)} . \tag{4.2}$$

As s increases, the trade-off approaches the integral

$$R_{BZ}(C) = 1 - H_q(\delta) - \delta \int_0^{1 - H_q(\delta)} \frac{dx}{H_q^{-1}(1 - x)} \, . \tag{4.3}$$

The convergence of $R^s_{BZ}(C)$ to $R_{BZ}(C)$ happens quite quickly even for small s such as $s = 10$.

Nested List Decoding

We will need to work with the following generalization of list decoding. The definition looks more complicated than it really is.

Definition 4.1 (Nested linear list decodable code). *Given a linear code C in terms of some generator matrix $G \in \mathbb{F}_q^{k \times n}$, an integer s that divides k, a vector $\mathbf{L} = \langle L_0, L_1, \ldots, L_{s-1} \rangle$ of integers L_j ($0 \leqslant j \leqslant s-1$), a vector $\rho = \langle \rho_0, \rho_1 \ldots, \rho_{s-1} \rangle$ with $0 < \rho_j < 1$, and a vector $\mathbf{r} = \langle r_0, \ldots, r_{s-1} \rangle$ of reals where $r_0 = k/n$ and $0 \leqslant r_{s-1} < \cdots < r_i < r_0$, C is called an $(\mathbf{r}, \rho, \mathbf{L})$-nested list decodable if the following holds:*

For every $0 \leqslant j \leqslant s-1$, C^j is a rate r_j code that is (ρ_j, L_j)-list decodable, where C^j is the subcode of C generated by the the last $r_j n$ rows of the generator matrix G.

4.5.2 Linear Codes with Good Nested List Decodability

In this section, we will prove the following result concerning the existence (and constructibility) of linear codes over any fixed alphabet with good nested list-decodable properties.

Theorem 4.4. *For any integer $s \geqslant 1$ and reals $0 < r_{s-1} < r_{s-2} < \cdots < r_1 < r_0 < 1$, $\varepsilon > 0$, let $\rho_j = H_q^{-1}(1 - r_j - 2\varepsilon)$ for every $0 \leqslant j \leqslant s-1$. Let $\mathbf{r} = \langle r_0, \ldots, r_{s-1} \rangle$, $\rho = \langle \rho_0, \rho_1, \ldots, \rho_{s-1} \rangle$ and $\mathbf{L} = \langle L_0, L_1, \ldots, L_{s-1} \rangle$, where $L_j = q^{1/\varepsilon}$. For large enough n, there exists a linear code (over fixed alphabet $\mathbb{F}_q$) that is $(\mathbf{r}, \rho, \mathbf{L})$-nested list decodable. Further, such a code can be constructed in time $q^{O(n/\varepsilon)}$.*

Proof. We will show the existence of the required codes via a simple use of the probabilistic method (in fact, we will show that a random linear code has the required properties with high probability). We will then use the method of conditional expectation ([2]) to derandomize the construction with the claimed time complexity.

Define $k_j = \lfloor r_i n \rfloor$ for every $0 \leqslant j \leqslant s-1$. We will pick a random $k_0 \times n$ matrix G with entries picked independently from $\mathbb{F}_q$. We will show that the linear code C generated by G has good nested list decodable properties with high probability. Let C_j, for $0 \leqslant j \leqslant s-1$ be the code generated by the "bottom" k_j rows of G. Recall that we have to show that with high probability C_j is $(\rho_j, q^{1/\varepsilon})$ list decodable for every $0 \leqslant j \leqslant s-1$ (C_j obviously has rate r_j). Finally for integers $J, k \geqslant 1$, and a prime power q, let $\mathrm{Ind}(q, k, J)$ denote the collection of subsets $\{x^1, x^2, \ldots, x^J\} \subseteq \mathbb{F}_q^k$ such that all vectors $x^1, \ldots, x^J$ are linearly independent over $\mathbb{F}_q$.

We recollect the following two straightforward facts: (i) Given any L distinct vectors from $\mathbb{F}_q^k$, for some $k \geqslant 1$, at least $\lceil \log_q L \rceil$ of them are linearly independent; (ii) Any set of linearly independent vectors in $\mathbb{F}_q^k$ are mapped to independent random vectors in $\mathbb{F}_q^n$ by

a random $k \times n$ matrix over $\mathbb{F}_q$. The first claim is obvious. For the second claim, first note that for any $\mathbf{v} \in \mathbb{F}_q^k$ and a random $k \times n$ matrix $\mathbf{G}$ (where each of the kn values are chosen uniformly and independently at random from $\mathbb{F}_q$) the values at the n different positions in $\mathbf{v} \cdot \mathbf{G}$ are independent. Further, the value at position $1 \leqslant i \leqslant n$, is given by $\mathbf{v} \cdot \mathbf{G}_i$, where $\mathbf{G}_i$ is the i^{th} column of $\mathbf{G}$. Now for fixed $\mathbf{v}$, $\mathbf{v} \cdot \mathbf{G}_i$ takes values from $\mathbb{F}_q$ uniformly at random (note that $\mathbf{G}_i$ is a random vector from $\mathbb{F}_q^k$). Finally, for linearly independent vectors $\mathbf{v}^1, \ldots, \mathbf{v}^m$ by a suitable linear invertible map can be mapped to the standard basis vectors $\mathbf{e}_1, \ldots, \mathbf{e}_m$. Obviously, the values $\mathbf{e}_1 \cdot \mathbf{G}_i \ \ldots, \mathbf{e}_m \cdot \mathbf{G}_i$ are independent.

We now move on to the proof of existence of linear codes with good nested list decodability. We will actually do the proof in a manner that will facilitate the derandomization of the proof. Define $J = \lceil \log_q(q^{1/\varepsilon} + 1) \rceil$. For any vector $\mathbf{y} \in \mathbb{F}_q^n$, integer $0 \leqslant j \leqslant s-1$, subset $T = \{x^1, \ldots, x^J\} \in \mathrm{Ind}(q, k_j, J)$ and any collection $\mathcal{S}$ of subsets $S_1, S_2, \ldots, S_J \subseteq \{1, \ldots, n\}$ of size at most $\rho_j n$, define an indicator variable $I(j, \mathbf{y}, T, \mathcal{S})$ in the following manner. $I(j, \mathbf{y}, T, \mathcal{S}) = 1$ if and only if for every $1 \leqslant i \leqslant J$, $C(x^i)$ differs from $\mathbf{y}$ in exactly the set S_i. Note that if for some $0 \leqslant j \leqslant s-1$, there are $q^{1/\varepsilon} + 1$ codewords in C_j all of which differ from some received word $\mathbf{y}$ in at most $\rho_j n$ places, then this set of codewords is a "counter-example" that shows that C is not $(\mathbf{y}, \rho, \mathbf{L})$-nested list decodable. Since the $q^{1/\varepsilon} + 1$ codewords will have some set T of J linearly independent codewords, the counter example will imply that $I(j, \mathbf{y}, T, \mathcal{S}) = 1$ for some collection of subsets $\mathcal{S}$. In other words, the indicator variable captures the set of bad events we would like to avoid. Finally define the sum of all the indicator variables as follows:

$$S_C = \sum_{j=0}^{s-1} \sum_{\mathbf{y} \in \mathbb{F}_q^n} \sum_{T \in \mathrm{Ind}(q,k_j,J)} \sum_{\substack{\mathcal{S} = \{S_1, \ldots, S_J\}, \\ S_i \subseteq \{1, \ldots, n\}, |S_i| \leqslant \rho_j n}} I(j, \mathbf{y}, T, \mathcal{S}).$$

Note that if $S_C = 0$, then C is $(\mathbf{y}, \rho, \mathbf{L})$-nested list decodable as required. Thus, we can prove the existence of such a C if we can show that $\mathbb{E}_{\mathbf{C}}[S_C] < 1$. By linearity of expectation, we have

$$\mathbb{E}[S_C] = \sum_{j=0}^{s-1} \sum_{\mathbf{y} \in \mathbb{F}_q^n} \sum_{T \in \mathrm{Ind}(q,k_j,J)} \sum_{\substack{\mathcal{S} = \{S_1, \ldots, S_J\}, \\ S_i \subseteq \{1, \ldots, n\}, |S_i| \leqslant \rho_j n}} \mathbb{E}[I(j, \mathbf{y}, T, \mathcal{S})]. \tag{4.4}$$

Fix some arbitrary $j, \mathbf{y}, T = \{x^1, x^2, \ldots, x^J\}, \mathcal{S} = \{S_1, S_2, \ldots, S_J\}$ (in their corresponding domains). Then we have

$$\begin{aligned}
\mathbb{E}[I(j, \mathbf{y}, T, \mathcal{S})] &= \Pr[I(j, \mathbf{y}, T, \mathcal{S}) = 1] \\
&= \prod_{x^i \in T} \Pr[C(x^i) \text{ differ from } \mathbf{y} \text{ in exactly the positions in } S_i] \\
&= \prod_{i=1}^{J} \left(\frac{q-1}{q}\right)^{|S_i|} \left(\frac{1}{q}\right)^{n-|S_i|}
\end{aligned} \tag{4.5}$$

$$= \prod_{i=1}^{J} \frac{(q-1)^{|S_i|}}{q^n} \,, \tag{4.6}$$

where the second and the third equality follow from the definition of the indicator variable, the fact that vectors in T are linearly independent and the fact that a random matrix maps linearly independent vectors to independent uniformly random vectors in $\mathbb{F}_q^n$. Using (4.6) in (4.4), we get

$$\begin{aligned}
\mathbb{E}[S_C] &= \sum_{j=0}^{s-1} \sum_{\mathbf{y}\in\mathbb{F}_q^n} \sum_{T\in \mathrm{Ind}(q,k_j,J)} \sum_{\substack{\mathcal{S}=\{S_1,\ldots,S_J\},\\ S_i\subseteq\{1,\ldots,n\},|S_i|\leqslant\rho_j n}} \prod_{i=1}^{J} \frac{(q-1)^{|S_i|}}{q^n} \\
&= \sum_{j=0}^{s-1} \sum_{\mathbf{y}\in\mathbb{F}_q^n} \sum_{T\in \mathrm{Ind}(q,k_j,J)} \sum_{(\ell_1,\ell_2,\ldots,\ell_J)\in\{0,1,\ldots,\rho_j n\}^J} \prod_{i=1}^{J} \binom{n}{\ell_i} \frac{(q-1)^{\ell_i}}{q^n} \\
&= \sum_{j=0}^{s-1} \sum_{\mathbf{y}\in\mathbb{F}_q^n} \sum_{T\in \mathrm{Ind}(q,k_j,J)} \left(\sum_{\ell=0}^{\rho_j n} \binom{n}{\ell} \frac{(q-1)^{\ell}}{q^n} \right)^J \\
&\leqslant \sum_{j=0}^{s-1} \sum_{\mathbf{y}\in\mathbb{F}_q^n} \sum_{T\in \mathrm{Ind}(q,k_j,J)} q^{nJ(H_q(\rho_j)-1)} \\
&\leqslant \sum_{j=0}^{s-1} q^n \cdot q^{Jk_j} \cdot q^{nJ(H_q(\rho_j)-1)} \\
&\leqslant \sum_{j=0}^{s-1} q^{nJ(1/J+r_j+1-r_j-2\varepsilon-1)} \\
&\leqslant sq^{-\varepsilon nJ} \,.
\end{aligned} \tag{4.7}$$

The first inequality follows from Proposition 2.1. The second inequality follows by upper bounding the number of J linearly independent vectors in $\mathbb{F}_q^{k_j}$ by q^{Jk_j}. The third inequality follows from the fact that $k_j = \lfloor r_j n \rfloor$ and $\rho_j = H_q^{-1}(1 - r_j - 2\varepsilon)$, The final inequality follows from the fact that $J = \lceil \log_q(q^{1/\varepsilon} + 1) \rceil$.

Thus, (4.7) shows that there exists a code C (in fact with high probability) that is $(\mathbf{y}, \rho, \mathbf{L})$-nested list decodable. In fact, this could have been proved using a simpler argument. However, the advantage of the argument above is that we can now apply the method of conditional expectations to derandomize the above proof.

The algorithm to deterministically generate a linear code C that is $(\mathbf{y}, \rho, \mathbf{L})$-nested list decodable is as follows. The algorithm consists of n steps. At any step $1 \leqslant i \leqslant n$, we choose the i^{th} column of the generator matrix to be the value $\mathbf{v}^i \in \mathbb{F}_q^{k_0}$ that minimizes the conditional expectation $\mathbb{E}[S_C | \mathbf{G}_1 = \mathbf{v}^1, \ldots, \mathbf{G}_{i-1} = \mathbf{v}^{i-1}, \mathbf{G}_i = \mathbf{v}^i]$, where $\mathbf{G}_i$ denotes the i^{th} column of $\mathbf{G}$ and $\mathbf{v}^1, \ldots, \mathbf{v}^{i-1}$ are the column vectors chosen in the previous $i-1$

steps. This algorithm would work if for any $1 \leqslant i \leqslant n$ and vectors $\mathbf{v}^1, \ldots, \mathbf{v}^i$, we can exactly compute $\mathbb{E}[S_C | \mathbf{G}_1 = \mathbf{v}^1, \ldots, \mathbf{G}_i = \mathbf{v}^i]$. Indeed from (4.4), we have $\mathbb{E}[S_C | \mathbf{G}_1 = \mathbf{v}^1, \ldots, \mathbf{G}_i = \mathbf{v}^i]$ is

$$\sum_{j=0}^{s-1} \sum_{\mathbf{y} \in \mathbb{F}_q^n} \sum_{T \in \mathrm{Ind}(q,k_j,J)} \sum_{\substack{\mathcal{S}=\{S_1,\ldots,S_J\}, \\ S_i \subseteq \{1,\ldots,n\}, |S_i| \leqslant \rho_j n}} \mathbb{E}[I(j,\mathbf{y},T,\mathcal{S}) | \mathbf{G}_1 = \mathbf{v}^1, \ldots, \mathbf{G}_i = \mathbf{v}^i].$$

Thus, we would be done if we can compute the following for every value of $j, \mathbf{y}, T = \{x^1, \ldots, x^J\}, \mathcal{S} = \{S_1, \ldots, S_J\}$: $\mathbb{E}[I(j,\mathbf{y},T,\mathcal{S}) = 1 | \mathbf{G}_1 = \mathbf{v}^1, \ldots, \mathbf{G}_i = \mathbf{v}^i]$. Note that fixing the first i columns of G implies fixing the value of the codewords in the first i positions. Thus, the indicator variable is 0 (or in other words, the conditional expectation we need to compute is 0) if for some message, the corresponding codeword does not disagree with $\mathbf{y}$ exactly as dictated by $\mathcal{S}$ in the first i positions. More formally, $I(j,\mathbf{y},T,\mathcal{S}) = 0$ if the following is true for some $1 \leqslant \ell \leqslant i$ and $0 \leqslant i' \leqslant J$: $x^{i'} \cdot \mathbf{G}_\ell \neq \mathbf{y}_\ell$, if $\ell \notin S_{i'}$ and $x^{i'} \cdot \mathbf{G}_\ell = \mathbf{y}_\ell$ otherwise. However, if none of these conditions hold, then using argument similar to the ones used to obtain (4.6), one can show that

$$\mathbb{E}[I(j,\mathbf{y},T,\mathcal{S}) | \mathbf{G}_1 = \mathbf{v}^1, \ldots, \mathbf{G}_i = \mathbf{v}^i] = \prod_{\ell=1}^{J} \left(\frac{q-1}{q}\right)^{|S'_\ell|} \left(\frac{1}{q}\right)^{n-i-|S'_\ell|},$$

where $S'_\ell = S_\ell \setminus \{1, 2, \ldots, i\}$ for every $1 \leqslant \ell \leqslant J$.

To complete the proof, we need to estimate the time complexity of the above algorithm. There are n steps and at every step i, the algorithm has to consider $q^{k_0} \leqslant q^n$ different choices of $\mathbf{v}^i$. For every choice of $\mathbf{v}^i$, the algorithm has to compute the conditional expectation of the indicator variables for all possible values of $j, \mathbf{y}, T, \mathcal{S}$. It is easy to check that there are $\sum_{i=1}^{s} q^n \cdot q^{Jk_j} \cdot 2^{nJ} \leqslant sq^{n(1+2J)}$ possibilities. Finally, the computation of the conditional expected value of a fixed indicator variable takes time $O(snJ)$. Thus, in all the total time taken is $O(n \cdot q^n \cdot sq^{n(1+2J)} \cdot snJ) = q^{O(n/\varepsilon)}$, as required. □

4.5.3 List Decoding Multilevel Concatenated Codes

In this section, we will see how one can list decode multilevel concatenated codes, provided the outer codes have good list recoverability and the inner code has good nested list decodability. We have the following result, which generalizes Theorem 4.2 for regular concatenated codes (the case $s = 1$).

Theorem 4.5. *Let $s \geqslant 1$ and $\ell \geqslant 1$ be integers. Let $0 < R_0 < R_1 < \cdots < R_{s-1} < 1$, $0 < r_0 < 1$ be rationals and $0 < \xi_0, \cdots, \xi_{s-1} < 1$, $0 < \rho_0, \cdots, \rho_{s-1} < 1$ and $\varepsilon > 0$ be reals. Let q be a prime power and let $Q = q^a$ for some integer $a > 1$. Further, let C_{out}^j ($0 \leqslant j \leqslant s-1$) be an $\mathbb{F}_q$-linear code over $\mathbb{F}_Q$ of rate R_j and block length N that is (ξ_j, ℓ, L)-list recoverable. Finally, let C_{in} be a linear $(\mathbf{r}, \rho, \mathbf{L})$-nested list decodable code over $\mathbb{F}_q$ of*

rate r_0 and block length $n = as/r_0$, where $\mathbf{r} = \langle r_0, \cdots, r_{s-1} \rangle$ with $r_i = (1 - i/s)r_0$, $\rho = \langle \rho_0, \cdots, \rho_{s-1} \rangle$ and $\mathbf{L} = \langle \ell, \ell, \cdots, \ell \rangle$. Then $C = (C_{out}^0 \times \cdots \times C_{out}^{s-1}) \circ C_{in}$ is a linear $(\min_j \xi_j \cdot \rho_j, L^s)$-list decodable code. Further, if the outer code C_{out}^j can be list recovered in time $T_j(N)$ and the inner code C_{in} can be list decoded in time $t_j(n)$ (for the j^{th} level), then C can be list decoded in time $O\left(\sum_{j=0}^{s-1} L^j \left(T_j(N) + N \cdot t_j(n)\right)\right)$.

Proof. Given list-recovery algorithms for C_{out}^j and list-decoding algorithms for C_{in} (and its subcodes C_{in}^j), we will design a list-decoding algorithm for C. Recall that the received word is an $n \times N$ matrix over $\mathbb{F}_q$. Each consecutive "chunk" of n/s rows should be decoded to a codeword in C_{out}^j. The details follow.

Before we describe the algorithm, we will need to fix some notation. Define $\delta = \min_j \xi_j \rho_j$. Let $\mathbf{Y} \in \mathbb{F}_q^{nN}$ be the received word, which we will think of as an $n \times N$ matrix over $\mathbb{F}_q$ (note that s divides n). For any $n \times N$ matrix M and for any $1 \leqslant i \leqslant N$, let $M_i \in \mathbb{F}_q^n$ denote the i^{th} column of the matrix M. Finally, for every $0 \leqslant j \leqslant s-1$, let C_{in}^j denote the subcode of C_{in} generated by all but the first ja rows of the generator matrix of C_{in}. We are now ready to describe our algorithm.

Recall that the algorithm needs to output all codewords in C that differ from $\mathbf{Y}$ in at most δ fraction of positions. For the ease of exposition, we will consider an algorithm that outputs matrices from $C_{out}^0 \times \cdots \times C_{out}^{s-1}$. The algorithm has s phases. At the end of phase j ($0 \leqslant j \leqslant s-1$), the algorithm will have a list of matrices (called $\mathcal{L}_j$) from $C_{out}^0 \times \cdots \times C_{out}^j$, where each matrix in $\mathcal{L}_j$ is a possible submatrix of some matrix that will be in the final list output by the algorithm. The following steps are performed in phase j (where we are assuming that the list-decoding algorithm for C_{in}^j returns a list of messages while the list-recovery algorithm for C_{out}^j returns a list of codewords).

1. Set $\mathcal{L}_j$ to be the empty set.

2. For every $\mathbf{c} = (c^0, \cdots, c^{j-1}) \in \mathcal{L}_{j-1}$ repeat the following steps (if this is the first phase, that is $j = 0$, then repeat the following steps once):

   (a) Let G_j be the first aj rows of the generator matrix of C_{in}. Let $\mathbf{X} = (G_j)^T \cdot \mathbf{c}$, where we think of $\mathbf{c}$ as an $ja \times N$ matrix over $\mathbb{F}_q$. Let $\mathbf{Z} = \mathbf{Y} - \mathbf{X}$ (for $j = 0$ we use the convention that $\mathbf{X}$ is the all 0s matrix). For every $1 \leqslant i \leqslant N$, use the list-decoding algorithm for C_{in}^j on column $\mathbf{Z}_i$ for up to ρ_j fraction of errors to obtain list $S_i^j \subseteq (\mathbb{F}_Q)^{s-j}$. Let $T_i^j \subseteq \mathbb{F}_Q$ be the projection of every vector in S_i^j on to its first component.

   (b) Run the list-recovery algorithm for C_{out}^j on set of lists $\{T_i^j\}_i$ obtained from the previous step for up to ξ_j fraction of errors. Store the set of codewords returned in I_j.

   (c) Add $\{(\mathbf{c}, \mathbf{v}) | \mathbf{v} \in I_j\}$ to $\mathcal{L}_j$.

At the end, remove all the matrices $M \in \mathcal{L}_{s-1}$, for which the codeword $(C_{in}(M_1), C_{in}(M_2), \cdots, C_{in}(M_N))$ is at a distance more than δ from $\mathbf{Y}$. Output the remaining matrices as the final answer.

We will first talk about the running time complexity of the algorithm. It is easy to check that each repetition of steps 2(a)-(c) takes time $O(T_j(N) + N \cdot t_j(n))$. To compute the final running time, we need to get a bound on number of times step 2 is repeated in phase j. It is easy to check that the number of repetitions is exactly $|\mathcal{L}_{j-1}|$. Thus, we need to bound $|\mathcal{L}_{j-1}|$. By the list recoverability property of C^j_{out}, we can bound $|I_j|$ by L. This implies that $|\mathcal{L}_j| \leqslant L|\mathcal{L}_{j-1}|$, and therefore by induction we have

$$|\mathcal{L}_i| \leqslant L^{i+1} \quad \text{for } i = 0, 1, \ldots, s-1 \; . \tag{4.8}$$

Thus, the overall running time and the size of the list output by the algorithm are as claimed in the statement of the theorem.

We now argue the correctness of the algorithm. That is, we have to show that for every $M \in C^0_{out} \times \cdots \times C^{s-1}_{out}$, such that $(C_{in}(M_1), C_{in}(M_2) \cdots, C_{in}(M_N))$ is at a distance at most δ from $\mathbf{Y}$ (call such an M a *good* matrix), $M \in \mathcal{L}_{s-1}$. In fact, we will prove a stronger claim: for every good matrix M and every $0 \leqslant j \leqslant s-1$, $M^j \in \mathcal{L}_j$, where M^j denotes the submatrix of M that lies in $C^0_{out} \times \cdots \times C^j_{out}$ (that is the first j "rows" of M). For the rest of the argument fix an arbitrary good matrix M. Now assume that the stronger claim above holds for $j' - 1$ ($< s-1$). In other words, $M^{j'-1} \in \mathcal{L}_{j'-1}$. Now, we need to show that $M^{j'} \in \mathcal{L}_{j'}$.

For concreteness, let $M = (m^0, \cdots, m^{s-1})^T$. As M is a good matrix and $\delta \leqslant \xi_{j'}\rho_{j'}$, $C_{in}(M_i)$ can disagree with $\mathbf{Y}_i$ on at least a fraction $\rho_{j'}$ of positions for at most $\xi_{j'}$ fraction of column indices i. The next crucial observation is that for any column index i, $C_{in}(M_i) = (G_{j'})^T \cdot (m^0_i, \cdots, m^{j'-1}_i) + (G \setminus G_{j'})^T \cdot (m^{j'}_i, \cdots, m^{s-1}_i)$, where $G_{j'}$ is as defined in step 2(a), $G \setminus G_{j'}$ is the submatrix of G obtained by "removing" $G_{j'}$ and $m^{j'}_i$ is the i^{th} component of the vector $m^{j'}$. Figure 4.5.3 might help the reader to visualize the different variables. Note that $G \setminus G_{j'}$ is the generator matrix of $C^{j'}_{in}$. Thus, for at most $\xi_{j'}$ fraction of column indices i, $(m^{j'}_i, \cdots, m^{s-1}_i) \cdot (G \setminus G_{j'})$ disagrees with $\mathbf{Y}_i - \mathbf{X}_i$ on at least $\rho_{j'}$ fraction of places, where $\mathbf{X}$ is as defined in Step 2(a), and $\mathbf{X}_i$ denotes the i'th column of $\mathbf{X}$. As $C^{j'}_{in}$ is $(\rho_{j'}, \ell)$-list decodable, for at least $1 - \xi_{j'}$ fraction of column index i, $M^{j'}_i$ will be in $S^{j'}_i$ (where $M^{j'}_i$ is M_i projected on it's last $s - j'$ co-ordinates and $S^{j'}_i$ is as defined in Step 2(a)). In other words, $m^{j'}_i$ is in $T^{j'}_i$ for at least $1 - \xi_{j'}$ fraction of i's. Further, as $|S^{j'}_i| \leqslant \ell$, $|T^{j'}_i| \leqslant \ell$. This implies with the list recoverability property of $C^{j'}_{out}$ that $m^{j'} \in I_{j'}$, where $I_{j'}$ is as defined in step 2(b). Finally, step 2(c) implies that $M^{j'} \in \mathcal{L}_{j'}$ as required.

The proof of correctness of the algorithm along with (4.8) shows that C is (δ, L^s)-list decodable, which completes the proof. $\square$

$$G^T \cdot M = \begin{pmatrix} (G_{j'})^T & (G \setminus G_{j'})^T \end{pmatrix} \cdot \begin{pmatrix} m_1^0 & \cdots & m_i^0 & \cdots & m_N^0 \\ & & \vdots & & \\ m_1^{j'-1} & \cdots & m_i^{j'-1} & \cdots & m_N^{j'-1} \\ m_1^{j'} & \cdots & m_i^{j'} & \cdots & m_N^{j'} \\ & & \vdots & & \\ m_1^{s-1} & \cdots & m_i^{s-1} & \cdots & m_N^{s-1} \end{pmatrix}$$
$$= \begin{pmatrix} \uparrow & & \uparrow & & \uparrow \\ C_{in}(M_1) & \cdots & C_{in}(M_i) & \cdots & C_{in}(M_N) \\ \downarrow & & \downarrow & & \downarrow \end{pmatrix}$$

Figure 4.4: Different variables in the proof of Theorem 4.5.

4.5.4 Putting it Together

We combine the results we have proved in the last couple of subsections to get the main result of this section.

Theorem 4.6. *For every fixed field $\mathbb{F}_q$, reals $0 < \delta < 1, 0 < r \leqslant 1 - H_q(\delta), \varepsilon > 0$ and integer $s \geqslant 1$, there exists linear codes C over $\mathbb{F}_q$ of block length N that are $(\delta - \varepsilon, L(N))$-list decodable with rate R such that*

$$R = r - \frac{r}{s} \sum_{i=0}^{s-1} \frac{\delta}{H_q^{-1}(1 - r + ri/s)}, \tag{4.9}$$

and $L(N) = (N/\varepsilon^2)^{O\left(s\varepsilon^{-3}\delta/(H_q^{-1}(1-r)-\delta)\right)}$. Finally, C can be constructed in time $(N/\varepsilon^2)^{O(s/(\varepsilon^6 r\delta))}$ and list decoded in time polynomial in N.

Proof. Let $\gamma > 0$ (we will define its value later). For every $0 \leqslant j \leqslant s-1$ define $r_j = r(1 - j/s)$ and $R_j = 1 - \frac{\delta}{H_q^{-1}(1-r_j)}$. The code C is going to be a multilevel concatenated code $(C_{out}^0 \times \cdots \times C_{out}^{s-1}) \circ C_{in}$, where C_{out}^j is the code from Corollary 3.7 of rate R_j and block length N' (over $\mathbb{F}_{q^a}$) and C_{in} is an $(\langle r_0, \ldots, r_{s-1} \rangle, \rho, \mathbf{L})$-nested list decodable code as guaranteed by Theorem 4.4, where for $0 \leqslant j \leqslant s-1$, $\rho_j = H_q^{-1}(1 - r_j - 2\gamma^2)$ and $L_j = q^{1/\gamma^2}$. Finally, we will use the property of C_{out}^j that it is $(1 - R_j - \gamma, q^{1/\gamma^2}, (N'/\gamma^2)^{O(\gamma^{-3}\log(1/R_j))})$-list recoverable for any $0 \leqslant \gamma \leqslant R_j$. Corollary 3.7 implies that such codes exist with (where we apply Corollary 3.7 with $R' = \max_j R_j = 1 - \delta/H_q^{-1}(1 - r/s)$)

$$q^a = (N'/\gamma^2)^{O(\gamma^{-4} H_q^{-1}(1-r/s)/\delta)}. \tag{4.10}$$

Further, as codes from Corollary 3.7 are $\mathbb{F}_q$-linear, C is a linear code.

The claims on the list decodability of C follows from the choices of R_j and r_j, Corollary 3.7 and Theorems 4.4 and 4.5. In particular, note that we invoke Theorem 4.5 with the following parameters: $\xi_j = 1 - R_j - \gamma$ and $\rho_j = H_q^{-1}(1 - r_j - 2\gamma^2)$ (which by Lemma 2.4 implies that that $\xi_j \rho_j \geqslant \delta - \varepsilon$ as long as $\gamma = \Theta(\varepsilon)$), $\ell = q^{1/\gamma^2}$ and $L = (N'/\gamma^2)^{O(\gamma^{-1}\log(\ell/R_j))}$. The choices of ℓ and γ imply that $L = (N/\varepsilon^2)^{O(\varepsilon^{-3}\log(1/R_j))}$. Now $\log(1/R_j) \leqslant \log(1/R_{min})$, where $R_{min} = \min_j R_j = 1 - \delta/H_q^{-1}(1-r)$. Finally, we use the fact that for any $0 < y < 1$, $\ln(1/y) \leqslant 1/y - 1$ to get that $\log(1/R_j) \leqslant O(1/R_{min} - 1) = O(\delta/(H_q^{-1}(1-r) - \delta))$. The claimed upper bound of $L(N)$ follows as $L(N) \leqslant L^s$ (by Theorem 4.5).

By the choices of R_j and r_j and (4.1), the rate of C is as claimed. The construction time for C is the time required to construct C_{in}, which by Theorem 4.4 is $2^{O(n/\gamma^2)}$ where n is the block length of C_{in}. Note that $n = as/r$, which by (4.10) implies that the construction time is $(N/\varepsilon^2)^{O(\varepsilon^{-6} s H_q^{-1}(1-r/s)/(r\delta))}$. The claimed running time follows by using the bound $H_q^{-1}(1 - r/s) \leqslant 1$.

We finally consider the running time of the list-decoding algorithm. We list decode the inner code(s) by brute force, which takes $2^{O(n)}$ time, that is, $t_j(n) = 2^{O(n)}$. Thus, Corollary 3.7, Theorem 4.5 and the bound on $L(N)$ implies the claimed running time complexity. □

Choosing the parameter r in the above theorem so as to maximize (4.9) gives us linear codes over any fixed field whose rate vs. list-decoding radius tradeoff meets the Blokh-Zyablov bound (4.2). As s grows, the trade-off approaches the integral form (4.3) of the Blokh-Zyablov bound.

4.6 Bibliographic Notes and Open Questions

We managed to reduce the alphabet size needed to approach capacity to a constant independent of n. However, this involved a brute-force search for a rather large code. Obtaining a "direct" algebraic construction over a constant-sized alphabet (such as variants of algebraic-geometric codes, or AG codes) might help in addressing these two issues. To this end, Guruswami and Patthak [55] define *correlated AG codes*, and describe list-decoding algorithms for those codes, based on a generalization of the Parvaresh-Vardy approach to the general class of algebraic-geometric codes (of which Reed-Solomon codes are a special case). However, to relate folded AG codes to correlated AG codes like we did for Reed-Solomon codes requires bijections on the set of rational points of the underlying algebraic curve that have some special, hard to guarantee, property. This step seems like a highly intricate algebraic task, and especially so in the interesting asymptotic setting of a family of asymptotically good AG codes over a fixed alphabet.

Our proof of existence of the requisite inner codes with good nested list decodable properties (and in particular the derandomization of the construction of such codes using

conditional expectation) is similar to the one used to establish list decodability properties of random "pseudolinear" codes in [52] (see also [49, Sec. 9.3]).

Concatenated codes were defined in the seminal thesis of Forney [40]. Its generalizations to linear multilevel concatenated codes were introduced by Blokh and Zyablov [20] and general multilevel concatenated codes were introduced by Zinoviev [108]. Our list-decoding algorithm is inspired by the argument for "unequal error protection" property of multilevel concatenated codes [109].

The results on capacity achieving list decodable codes over small alphabets (Section 4.2) and binary linear codes that are list decodable up to the Zyablov bound (Section 4.3) appeared in [58]. The result on linear codes that are list decodable up to the Blokh Zyablov bound (Section 4.5) appeared in [60].

The biggest and perhaps most challenging question left unresolved by our work is the following.

Open Question 4.1. *For every $0 < \rho < 1/2$ and every $\varepsilon > 0$ give explicit construction of binary codes that are $(\rho, O(1/\varepsilon))$-list decodable with rate $1 - H(\rho) - \varepsilon$. Further, design polynomial time list decoding algorithms that can correct up to ρ fraction of errors.*

In fact, just resolving the above question for *any* fixed ρ (even with an exponential time list-decoding algorithm) is widely open at this point.

Chapter 5

LIST DECODABILITY OF RANDOM LINEAR CONCATENATED CODES

5.1 Introduction

In Chapter 2, we saw that for any fixed alphabet of size $q \geqslant 2$ there exist codes of rate R that can be list decoded up to $H_q^{-1}(1 - R - \varepsilon)$ fraction of errors with list of size $O(1/\varepsilon)$. For linear codes one can show a similar result with lists of size $q^{O(1/\varepsilon)}$. These results are shown by choosing the code at random. However, as we saw in Chapter 4 the explicit constructions of codes over finite alphabets are nowhere close to achieving list-decoding capacity.

The linear codes in Chapter 4 are based on code concatenation. A natural question to ask is whether linear codes based on code concatenation can get us to list-decoding capacity for fixed alphabets.

In this chapter, we answer the question above in the affirmative. In particular, in Section 5.4 we show that if the outer code is random linear code and the inner codes are also (independent) random linear codes, then the resulting concatenated codes can get to within ε of the list-decoding capacity with list of constant size depending on ε only. In Section 5.5, we also show a similar result when the outer code is the folded Reed-Solomon code from Chapter 3. However, we can only show the latter result with polynomial-sized lists.

The way to interpret the results in this chapter is the following. We exhibit an ensemble of random linear codes with more structure than general random (linear) codes that achieve the list-decoding capacity. This structure gives rise to the hope of being able to list decode such a random ensemble of codes up to the list-decoding capacity. Furthermore, for designing explicit codes that meet the list-decoding capacity, one can concentrate on concatenated codes. Another corollary of our result is that we need fewer random bits to construct a code that achieves the list-decoding capacity. In particular, a general random linear code requires number of random bits that grows quadratically with the block length. On the other hand, random concatenated codes with outer codes as folded Reed-Solomon code require number of random bits that grows quasi-linearly with the block length.

The results in this chapter (and their proofs) are inspired by the following results due to Blokh and Zyablov [19] and Thommesen [102]. Blokh and Zybalov show that random concatenated linear binary codes (where both the outer and inner codes are chosen uniformly at random) have with high probability the same minimum distance as general random linear codes. Thommesen shows a similar result when the outer code is the Reed-Solomon code. The rate versus distance tradeoff achieved by random linear codes satisfies the so

called Gilbert-Varshamov (GV) bound. However, like list decodability of binary codes, explicit codes that achieve the GV bound are not known. Coming up with such explicit constructions is one of the biggest open questions in coding theory.

5.2 Preliminaries

We will consider outer codes that are defined over $\mathbb{F}_Q$, where $Q = q^k$ for some fixed $q \geqslant 2$. The outer code will have rate and block length of R and N respectively. The outer code C_{out} will either be a random linear code over $\mathbb{F}_Q$ or the folded Reed-Solomon code from Chapter 3. In the case when C_{out} is random, we will pick C_{out} by selecting $K = RN$ vectors uniformly at random from $\mathbb{F}_Q^N$ to form the rows of the generator matrix. For every position $1 \leqslant i \leqslant N$, we will choose an inner code C_{in}^i to be a random linear code over $\mathbb{F}_q$ of block length n and rate $r = k/n$. In particular, we will work with the corresponding generator matrices $\mathbf{G}_i$, where every $\mathbf{G}_i$ is a random $k \times n$ matrix over $\mathbb{F}_q$. All the generator matrices $\mathbf{G}_i$ (as well as the generator matrix for C_{out}, when we choose a random C_{out}) are chosen independently. This fact will be used crucially in our proofs.

Given the outer code C_{out} and the inner codes C_{in}^i, the resulting concatenated code $C = C_{out} \circ (C_{in}^1, \ldots, C_{in}^N)$ is constructed as follows.[1] For every codeword $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_N) \in C_{out}$, the following codeword is in C:

$$\mathbf{uG} \stackrel{def}{=} (\mathbf{u}_1\mathbf{G}_1, \mathbf{u}_2\mathbf{G}_2, \ldots, \mathbf{u}_N\mathbf{G}_N),$$

where the operations are over $\mathbb{F}_q$.

We will need the following notions of the weight of a vector. Given a vector $\mathbf{v} \in \mathbb{F}_q^{nN}$, its Hamming weight is denoted by $wt(\mathbf{v})$. Given a vector $\mathbf{y} = (y_1, \ldots, y_N) \in (\mathbb{F}_q^n)^N$ and a subset $S \subseteq [N]$, we will use $wt_S(\mathbf{y})$ to denote the Hamming weight over $\mathbb{F}_q$ of the subvector $(y_i)_{i \in S}$. Note that $wt(\mathbf{y}) = wt_{[N]}(y)$.

We will need the following lemma due to Thommesen, which is stated in a slightly different form in [102]. For the sake of completeness we also present its proof.

Lemma 5.1 ([102]). *Given a fixed outer code C_{out} of block length N and an ensemble of random inner linear codes of block length n given by generator matrices $\mathbf{G}_1, \ldots, \mathbf{G}_N$ the following is true. Let $\mathbf{y} \in \mathbb{F}_q^{nN}$. For any codeword $\mathbf{u} \in C_{out}$, any non-empty subset $S \subseteq [N]$ such that $\mathbf{u}_i \neq 0$ for all $i \in S$ and any integer $h \leqslant n|S| \cdot \left(1 - \frac{1}{q}\right)$:*

$$\Pr[wt_S(\mathbf{uG} - \mathbf{y}) \leqslant h] \leqslant q^{-n|S|\left(1 - H_q\left(\frac{h}{n|S|}\right)\right)},$$

where the probability is taken over the random choices of $\mathbf{G}_1, \ldots, \mathbf{G}_N$.

---

[1] Note that this is a slightly general form of code concatenation that is considered in Chapter 4. We did consider the current generalization briefly in Section 4.4.

Proof. Let $|S| = s$ and w.l.o.g. assume that $S = [s]$. As the choices for $\mathbf{G}_1, \ldots, \mathbf{G}_N$ are made independently, it is enough to show that the claimed probability holds for the random choices for $\mathbf{G}_1, \ldots, \mathbf{G}_s$. For any $1 \leqslant i \leqslant s$ and any $y \in \mathbb{F}_q^n$, since $\mathbf{u}_i \neq 0$, we have $\Pr_{\mathbf{G}_i}[\mathbf{u}_i\mathbf{G}_i = y] = q^{-n}$. Further, these probabilities are independent for every i. Thus, for any $\mathbf{y} = \langle y_1, \ldots, y_s \rangle \in (\mathbb{F}_q^n)^s$, $\Pr_{\mathbf{G}_1,\ldots,\mathbf{G}_s}[\mathbf{u}_i\mathbf{G}_i = y_i \text{ for every } 1 \leqslant i \leqslant s] = q^{-ns}$. This implies that:

$$\Pr_{\mathbf{G}_1,\ldots,\mathbf{G}_s}[wt_S(\mathbf{uG} - \mathbf{y}) \leqslant h] = q^{-ns} \sum_{j=0}^{h} \binom{ns}{j} (q-1)^j.$$

The claimed result follows from the Proposition 2.1. □

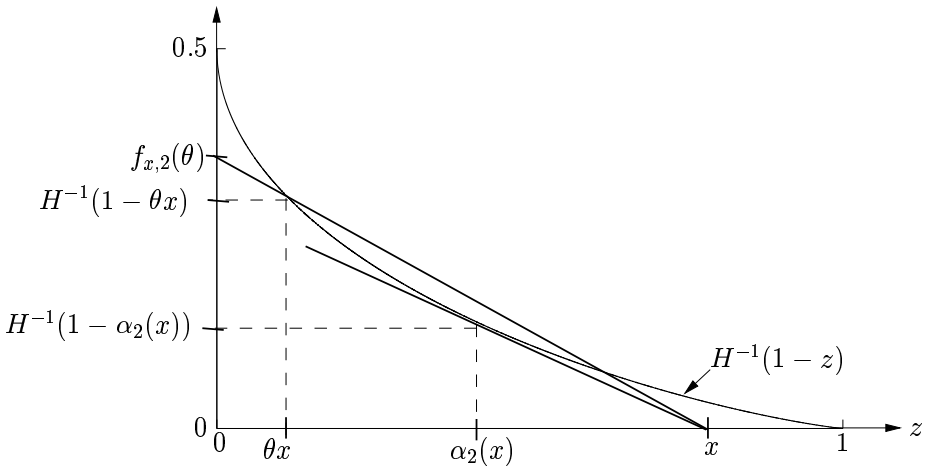


Figure 5.1: Geometric interpretations of functions $\alpha_2(\cdot)$ and $f_{x,2}(\cdot)$.

For $0 \leqslant z \leqslant 1$ define

$$\alpha_q(z) = 1 - H_q(1 - q^{z-1}). \tag{5.1}$$

We will need the following property of the function above.

Lemma 5.2. *Let $q \geqslant 2$ be an integer. For every $0 \leqslant z \leqslant 1$,*

$$\alpha_q(z) \leqslant z.$$

Proof. The proof follows from the subsequent sequence of relations:

$$\begin{aligned}\alpha_q(z) &= 1 - H_q(1-q^{z-1}) \\ &= 1 - (1-q^{z-1})\log_q(q-1) + (1-q^{z-1})\log_q(1-q^{z-1}) + q^{z-1}(z-1) \\ &= zq^{z-1} + (1-q^{z-1})\left(1 - \log_q\left(\frac{q-1}{1-q^{z-1}}\right)\right) \\ &\leqslant z,\end{aligned}$$

where the last inequality follows from the facts that $q^{z-1} \leqslant 1$ and $1 - q^{z-1} \leqslant 1 - 1/q$, which implies that $\log_q\left(\frac{q-1}{1-q^{z-1}}\right) \geqslant 1$. □

We will consider the following function

$$f_{x,q}(\theta) = (1-\theta)^{-1} \cdot H_q^{-1}(1-\theta x),$$

where $0 \leqslant \theta, x \leqslant 1$. We will need the following property of this function.[2]

Lemma 5.3 ([102]). *Let $q \geqslant 2$ be an integer. For any $x \geqslant 0$ and $0 \leqslant y \leqslant \alpha_q(x)/x$,*

$$\min_{0 \leqslant \theta \leqslant y} f_{x,q}(\theta) = (1-y)^{-1} H_q^{-1}(1-xy).$$

Proof. The proof follows from the subsequent geometric interpretations of $f_{x,q}(\cdot)$ and $\alpha_q(\cdot)$. See Figure 5.1 for a pictorial illustration of the arguments used in this proof (for $q = 2$).

First, we claim that for any $0 \leqslant z_0 \leqslant 1$, $\alpha_q(z)$ satisfies the following property: the line segment between $(\alpha_q(z_0), H_q^{-1}(1-\alpha_q(z_0)))$ and $(z_0, 0)$ is tangent to the curve $H_q^{-1}(1-z)$ at $\alpha_q(z_0)$.

Thus, we need to show that

$$\frac{-H_q^{-1}(1-\alpha_q(z_0))}{z_0 - \alpha_q(z_0)} = (H_q^{-1})'(1-\alpha_q(z_0)). \tag{5.2}$$

One can check that $(H_q^{-1})'(1-x) = \frac{-1}{H_q'(H_q^{-1}(1-x))} = \frac{-1}{\log_q(q-1) - \log_q(H_q^{-1}(1-x)) + \log_q(1-H_q^{-1}(1-x))}$. Now,

$$\begin{aligned}z_0 - \alpha_q(z_0) = {} & z_0 - 1 + (1-q^{z_0-1})\log_q(q-1) - (1-q^{z_0-1})\log_q(1-q^{z_0-1}) \\ & - q^{z_0-1}(z_0-1) \\ = {} & (1-q^{z_0-1}) \cdot \big(\log_q(q-1) - \log_q(1-q^{z_0-1}) + z_0 - 1\big) \\ = {} & H_q^{-1}(1-\alpha_q(z_0)) \cdot \big(\log_q(q-1) - \log_q(H_q^{-1}(1-\alpha_q(z_0))) \\ & + \log_q(1 - H_q^{-1}(1-\alpha_q(z_0)))\big) \\ = {} & \frac{-H_q^{-1}(1-\alpha_q(z_0))}{(H_q^{-1})'(1-\alpha_q(z_0))},\end{aligned}$$

[2] Lemma 5.3 was proven in [102] for the $q = 2$ case. Here we present the straightforward extension of the result for general q.

which proves (5.2) (where we have used the expression for $\alpha_q(z)$ and $(H_q^{-1})'(1-z)$ and the fact that $1 - q^{z-1} = H_q^{-1}(1 - \alpha_q(z))$).

We now claim that $f_{x,q}(\theta)$ is the intercept of the line segment through $(x, 0)$ and $(\theta x, H_q^{-1}(1-\theta x))$ on the "y-axis." Indeed, the "y-coordinate" increases by $H_q^{-1}(1-\theta x)$ in the line segment from x to θx. Thus, when the line segment crosses the "y-axis", it would cross at an intercept of $1/(1-\theta)$ times the "gain" going from x to θx. The lemma follows from the fact that the function $H_q^{-1}(1-r)$ is a decreasing (strictly) convex function of r and thus, the minimum of $f_{x,q}(\theta)$ would occur at $\theta = y$ provided $yx \leqslant \alpha_q(x)$. □

5.3 Overview of the Proof Techniques

In this section, we will highlight the main ideas in our proofs. Our proofs are inspired by Thommesen's proof of the following result [102]. Binary linear concatenated codes with an outer Reed-Solomon code and independently and randomly chosen inner codes meet the Gilbert-Varshamov bound[3]. Given that our proof builds on the proof of Thommesen, we start out by reviewing the main ideas in his proof.

The outer code C_{out} in [102] is a Reed-Solomon code of length N and rate R (over $\mathbb{F}_Q$) and the inner codes (over $\mathbb{F}_q$ such that $Q = q^k$ for some $k \geqslant 1$) are generated by N randomly chosen $k \times n$ generator matrices $\mathbf{G} = (\mathbf{G}_1, \ldots, \mathbf{G}_N)$, where $r = k/n$. Note that since the final code will have distance close to $H^{-1}(1-rR)$, it is enough to show that the probability of the Hamming weight of $\mathbf{uG}$ over $\mathbb{F}_q$ being at most $(H^{-1}(1-rR) - \varepsilon)nN$ (for some Reed-Solomon codeword $\mathbf{u} = (\mathbf{u}_1, \ldots, \mathbf{u}_N)$), is small. Let us now concentrate on a fixed codeword $\mathbf{u} \in C_{out}$. Now note that if for some $1 \leqslant i \leqslant N$, $\mathbf{u}_i = 0$, then for every choice of $\mathbf{G}_i$, $\mathbf{u}_i\mathbf{G}_i = 0$. Thus, only the non-zero symbols of $\mathbf{u}$ contribute to $wt(\mathbf{uG})$. Further, for a non-zero $\mathbf{u}_i$, $\mathbf{u}_i\mathbf{G}_i$ takes all the values in $\mathbb{F}_q^n$ with equal probability over the random choices of $\mathbf{G}_i$. Also for two different non-zero positions $i_1 \neq i_2$ in $\mathbf{u}$, the random variables $\mathbf{u}_{i_1}\mathbf{G}_{i_1}$ and $\mathbf{u}_{i_2}\mathbf{G}_{i_2}$ are *independent* (as the choices for $\mathbf{G}_{i_1}$ and $\mathbf{G}_{i_2}$ are independent). This implies that $\mathbf{uG}$ takes each of the possible $q^{n\cdot wt(\mathbf{u})}$ values in $\mathbb{F}_q^{nN}$ with the same probability. Thus, the total probability that $\mathbf{uG}$ has a Hamming weight of at most h is $\sum_{w=0}^{h} \binom{n\cdot wt(\mathbf{u})}{w} q^{-n\cdot wt(\mathbf{u})} \leqslant q^{-n\cdot wt(\mathbf{u})\left(1-H\left(\frac{h}{n\cdot wt(\mathbf{u})}\right)\right)}$. The rest of the argument follows by doing a careful union bound of this probability for all non zero codewords in C_{out} (using the known weight distribution of Reed-Solomon codes[4]).

Let us now try to extend the idea above to show a similar result for list decoding of a code similar to the one above (the inner codes are the same but we might change the outer

[3] A binary code of rate $\mathcal{R}$ satisfies the Gilbert-Varshamov bound if it has relative distance at least $H^{-1}(1-\mathcal{R})$.

[4] In fact, the argument works just as well for any code that has a weight distribution that is close to that of the Reed-Solomon code. In particular, it also works for folded Reed-Solomon codes– we alluded to this fact in Section 4.4.

code). We want to show that for any Hamming ball of radius at most $h = (H^{-1}(1-rR) - \varepsilon)nN$ has at most L codewords from the concatenated code C (assuming we want to show that L is the worst case list size). To show this let us look at a set of $L+1$ codewords from C and try to prove that the probability that all of them lie within some ball of radius h is small. Let $\mathbf{u}^1, \ldots, \mathbf{u}^{L+1}$ be the corresponding codewords in C_{out}. As a warm up, let us try and show this for a Hamming ball centered around $\mathbf{0}$. Thus, we need to show that all of the $L+1$ codewords $\mathbf{u}^1\mathbf{G}, \ldots, \mathbf{u}^{L+1}\mathbf{G}$ have Hamming weight at most h. Note that $L = 0$ reverts back to the setup of Thommesen, that is, any fixed codeword has weight at most h with small probability. However, we need all the codewords to have small weight. Extending Thommesen's proof would be straightforward if the random variables corresponding to each of $\mathbf{u}^i\mathbf{G}$ having small weight were independent. In particular, if we can show that for every position $1 \leqslant i \leqslant N$, all the non-zero symbols in $\{\mathbf{u}_i^1, \mathbf{u}_i^2, \ldots, \mathbf{u}_i^{L+1}\}$ are linearly independent[5] over $\mathbb{F}_q$ then the generalization of Thommesen's proof is immediate.

Unfortunately, the notion of independence discussed above does *not* hold for every $L+1$ tuple of codewords from C_{out}. A fairly common trick to get independence when dealing with linear codes is to look at messages that are linearly independent. It turns out that if C_{out} is a random linear code over $\mathbb{F}_Q$ then we have a good approximation of the the notion of independence above. Specifically, we show that with very high probability for a linearly independent (over $\mathbb{F}_Q$) set of messages[6] $\mathbf{m}^1, \ldots, \mathbf{m}^{L+1}$, the set of codewords $\mathbf{u}^1 = C_{out}(\mathbf{m}^1), \ldots, \mathbf{u}^N = C_{out}(\mathbf{m}^N)$ have the following approximate independence property. For most of the positions $1 \leqslant i \leqslant N$, most of the non-zero symbols in $\{\mathbf{u}_i^1, \ldots, \mathbf{u}_i^N\}$ are linearly independent over $\mathbb{F}_q$. It turns out that this approximate notion of independence is enough for Thommesen's proof to go through. Generalizing this argument to the case when the Hamming ball is centered around an arbitrary vector from $\mathbb{F}_q^{nN}$ is straightforward.

We remark that the notion above crucially uses the fact that the outer code is a random linear code. However, the argument is bit more tricky when C_{out} is fixed to be (say) the Reed-Solomon code. Now even if the messages $\mathbf{m}^1, \ldots, \mathbf{m}^{L+1}$ are linearly independent it is not clear that the corresponding codewords will satisfy the notion of independence in the above paragraph. Interestingly, we can show that this notion of independence is equivalent to showing good list recoverability properties for C_{out}. Reed-Solomon codes are however not known to have optimal list recoverability (which is what is required in our case). In fact, the results in Chapter 6 show that this is *impossible* for Reed-Solomon codes in general. However, as we saw in Chapter 3, folded Reed-Solomon codes *do* have optimal list recoverability and we exploit this fact in this chapter.

---

[5]Recall that $\mathbb{F}_{q^k}$ is isomorphic to $\mathbb{F}_q^k$ and hence, we can think of the symbols in $\mathbb{F}_Q$ as vectors over $\mathbb{F}_q$.

[6]Again any set of $L+1$ messages need not be linearly independent. However, it is easy to see that some subset of $J = \lceil \log_Q(L+1) \rceil$ of messages are indeed linearly independent. Hence, we can continue the argument by replacing $L+1$ with J.

5.4 List Decodability of Random Concatenated Codes

In this section, we will look at the list decodability of concatenated codes when both the outer code and the inner codes are (independent) random linear codes.

The following is the main result of this section.

Theorem 5.1. *Let q be a prime power and let $0 < r < 1$ be an arbitrary rational. Let $0 < \varepsilon < \alpha_q(r)$ be an arbitrary real, where $\alpha_q(r)$ is as defined in (5.1), and $0 < R \leqslant (\alpha_q(r) - \varepsilon)/r$ be a rational. Then the following holds for large enough integers n, N such that there exist integers k and K that satisfy $k = rn$ and $K = RN$. Let C_{out} be a random linear code over $\mathbb{F}_{q^k}$ that is generated by a random $K \times N$ matrix over $\mathbb{F}_{q^k}$. Let $C_{in}^1, \ldots, C_{in}^N$ be random linear codes over $\mathbb{F}_q$, where C_{in}^i is generated by a random $k \times n$ matrix $\mathbf{G}_i$ and the random choices for $C_{out}, \mathbf{G}_1, \ldots, \mathbf{G}_N$ are all independent. Then the concatenated code $C = C_{out} \circ (C_{in}^1, \ldots, C_{in}^N)$ is a $\left(H_q^{-1}(1 - Rr) - \varepsilon, q^{O\left(\frac{rn}{\varepsilon^2(1-R)}\right)}\right)$-list decodable code with probability at least $1 - q^{-\Omega(nN)}$ over the choices of $C_{out}, \mathbf{G}_1, \ldots, \mathbf{G}_N$. Further, with high probability, C has rate rR.*

In the rest of this section, we will prove the above theorem.

Define $Q = q^k$. Let L be the worst-case list size that we are shooting for (we will fix its value at the end). The first observation is that any $L+1$-tuple of messages $(\mathbf{m}^1, \ldots, \mathbf{m}^{L+1}) \in (\mathbb{F}_Q^K)^{L+1}$ contains at least $J = \lceil \log_Q(L+1) \rceil$ many messages that are linearly independent over $\mathbb{F}_Q$. Thus, to prove the theorem it suffices to show that with high probability, no Hamming ball over $\mathbb{F}_q^{nN}$ of radius $(H_q^{-1}(1 - rR) - \varepsilon)nN$ contains a J-tuple of codewords $(C(\mathbf{m}^1), \ldots, C(\mathbf{m}^J))$, where $\mathbf{m}^1, \ldots, \mathbf{m}^J$ are linearly independent over $\mathbb{F}_Q$.

Define $\rho = H_q^{-1}(1 - Rr) - \varepsilon$. For every J-tuple of linearly independent messages $(\mathbf{m}^1, \ldots, \mathbf{m}^J) \in (\mathbb{F}_Q^K)^J$ and received word $\mathbf{y} \in \mathbb{F}_q^{nN}$, define an indicator random variable $\mathbf{I}(\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J)$ as follows. $\mathbf{I}(\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J) = 1$ if and only if for every $1 \leqslant j \leqslant J$, $wt(C(\mathbf{m}^j) - \mathbf{y}) \leqslant \rho nN$. That is, it captures the bad event that we want to avoid. Define

$$X_C = \sum_{\mathbf{y} \in \mathbb{F}_q^{nN}} \sum_{(\mathbf{m}^1, \ldots, \mathbf{m}^J) \in \mathrm{Ind}(Q,K,J)} \mathbf{I}(\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J)$$

where $\mathrm{Ind}(Q, K, J)$ denotes the collection of subsets of $\mathbb{F}_Q$-linearly independent vectors from $\mathbb{F}_Q^K$ of size J. We want to show that with high probability $X_C = 0$. By Markov's inequality, the theorem would follow if we can show that:

$$\mathbb{E}[X_C] = \sum_{\mathbf{y} \in \mathbb{F}_q^{nN}} \sum_{(\mathbf{m}^1, \ldots, \mathbf{m}^J) \in \mathrm{Ind}(Q,K,J)} \mathbb{E}[\mathbf{I}(\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J)] \text{ is } q^{-\Omega(nN)}. \tag{5.3}$$

Note that the number of distinct possibilities for $\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J$ is upper bounded by $q^{nN} \cdot Q^{RNJ} = q^{nN(1+rRJ)}$. Fix some arbitrary choice of $\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J$. To prove (5.3), we will show that

$$q^{nN(1+rRJ)} \cdot \mathbb{E}[\mathbf{I}(\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J)] \text{ is } q^{-\Omega(nN)}. \tag{5.4}$$

Before we proceed, we need some more notation. Given vectors $\mathbf{u}^1, \ldots, \mathbf{u}^J \in \mathbb{F}_Q^N$, we define $\mathbf{Z}(\mathbf{u}^1, \ldots, \mathbf{u}^J) = (Z_1, \ldots, Z_N)$ as follows. For every $1 \leqslant i \leqslant N$, $Z_i \subseteq [J]$ denotes the largest subset such that the elements $(u_i^j)_{j \in Z_i}$ are linearly independent over $\mathbb{F}_q$ (in case of a tie choose the lexically first such set), where $\mathbf{u}^j = (u_1^j, \ldots, u_N^j)$. A subset of $\mathbb{F}_Q$ is linearly independent over $\mathbb{F}_q$ if its elements, when viewed as vectors from $\mathbb{F}_q^k$ (recall that $\mathbb{F}_{q^k}$ is isomorphic to $\mathbb{F}_q^k$) are linearly independent over $\mathbb{F}_q$. If $u_i^j \in Z_i$ then we will call u_i^j a *good* symbol. Note that a good symbol is always non-zero. We will also define another partition of all the good symbols, $\mathbf{T}(\mathbf{u}^1, \ldots, \mathbf{u}^J) = (T_1, \ldots, T_J)$ by setting $T_j = \{i | j \in Z_i\}$ for $1 \leqslant j \leqslant J$.

Since $\mathbf{m}^1, \ldots, \mathbf{m}^J$ are linearly independent over $\mathbb{F}_Q$, the corresponding codewords in C_{out} are distributed uniformly in $\mathbb{F}_Q^N$. In other words, for any fixed $(\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (\mathbb{F}_Q^N)^J$,

$$\Pr_{C_{out}}\left[\bigwedge_{j=1}^{J} C_{out}(\mathbf{m}^j) = \mathbf{u}^j\right] = Q^{-NJ} = q^{-rnNJ}. \tag{5.5}$$

Recall that we denote the (random) generator matrices for the inner code C_{in}^i by $\mathbf{G}_i$ for every $1 \leqslant i \leqslant N$. Also note that every $(\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (\mathbb{F}_Q^N)^J$ has a unique $\mathbf{Z}(\mathbf{u}^1, \ldots, \mathbf{u}^J)$. In other words, the 2^{NJ} choices of $\mathbf{Z}$ partition the tuples in $(\mathbb{F}_Q^N)^J$.

Let $h = \rho n N$. Consider the following calculation (where the dependence of $\mathbf{Z}$ and $\mathbf{T}$ on $\mathbf{u}^1, \ldots, \mathbf{u}^J$ have been suppressed for clarity):

$$\begin{aligned}
\mathbb{E}[\mathbf{I}(\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J)] = & \sum_{(\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (\mathbb{F}_Q^N)^J} \Pr_{\mathbf{G}=(\mathbf{G}_1, \ldots, \mathbf{G}_N)}\left[\bigwedge_{j=1}^{J} wt(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right] && (5.6)\\
& \qquad \cdot \Pr_{C_{out}}\left[\bigwedge_{j=1}^{J} C_{out}(\mathbf{m}^j) = \mathbf{u}^j\right] \\
= \; & q^{-rnNJ} \sum_{(\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (\mathbb{F}_Q^N)^J} \Pr_{\mathbf{G}=(\mathbf{G}_1, \ldots, \mathbf{G}_N)}\left[\bigwedge_{j=1}^{J} wt(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right] && (5.7)\\
\leqslant \; & q^{-rnNJ} \sum_{(\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (\mathbb{F}_Q^N)^J} \Pr_{\mathbf{G}=(\mathbf{G}_1, \ldots, \mathbf{G}_N)}\left[\bigwedge_{j=1}^{J} wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right] && (5.8)\\
= \; & q^{-rnNJ} \sum_{(\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (\mathbb{F}_Q^N)^J} \prod_{j=1}^{J} \Pr_{\mathbf{G}}\left[wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right] && (5.9)
\end{aligned}$$

In the above (5.6) follows from the fact that the (random) choices for C_{out} and $\mathbf{G} = (\mathbf{G}_1, \ldots, \mathbf{G}_N)$ are all independent. (5.7) follows from (5.5). (5.8) follows from the simple

fact that for every $\mathbf{y} \in (\mathbb{F}_q^n)^N$ and $T \subseteq [N]$, $wt_T(\mathbf{y}) \leqslant wt(\mathbf{y})$. (5.9) follows from the subsequent argument. By definition of conditional probability, $\Pr_{\mathbf{G}}\left[\bigwedge_{j=1}^{J} wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right]$ is the same as

$$\Pr_{\mathbf{G}}\left[wt_{T_J}(\mathbf{u}^J\mathbf{G} - \mathbf{y}) \leqslant h \middle| \bigwedge_{j=1}^{J-1} wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right] \cdot \Pr_{\mathbf{G}}\left[\bigwedge_{j=1}^{J-1} wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right].$$

Now as all symbols corresponding to T_J are good symbols, for every $i \in T_J$, the value of $\mathbf{u}_i^J\mathbf{G}_i$ is independent of the values of $\{\mathbf{u}_i^1\mathbf{G}_i, \ldots, \mathbf{u}_i^{J-1}\mathbf{G}_i\}$. Further since each of $\mathbf{G}_1, \ldots, \mathbf{G}_N$ are chosen independently (at random), the event $wt_{T_J}(\mathbf{u}^J\mathbf{G} - \mathbf{y}) \leqslant h$ is independent of the event $\bigwedge_{j=1}^{J-1} wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h$. Thus, $\Pr_{\mathbf{G}}\left[\bigwedge_{j=1}^{J} wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right]$ is

$$\Pr_{\mathbf{G}}\left[wt_{T_J}(\mathbf{u}^J\mathbf{G} - \mathbf{y}) \leqslant h\right] \cdot \Pr_{\mathbf{G}}\left[\bigwedge_{j=1}^{J-1} wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right].$$

Inductively applying the argument above gives (5.9).

Further,

$$\mathbb{E}[\mathbf{I}(\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J)] = \sum_{(\mathbf{u}^1,\ldots,\mathbf{u}^J)\in(\mathbb{F}_Q^N)^J} \prod_{j=1}^{J} q^{-rnN} \cdot \Pr_{\mathbf{G}}\left[wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right] \tag{5.10}$$

$$= \sum_{(d_1,\ldots,d_J)\in\{0,\ldots,N\}^J} \sum_{\substack{(\mathbf{u}^1,\ldots,\mathbf{u}^J)\in(\mathbb{F}_Q^N)^J,\\ (|T_1|=d_1,\ldots,|T_J|=d_J)}} \prod_{j=1}^{J} \frac{\Pr_{\mathbf{G}}\left[wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right]}{q^{rnN}} \tag{5.11}$$

$$\leqslant \sum_{\substack{(d_1,\ldots,d_J)\\ \in\{0,\ldots,N\}^J}} q^{JN+(rn+J)\sum_{j=1}^{J} d_j} \prod_{\substack{j=1,\\ |T_j|=d_j}}^{J} \frac{\Pr_{\mathbf{G}}\left[wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right]}{q^{rnN}} \tag{5.12}$$

$$= \sum_{(d_1,\ldots,d_J)\in\{0,\ldots,N\}^J} \prod_{\substack{j=1,\\ |T_j|=d_j}}^{J} \frac{\Pr_{\mathbf{G}}\left[wt_{T_j}(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h\right]}{q^{n\left(-r(d_j-N)-\frac{Jd_j}{n}-\frac{N}{n}\right)}} \tag{5.13}$$

In the above (5.10), (5.11), (5.13) follow from rearranging and grouping the summands. (5.12) uses the following argument. Given a fixed $\mathbf{Z} = (Z_1, \ldots, Z_N)$, the number of tuples $(\mathbf{u}^1, \ldots, \mathbf{u}^J)$ such that $\mathbf{Z}(\mathbf{u}^1, \ldots, \mathbf{u}^J) = \mathbf{Z}$ is at most $U = \prod_{i=1}^{N} q^{|Z_i|k} \cdot q^{|Z_i|(J-|Z_i|)}$, where the $q^{|Z_i|k}$ is an upper bound on the number of $|Z_i|$ linearly independent vectors from $\mathbb{F}_q^k$ and $q^{|Z_i|(J-|Z_i|)}$ follows from the fact that every bad symbol $\{u_i^j\}_{j\notin Z_i}$ has to take a value that is a

linear combination of the symbols $\{u_i^j\}_{j \in Z_i}$. Now $U \leqslant \prod_{i=1}^{N} q^{|Z_i|(k+J)} = q^{(k+J)\sum_{i=1}^{N}|Z_i|} = q^{(k+J)\sum_{j=1}^{J}|T_j|}$. Finally, there are $2^{JN} \leqslant q^{JN}$ distinct choices for $\mathbf{Z}$.

(5.13) implies the following

$$q^{nN(1+rRJ)} \cdot \mathbb{E}[\mathbf{I}(\mathbf{y}, \mathbf{m}^1, \ldots, \mathbf{m}^J)] \leqslant \sum_{(d_1,\ldots,d_J) \in \{0,\ldots,N\}^J} \prod_{j=1}^{J} E_j$$

where

$$E_j = q^{-n\left(-r(d_j - N(1-R)) - \frac{N}{J} - \frac{Jd_j}{n} - \frac{N}{n}\right)} \cdot \Pr_{\mathbf{G}}\left[wt_{T_j}(\mathbf{u}_j\mathbf{G} - \mathbf{y}) \leqslant h\right].$$

We now proceed to upper bound E_j by $q^{-\Omega(nN)}$ for every $1 \leqslant j \leqslant J$. Note that this will imply the claimed result as there are at most $(N+1)^J = q^{o(nN)}$ choices for different values of d_j's.

We first start with the case when $d_j < d^*$, where

$$d^* = N(1 - R - \gamma),$$

for some parameter $0 < \gamma < 1 - R$ to be defined later. In this case we use the fact that $\Pr_{\mathbf{G}}\left[wt_{T_j}(\mathbf{u}_j\mathbf{G} - \mathbf{y}) \leqslant h\right] \leqslant 1$. Thus, we would be done if we can show that

$$\frac{1}{N}\left(r\left(d_j - N(1-R)\right) + \frac{N}{J} + \frac{Jd_i}{n} + \frac{N}{n}\right) \leqslant -\delta' < 0,$$

for some $\delta' > 0$ that we will choose soon. The above would be satisfied if

$$\frac{d_j}{N} < (1-R) - \frac{1}{r}\left(\frac{1}{J} + \frac{Jd_j}{nN} + \frac{1}{n}\right) - \frac{\delta'}{r},$$

which is satisfied if we choose $\gamma > \frac{2}{r}\left(\frac{1}{J} + \frac{Jd_j}{nN} + \frac{1}{n}\right) + \frac{\delta'}{r}$ as $d_j < d^*$. Note that if $n > 2J\left(\frac{Jd_j}{N} + 1\right)$ and if we set $\delta' = \frac{1}{J}$, it is enough to choose $\gamma = \frac{4}{Jr}$.

We now turn our attention to the case when $d_j \geqslant d^*$. The arguments are very similar to the ones employed by Thommesen in the proof of his main theorem in [102]. In this case, by Lemma 5.1 we have

$$E_j \leqslant q^{-nd_j\left(1 - H_q\left(\frac{h}{nd_j}\right) - r\left(1 - \frac{N(1-R)}{d_j}\right) - \frac{N}{d_j J} - \frac{J}{n} - \frac{N}{nd_j}\right)}.$$

The above implies that we can show that E_j is $q^{-\Omega(nN(1-R-\gamma))}$ provided we show that for every $d^* \leqslant d \leqslant N$,

$$h/(nd) \leqslant H_q^{-1}\left(1 - r\left(1 - \frac{N(1-R)}{d}\right) - \frac{N}{dJ} - \frac{J}{n} - \frac{N}{nd}\right) - \delta,$$

for $\delta = \varepsilon/3$. Now if $n \geqslant 2J^2$, then both $\frac{J}{n} \leqslant \frac{N}{2Jd}$ and $\frac{N}{nd} \leqslant \frac{N}{2Jd}$. In other words, $\frac{J}{n} + \frac{N}{nd} \leqslant \frac{N}{Jd}$. Using the fact that H_q^{-1} is increasing, the above is satisfied if

$$h/(nd) \leqslant H_q^{-1}\left(1 - r\left(1 - \frac{N(1-R-\gamma)}{d}\right) - \frac{2N}{dJ}\right) - \delta,$$

By Lemma 5.4, as long as $J \geqslant 4c'_q/(\delta^2(1-R))$ (and the conditions on γ are satisfied), the above can be satisfied by picking

$$h/(nN) = H_q^{-1}(1 - rR) - 3\delta = \rho,$$

as required. We now verify that the conditions on γ in Lemma 5.4 are satisfied by our choice of $\gamma = \frac{4}{Jr}$. Note that if we choose $J = 4c'_q/(\delta^2(1-R))$, we will have $\gamma = \frac{\delta^2(1-R)}{c'_q r}$. Now, as $R < 1$, we also have $\gamma \leqslant \delta^2/(rc'_q)$. Finally, we show that $\gamma \leqslant (1-R)/2$. Indeed

$$\gamma = \frac{\delta^2(1-R)}{c'_q r} = \frac{\varepsilon^2(1-R)}{9c'_q r} \leqslant \frac{\varepsilon(1-R)}{9r} \leqslant \frac{\alpha_q(r)(1-R)}{9r} < \frac{1-R}{2},$$

where the first inequality follows from the facts that $c'_q \geqslant 1$ and $\varepsilon \leqslant 1$. The second inequality follows from the assumption on ε. The third inequality follows from Lemma 5.2.

Note that $J = O\left(\frac{1}{(1-R)\varepsilon^2}\right)$, which implies $L = Q^{O(1/((1-R)\varepsilon^2)}$ as claimed in the statement of the theorem.

We still need to argue that with high probability the rate of the code $C = C_{out} \circ (C_{in}^1, \ldots, C_{in}^N)$ is rR. One way to argue this would be to show that with high probability all of the generator matrices have full rank. However, this is not the case: in fact, with some non-negligible probability at least one of them will not have full rank. However, we claim that with high probability C has distance > 0. Note that as C is a linear code, this implies that for every distinct pair of messages $\mathbf{m}^1 \neq \mathbf{m}^2 \in \mathbb{F}_Q^K$ are mapped to distinct codewords, which implies that C has q^{rnRN} codewords, as desired. We now briefly argue why C has distance > 0. The proof above in fact implies that with high probability C has distance about $H_q^{-1}(1-rR)nN$. It is easy to see that to show that C has distance at least h, it is enough to show that with high probability $\sum_{\mathbf{m} \in \mathbb{F}_Q^K} \mathbf{I}(\mathbf{0}, \mathbf{m}) = 0$. Note that this is a special case of our proof, with $J = 1$ and $\mathbf{y} = \mathbf{0}$ and hence, with probability at least $1 - q^{\Omega(nN)}$, the code C has large distance. The proof is complete.

Remark 5.1. *In a typical use of concatenated codes, the block lengths of the inner and outer codes satisfy $n = \Theta(\log N)$, in which case the concatenated code of Theorem 5.1 is list decodable with lists of size $N^{O\left(\varepsilon^{-2}(1-R)^{-1}\right)}$. However, the proof of Theorem 5.1 also works with smaller n. In particular as long as n is at least $3J^2$, the proof of Theorem 5.1 goes through. Thus, with n in $\Theta(J^2)$, one can get concatenated codes that are list decodable up to the list-decoding capacity with lists of size $q^{O\left(\varepsilon^{-6}(1-R)^{-3}\right)}$.*

Lemma 5.4. *Let q be a prime power, and $1 \leqslant n \leqslant N$ be integers. Let $0 < r, R < 1$ be rationals and $\delta > 0$ be a real such that $R \leqslant (\alpha_q(r) - \delta)/r$ and $\delta \leqslant \alpha_q(r)$, where $\alpha_q(r)$ is as defined in (5.1). Let $\gamma > 0$ be a real such that $\gamma \leqslant \min\left(\frac{1-R}{2}, \frac{\delta^2}{c'_q r}\right)$, where c'_q is the constant that depends only on q from Lemma 2.4. Then for all integers $J \geqslant \frac{4c'_q}{\delta^2(1-R)}$ and $h \leqslant (H_q^{-1}(1-rR) - 2\delta)nN$ the following is satisfied. For every integer $(1-R-\gamma)N \leqslant d \leqslant N$,*

$$\frac{h}{nd} \leqslant H_q^{-1}\left(1 - r\left(1 - \frac{N(1-R-\gamma)}{d}\right) - \frac{2N}{Jd}\right). \tag{5.14}$$

Proof. Using the fact H_q^{-1} is an increasing function, (5.14) is satisfied if the following is true (where $d^* = (1-R-\gamma)N$):

$$\frac{h}{nN} \leqslant \min_{d^* \leqslant d \leqslant N}\left\{\left(\frac{d}{N}\right) \cdot H_q^{-1}\left(1 - r\left(1 - \frac{N(1-R-\gamma)}{d}\right) - \frac{2N}{d^*J}\right)\right\}.$$

Define a new variable $\theta = 1 - N(1-R-\gamma)/d$. Note that as $d^* = (1-R-\gamma)N \leqslant d \leqslant N$, $0 \leqslant \theta \leqslant R+\gamma$. Also $d/N = (1-R-\gamma)(1-\theta)^{-1}$. Thus, the above inequality would be satisfied if

$$\frac{h}{nN} \leqslant (1-R-\gamma) \min_{0 \leqslant \theta \leqslant R+\gamma}\left\{(1-\theta)^{-1} H_q^{-1}\left(1 - r\theta - \frac{2}{(1-R-\gamma)J}\right)\right\}.$$

Again using the fact that H_q^{-1} is an increasing function along with the fact that $\gamma \leqslant (1-R)/2$, we get that the above is satisfied if

$$\frac{h}{nN} \leqslant (1-R-\gamma) \min_{0 \leqslant \theta \leqslant R+\gamma}\left\{(1-\theta)^{-1} H_q^{-1}\left(1 - r\theta - \frac{4}{(1-R)J}\right)\right\}.$$

By Lemma 2.4, if $J \geqslant \frac{4c'_q}{\delta^2(1-R)}$, then[7] $H_q^{-1}\left(1 - r\theta - \frac{4}{(1-R)J}\right) \geqslant H_q^{-1}(1-r\theta) - \delta$. Since for every $0 \leqslant \theta \leqslant R+\gamma$, $(1-R-\gamma)(1-\theta)^{-1}\delta \leqslant \delta$, the above equation would be satisfied if

$$\frac{h}{nN} \leqslant (1-R-\gamma) \min_{0 < \theta \leqslant R+\gamma} f_{r,q}(\theta) - \delta.$$

Note that the assumptions $\gamma \leqslant \delta^2/(rc'_q) \leqslant \delta/r$ (as $\delta \leqslant 1$ and $c'_q \geqslant 1$) and $R \leqslant (\alpha_q(r) - \delta)/r$, we have $R + \gamma \leqslant \alpha_q(r)/r$. Thus, by using Lemma 5.3 we get that $(1-R-\gamma)\min_{0<\theta\leqslant R+\gamma} f_{r,q}(\theta) = H_q^{-1}(1-rR-r\gamma)$. By Lemma 2.4, the facts that $\gamma \leqslant \delta^2/(rc'_q)$ and H_q^{-1} is increasing, we have $H_q^{-1}(1-rR-r\gamma) \geqslant H_q^{-1}(1-rR) - \delta$. This implies that (5.14) is satisfied if $h/(nN) \leqslant H_q^{-1}(1-rR) - 2\delta$, as desired. □

[7] We also use the fact that H_q^{-1} is increasing.

5.5 Using Folded Reed-Solomon Code as Outer Code

In this section, we will prove a result similar to Theorem 5.1, with the outer code being the folded Reed-Solomon code from Chapter 3. The proof will make crucial use of the list recoverability of folded Reed-Solomon codes. Before we begin we will need the following definition and results.

5.5.1 Preliminaries

We will need the following notion of independence.

Definition 5.1 (Independent tuples). *Let C be a code of block length N and rate R defined over $\mathbb{F}_{q^k}$. Let $J \geqslant 1$ and $0 \leqslant d_1, \ldots, d_J \leqslant N$ be integers. Let $\mathbf{d} = \langle d_1, \ldots, d_J \rangle$. An ordered tuple of codewords $(c^1, \ldots, c^J)$, $c^j \in C$ is said to be $(\mathbf{d}, \mathbb{F}_q)$-independent if the following holds. $d_1 = wt(c^1)$ and for every $1 < j \leqslant J$, d_j is the number of positions i such that c_i^j is $\mathbb{F}_q$-independent of the vectors $\{c_i^1, \ldots, c_i^{j-1}\}$, where $c^\ell = (c_1^\ell, \ldots, c_N^\ell)$.*

Note that for any tuple of codewords $(c^1, \ldots, c^J)$ there exists a unique $\mathbf{d}$ such that it is $(\mathbf{d}, \mathbb{F}_q)$-independent.

The next result will be crucial in our proof.

Lemma 5.5. *Let C be a folded Reed-Solomon code of block length N that is defined over $\mathbb{F}_Q$ with $Q = q^k$ as guaranteed by Theorem 3.6. For any L-tuple of codewords from C, where $L \geqslant J \cdot (N/\varepsilon^2)^{O\left(\varepsilon^{-1} J \log(q/R)\right)}$ (where $\varepsilon > 0$ is same as the one in Theorem 3.6), there exists a sub-tuple of J codewords such that the J-tuple is $(\mathbf{d}, \mathbb{F}_q)$-independent, where $\mathbf{d} = \langle d_1, \ldots, d_J \rangle$ such that for every $1 \leqslant j \leqslant J$, $d_j \geqslant (1 - R - \varepsilon)N$.*

Proof. The proof is constructive. In particular, given an L-tuple of codewords, we will construct a J sub-tuple with the required property. The correctness of the procedure will hinge on the list recoverability of the folded Reed-Solomon code as guaranteed by Theorem 3.6.

We will construct the final sub-tuple iteratively. In the first step, pick any non-zero codeword in the L-tuple– call it c^1. Note that as C has distance $(1 - R)N$ (and $\mathbf{0} \in C$), c^1 is non-zero in at least $d_1 \geqslant (1 - R)N > (1 - R - \varepsilon)N$ many places. Note that c^1 is vacuously independent of the "previous" codewords in these positions. Now, say that the procedure has chosen codewords $c^1, \ldots, c^s$ such that the tuple is $(\mathbf{d}', \mathbb{F}_q)$-independent for $\mathbf{d}' = \langle d_1, \ldots, d_s \rangle$, where for every $1 \leqslant j \leqslant s$, $d_j \geqslant (1 - R - \varepsilon)N$. For every $1 \leqslant i \leqslant N$, define S_i to be the $\mathbb{F}_q$-span of the vectors $\{c_i^1, \ldots, c_i^s\}$ in $\mathbb{F}_q^k$. Note that $|S_i| \leqslant q^s$. Call $c = (c_1, \ldots, c_N) \in C$ to be a *bad* codeword, if there does not exist any $d_{s+1} \geqslant (1 - R - \varepsilon)N$ such that $(c^1, \ldots, c^s, c)$ is $(\mathbf{d}, \mathbb{F}_q)$-independent for $\mathbf{d} = \langle d_1, \ldots, d_{s+1} \rangle$. In other words, c is a bad codeword if and only if some $T \subset [N]$ with $|T| = (R + \varepsilon)N$ satisfies $c_i \in S_i$ for every $i \in T$. Put differently, c satisfies the condition of being in the output list for list recovering C with input $S_1, \ldots, S_N$ and agreement fraction $R + \varepsilon$. Thus, by Theorem 3.6, the number of such bad codewords is $U = (N/\varepsilon^2)^{O\left(\varepsilon^{-1} s \log(q/R)\right)} \leqslant (N/\varepsilon^2)^{O\left(\varepsilon^{-1} J \log(q/R)\right)}$, where J is

the number of steps for which this greedy procedure can be applied. Thus, as long as at each step there are strictly more than U codewords from the original L-tuple of codewords left, we can continue this greedy procedure. Note that we can continue this procedure J times, as long as $J \leqslant L/U$. The proof is complete. □

Finally, we will need a bound on the number of independent tuples for folded Reed-Solomon codes.

Lemma 5.6. *Let C be a folded Reed-Solomon code of block length N and rate $0 < R < 1$ that is defined over $\mathbb{F}_Q$, where $Q = q^k$. Let $J \geqslant 1$ and $0 \leqslant d_1, \ldots, d_J \leqslant N$ be integers and define $\mathbf{d} = \langle d_1, \ldots, d_J \rangle$. Then the number of $(\mathbf{d}, \mathbb{F}_q)$-independent tuples in C is at most*

$$q^{NJ(J+1)} \prod_{j=1}^{J} Q^{\max(d_j - N(1-R)+1, 0)}.$$

Proof. Given a tuple $(c^1, \ldots, c^J)$ that is $(\mathbf{d}, \mathbb{F}_q)$-independent, define $T_j \subseteq [N]$ with $|T_j| = d_j$, for $1 \leqslant j \leqslant J$ to be the set of positions i, where c_i^j is linearly independent of the values $\{c_i^1, \ldots, c_i^{j-1}\}$. We will estimate the number of $(\mathbf{d}, \mathbb{F}_q)$-independent tuples by first estimating a bound U_j on the number of choices for the j^{th} codeword in the tuple (given a fixed choice of the first $j-1$ codewords). To complete the proof, we will show that

$$U_j \leqslant q^{N(J+1)} \cdot Q^{\max(d_j - N(1-R)+1, 0)}.$$

A codeword $c \in C$ can be the j^{th} codeword in the tuple in the following way. Now for every position in $[N] \setminus T_j$, c can take at most $q^{j-1} \leqslant q^J$ values (as in these position the value has to lie in the $\mathbb{F}_q$ span of the values of the first $j-1$ codewords in that position). Since C is folded Reed-Solomon, once we fix the values at positions in $[N] \setminus T_j$, the codeword will be completely determined once any $\max(RN - (N - d_j) + 1, 0) = \max(d_j - N(1-R) + 1, 0)$ positions in T_j are chosen (w.l.o.g. assume that they are the "first" so many positions). The number of choices for T_j is $\binom{N}{d_j} \leqslant 2^N \leqslant q^N$. Thus, we have

$$U_j \leqslant q^N \cdot (q^J)^{N-d_j} \cdot Q^{\max(d_j - N(1-R)+1, 0)} \leqslant q^{N(J+1)} \cdot Q^{\max(d_j - N(1-R)+1), 0)},$$

as desired. □

5.5.2 The Main Result

We will now prove the following result.

Theorem 5.2. *Let q be a prime power and let $0 < r < 1$ be an arbitrary rational. Let $0 < \varepsilon < \alpha_q(r)$ an arbitrary real, where $\alpha_q(r)$ is as defined in (5.1), and $0 < R \leqslant (\alpha_q(r) - \varepsilon)/r$ be a rational. Then the following holds for large enough integers n, N such that there exist integers k and K that satisfy $k = rn$ and $K = RN$. Let C_{out} be a folded Reed-Solomon*

code over $\mathbb{F}_{q^k}$ of block length N and rate R. Let $C_{in}^1, \ldots, C_{in}^N$ be random linear codes over $\mathbb{F}_q$, where C_{in}^i is generated by a random $k \times n$ matrix $\mathbf{G}_i$ over $\mathbb{F}_q$ and the random choices for $\mathbf{G}_1, \ldots, \mathbf{G}_N$ are all independent. Then the concatenated code $C = C_{out} \circ (C_{in}^1, \ldots, C_{in}^N)$ is a $\left(H_q^{-1}(1-Rr) - \varepsilon, \left(\frac{N}{\varepsilon^2}\right)^{O\left(\varepsilon^{-4}(1-R)^{-2}\log(1/R)\right)}\right)$-list decodable code with probability at least $1 - q^{-\Omega(nN)}$ over the choices of $\mathbf{G}_1, \ldots, \mathbf{G}_N$. Further, with high probability, C has rate rR.

In the rest of this section, we will prove the above theorem.

Define $Q = q^k$. Let L be the worst-case list size that we are shooting for (we will fix its value at the end). By Lemma 5.5, any $L+1$-tuple of C_{out} codewords $(\mathbf{u}^0, \ldots, \mathbf{u}^L) \in (C_{out})^{L+1}$ contains at least $J = \left\lfloor (L+1)/(N/\gamma^2)^{O\left(\gamma^{-1}J\log(q/R)\right)} \right\rfloor$ codewords that form an $(\mathbf{d}, \mathbb{F}_q)$-independent tuple, for some $\mathbf{d} = \langle d_1, \ldots, d_J \rangle$, with $d_j \geqslant (1-R-\gamma)N$ (we will specify γ, $0 < \gamma < 1-R$, later). Thus, to prove the theorem it suffices to show that with high probability, no Hamming ball in $\mathbb{F}_q^{nN}$ of radius $(H_q^{-1}(1-rR) - \varepsilon)nN$ contains a J-tuple of codewords $(\mathbf{u}^1\mathbf{G}, \ldots, \mathbf{u}^J\mathbf{G})$, where $(\mathbf{u}^1, \ldots, \mathbf{u}^J)$ is a J-tuple of folded Reed-Solomon codewords that is $(\mathbf{d}, \mathbb{F}_q)$-independent. For the rest of the proof, we will call a J-tuple of C_{out} codewords $(\mathbf{u}^1, \ldots, \mathbf{u}^J)$ a *good* tuple if it is $(\mathbf{d}, \mathbb{F}_q)$-independent for some $\mathbf{d} = \langle d_1, \ldots, d_J \rangle$, where $d_j \geqslant (1-R-\gamma)N$ for every $1 \leqslant j \leqslant J$.

Define $\rho = H_q^{-1}(1-Rr) - \varepsilon$. For every good J-tuple of C_{out} codewords $(\mathbf{u}^1, \ldots, \mathbf{u}^J)$ and received word $\mathbf{y} \in \mathbb{F}_q^{nN}$, define an indicator variable $\mathbf{I}(\mathbf{y}, \mathbf{u}^1, \ldots, \mathbf{u}^J)$ as follows. $\mathbf{I}(\mathbf{y}, \mathbf{u}^1, \ldots, \mathbf{u}^J) = 1$ if and only if for every $1 \leqslant j \leqslant J$, $wt(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant \rho nN$. That is, it captures the bad event that we want to avoid. Define

$$X_C = \sum_{\mathbf{y} \in \mathbb{F}_q^{nN}} \sum_{\text{good } (\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (C_{out})^J} \mathbf{I}(\mathbf{y}, \mathbf{u}^1, \ldots, \mathbf{u}^J).$$

We want to show that with high probability $X_C = 0$. By Markov's inequality, the theorem would follow if we can show that:

$$\mathbb{E}[X_C] = \sum_{\mathbf{y} \in \mathbb{F}_q^{nN}} \sum_{\text{good } (\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (C_{out})^J} \mathbb{E}[\mathbf{I}(\mathbf{y}, \mathbf{u}^1, \ldots, \mathbf{u}^J)] \leqslant q^{-\Omega(nN)}. \tag{5.15}$$

Before we proceed, we need a final bit of notation. For a good tuple $(\mathbf{u}^1, \ldots, \mathbf{u}^J)$ and every $1 \leqslant j \leqslant J$, define $T_j(\mathbf{u}^1, \ldots, \mathbf{u}^J) \subseteq [N]$ to be the set of positions i such that $\mathbf{u}_i^j$ is $\mathbb{F}_q$-independent of the set $\{\mathbf{u}_i^1, \ldots, \mathbf{u}_i^{j-1}\}$. Note that since the tuple is good, $|T_j(\mathbf{u}^1, \ldots, \mathbf{u}^J)| \geqslant (1-R-\gamma)N$.

Let $h = \rho nN$. Consider the following sequence of inequalities (where below we have suppressed the dependence of T_j on $(\mathbf{u}^1, \ldots, \mathbf{u}^J)$ for clarity):

$$\mathbb{E}[X_C] = \sum_{\mathbf{y} \in \mathbb{F}_q^{nN}} \sum_{\text{good } (\mathbf{u}^1, \ldots, \mathbf{u}^J) \in (C_{out})^J} \Pr_{\mathbf{G}=(\mathbf{G}_1, \ldots, \mathbf{G}_N)} \left[\bigwedge_{j=1}^{J} wt(\mathbf{u}^j\mathbf{G} - \mathbf{y}) \leqslant h \right] \tag{5.16}$$

$$\leqslant \sum_{\mathbf{y}\in\mathbb{F}_q^{nN}} \sum_{\text{good } (\mathbf{u}^1,\ldots,\mathbf{u}^J)\in(C_{out})^J} \Pr_{\mathbf{G}=(\mathbf{G}_1,\ldots,\mathbf{G}_N)}\left[\bigwedge_{j=1}^{J} wt_{T_j}(\mathbf{u}^j\mathbf{G}-\mathbf{y}) \leqslant h\right] \tag{5.17}$$

$$= \sum_{\mathbf{y}\in\mathbb{F}_q^{nN}} \sum_{\text{good } (\mathbf{u}^1,\ldots,\mathbf{u}^J)\in(C_{out})^J} \prod_{j=1}^{J} \Pr_{\mathbf{G}}\left[wt_{T_j}(\mathbf{u}^i\mathbf{G}-\mathbf{y}) \leqslant h\right] \tag{5.18}$$

$$\leqslant \sum_{\mathbf{y}\in\mathbb{F}_q^{nN}} \sum_{\text{good } (\mathbf{u}^1,\ldots,\mathbf{u}^J)\in(C_{out})^J} \prod_{j=1}^{J} q^{-n|T_j|\left(1-H_q\left(\frac{h}{n|T_j|}\right)\right)} \tag{5.19}$$

$$= \sum_{\mathbf{y}\in\mathbb{F}_q^{nN}} \sum_{\substack{(d_1,\ldots,d_J)\\ \in\{(1-R-\gamma)N,\ldots,N\}^J}} \sum_{\substack{\text{good } (\mathbf{u}^1,\ldots,\mathbf{u}^J)\in(C_{out})^J,\\ (|T_1|=d_1,\ldots,|T_J|=d_J)}} \prod_{j=1}^{J} q^{-nd_j\left(1-H_q\left(\frac{h}{nd_j}\right)\right)} \tag{5.20}$$

$$\leqslant \sum_{\substack{(d_1,\ldots,d_J)\\ \in\{(1-R-\gamma)N,\ldots,N\}^J}} q^{nN}\cdot q^{NJ(J+1)} \prod_{j=1}^{J} Q^{\max(d_j-(1-R)N+1,0)} \prod_{j=1}^{J} q^{-nd_j\left(1-H_q\left(\frac{h}{nd_j}\right)\right)} \tag{5.21}$$

$$\leqslant \sum_{\substack{(d_1,\ldots,d_J)\\ \in\{(1-R-\gamma)N,\ldots,N\}^J}} q^{nN}\cdot q^{NJ(J+1)} \prod_{j=1}^{J} Q^{d_j-(1-R-\gamma)N} \prod_{j=1}^{J} q^{-nd_j\left(1-H_q\left(\frac{h}{nd_j}\right)\right)} \tag{5.22}$$

$$= \sum_{\substack{(d_1,\ldots,d_J)\\ \in\{(1-R-\gamma)N,\ldots,N\}^J}} \prod_{j=1}^{J} q^{-nd_j\left(1-H_q\left(\frac{h}{nd_j}\right)-r\left(1-\frac{(1-R-\gamma)N}{d_j}\right)-\frac{N}{Jd_j}-\frac{N(J+1)}{nd_j}\right)}. \tag{5.23}$$

In the above (5.16) follows from the definition of the indicator variable. (5.17) follows from the simple fact that for every vector $\mathbf{u}$ of length N and every $T \subseteq [N]$, $wt_T(\mathbf{u}) \leqslant wt(\mathbf{u})$. (5.18) follows by an argument similar to the one used to argue (5.9) from (5.8) in the proof of Theorem 5.1. Basically, we need to write out the probability as a product of conditional probabilities (with $j = J$ "taken out" first) and then using the facts that the tuple $(\mathbf{u}^1, \ldots, \mathbf{u}^J)$ is good and the choices for $\mathbf{G}_1, \ldots, \mathbf{G}_N$ are independent.[8] (5.19) follows from Lemma 5.1. (5.20) follows from rearranging the summand and using the fact that the tuple is good (and hence $d_j \geqslant (1 - R - \gamma)N$). (5.21) follows from the fact that there are q^{nN} choices[9] for $\mathbf{y}$ and Lemma 5.6. (5.22) follows from the fact that

[8] In Theorem 5.1, the tuple of codewords were not ordered while they are ordered here. However, it is easy to check that the argument in Theorem 5.1 also works for ordered tuples as long as the induction is applied in the right order.

[9] As the final code C will be linear, it is sufficient to only look at received words that have Hamming weight at most ρnN. However, this gives a negligible improvement to the final result and hence, we just bound the number of choices for $\mathbf{y}$ by q^{nN}.

$d_j - (1-R)N + 1 \leqslant d_j - (1-R-\gamma)N$ (for $N \geqslant 1/\gamma$) and that $d_j \geqslant (1-R-\gamma)N$. (5.23) follows by rearranging the terms.

Now, as long as $n \geqslant J(J+1)$, we have $\frac{N(J+1)}{nd} \leqslant \frac{N}{Jd}$. (5.23) will imply (5.15) (along with the fact that H_q^{-1} is increasing) if we can show that for every $(1-R-\gamma)N \leqslant d \leqslant N$,

$$\frac{h}{nd} \leqslant H_q^{-1}\left(1 - r\left(1 - \frac{(1-R-\gamma)N}{d}\right) - \frac{2N}{Jd}\right) - \delta,$$

for $\delta = \varepsilon/3$. Thus, by Lemma 5.4 (and using the arguments used in the proof of Theorem 5.1 to show that the conditions of Lemma 5.4 are satisfied), we can select J in $\Theta\left(\frac{1}{\varepsilon^2(1-R)}\right)$ (and γ in $\Theta(\varepsilon^2(1-R)/r)$), and pick

$$h/(nN) = H_q^{-1}(1-rR) - \varepsilon = \rho,$$

as desired. This along with Lemma 5.5, implies that we can set

$$L = (N/\varepsilon^2)^{O\left(\varepsilon^{-4}(1-R)^{-2}\log(q/R)\right)},$$

as required.

Using arguments similar to those in the proof of Theorem 5.1, one can show that the code $C_{out} \circ (C_{in}^1, \ldots, C_{in}^N)$ with high probability has rate rR.

Remark 5.2. *The idea of using list recoverability to argue independence can also be used to prove Theorem 5.1. That is, first show that with good probability, a random linear outer code will have good list recoverability. Then the argument in this section can be used to prove Theorem 5.1. However, this gives worse parameters than the proof presented in Section 5.4. In particular, by a straightforward application of the probabilistic method, one can show that a random linear code of rate R over $\mathbb{F}_Q$ is $(R+\gamma, \ell, Q^{\ell/\gamma})$-list recoverable [49, Sec 9.3.2]. In proof of Theorem 5.2, ℓ is roughly q^J, where J is roughly $1/\varepsilon^2$. Thus, if we used the arguments in the proof of Theorem 5.2, we would be able to prove Theorem 5.1 but with lists of size of $Q^{q^{O\left(\varepsilon^{-2}(1-R)^{-1}\right)}}$, which is worse than the list size of $Q^{O\left(\varepsilon^{-2}(1-R)^{-1}\right)}$ guaranteed by Theorem 5.1.*

5.6 Bibliographic Notes and Open Questions

The material presented in this chapter appears in [61].

Theorem 5.1 in some sense generalizes the following result of Blokh and Zyablov [19]. Blokh and Zyablov show that the concatenated code where both the outer and inner codes are chosen to be random linear codes with high probability lies on the Gilbert-Varshamov bound of relative minimum distance is (at least) $H_q^{-1}(1-R)$ for rate R.

The arguments used in this chapter also generalize Thommesen's proof that concatenated codes obtained by setting Reed-Solomon codes as outer codes and independent random inner code lie on the Gilbert-Varshamov bound [102]. In particular by using $\mathbf{y}$ to be

the all zero vector and $J = L = 1$ in proof of Theorem 5.2, one can recover Thommesen's proof. Note that when $J = 1$, a codeword $\mathbf{c}$ is $(\langle w \rangle, \mathbb{F}_q)$-independent if $wt(\mathbf{w}) = w$. Thus, the proof of Thommesen only required a knowledge of the weight distribution of the Reed-Solomon code. However, for our purposes, we need a stronger form of independence in the proof of Theorem 5.2 for which we used the strong list-recoverability property of folded Reed-Solomon codes.

Theorem 5.2 leads to the following intriguing possibility.

Open Question 5.1. *Can one list decode the concatenated codes from Theorem 5.2 up to the fraction of errors for which Theorem 5.2 guarantees it to be list decodable (with high probability)?*

Current list-decoding algorithms for concatenated codes work in two stages. In the first stage, the inner code(s) are list decoded and in the second stage the outer code is list recovered (for example see Chapter 4). In particular, the fact that in these algorithms the first phase is oblivious to the outer codes seems to be a bottleneck. Somehow "merging" the two stages might lead to a positive resolution of the question above.

Chapter 6

LIMITS TO LIST DECODING REED-SOLOMON CODES

6.1 Introduction

In Chapters 3 and 4 we were interested in the following question: Can one construct explicit codes along with efficient list-decoding algorithms that can correct errors up to the list-decoding capacity ? Note that in the question above, we have the freedom to pick the code. In this chapter, we will turn around the question by focusing on a fixed code and then asking what is the best possible tradeoff between rate and fraction of errors (that can be corrected via efficient list decoding) for the given code.

In this chapter, we will primarily focus on Reed-Solomon codes. Reed-Solomon codes are an important and extensively studied family of error-correcting codes. The codewords of a Reed-Solomon code (henceforth, RS code) over a field $\mathbb{F}$ are obtained by evaluating low degree polynomials at distinct elements of $\mathbb{F}$. The rate versus distance tradeoff for Reed-Solomon codes meets the Singleton bound, which along with the code's nice algebraic properties, give RS codes a prominent place in coding theory. As a result the problem of decoding RS codes has received much attention.

As we already saw in Section 3.1, in terms of fraction of errors corrected, the best known polynomial time list algorithm today can, for Reed-Solomon codes of rate R, correct up to a $1 - \sqrt{R}$ ([97, 63]) fraction of errors. The performance of the algorithm in [63] matches the so-called Johnson bound (cf. [64]) which gives a general lower bound on the number of errors one can correct using small lists in *any* code, as a function of the distance of the code. As we saw in Chapter 3, there are explicit codes known that have better list decodable properties than Reed-Solomon codes. However, Reed-Solomon codes have been instrumental in all the algorithmic progress in list decoding (see Section 3.1 for more details on these developments). In addition, Reed-Solomon codes have important practical applications. Thus, given the significance (both theoretical and practical) of Reed-Solomon codes, it is an important question to pin down the optimal tradeoff between the rate and list decodability of Reed-Solomon codes.

This chapter is motivated by the question of whether the Guruswami-Sudan result is the best possible (i.e., whether the Johnson bound is "tight" for Reed-Solomon codes). By this we mean whether attempting to decode with a larger error parameter might lead to super-polynomially large lists as output, which of course will preclude a polynomial time algorithm. While we don't quite show this to be the case, we give evidence in this direction by demonstrating that in the more general setting of list recovery (to which also the algorithm of Guruswami and Sudan [63] applies) its performance is indeed the best

possible.

We also present constructions of explicit "bad list-decoding configurations" for Reed-Solomon codes. The details follow.

6.2 Overview of the Results

6.2.1 Limitations to List Recovery

The algorithm in [63] in fact solves the following more general *polynomial reconstruction* problem in polynomial time: Given n' distinct pairs $(\beta_i, \gamma_i) \in \mathbb{F}^2$ output a list of all polynomials p of degree k that satisfy $p(\beta_i) = \gamma_i$ for more than $\sqrt{kn'}$ values of $i \in \{1, 2, \ldots, n'\}$ (we stress that the β_i's need **not** be distinct). In particular, the algorithm can solve the list recovery problem (see Definition 2.4). As a special case, it can solve the following "error-free" or "noiseless" version of the list recovery problem.

Definition 6.1 (Noiseless List Recovery). *For a q-ary code C of block length n, the noiseless list recovery problem is the following. We are given a set $S_i \subseteq \mathbb{F}_q$ of possible symbols for the i'th symbol for each position i, $1 \leqslant i \leqslant n$, and the goal is to output all codewords $c = \langle c_1, \ldots, c_n \rangle$ such that $c_i \in S_i$ for every i. When each S_i has at most ℓ elements, we refer to the problem as noiseless list recovery with input lists of size ℓ.*

Note that if a code C is $(0, \ell, L)$-list recoverable then L is the worst case output list size when one solves the noiseless list recovery problem on C with input lists of size ℓ.

Guruswami and Sudan algorithm [63] can solve the noiseless list recovery problem for Reed-Solomon codes with input lists of size $\ell < \lceil \frac{n}{k} \rceil$ in polynomial time. That is, Reed-Solomon codes are $(0, \lceil \frac{n}{k} \rceil - 1, L(n))$-list recoverable for some polynomially bounded function $L(n)$. In Section 6.3, we demonstrate that this latter performance is the best possible with surprising accuracy — specifically, we show that when $\ell = \lceil \frac{n}{k} \rceil$, there are settings of parameters for which the list of output polynomials needs to be super-polynomially large in n (Theorem 6.3). In fact, our result also applies to the model considered by Ar et al. [3], where the input lists are "mixtures of codewords." In particular, in their model the lists at every position take values from a collection of ℓ *fixed* codewords.

As a corollary, this rules out an efficient solution to the polynomial reconstruction algorithm that works even under the slightly weaker condition on the agreement parameter: $t > \sqrt{kn'} - k/2$.[1] In this respect, the "square root" bound achieved by [63] is optimal, and any improvement to their list-decoding algorithm which works with agreement fraction $t/n < \sqrt{R}$ where $R = (k+1)/n$ is the rate of the code, or in other words that works beyond the Johnson bound, must exploit the fact that the evaluation points β_i are distinct (or "almost distinct").

[1] This in turn rules out, for every $\varepsilon > 0$, a solution to the polynomial reconstruction algorithm that works as long as $t \geqslant \sqrt{(1-\varepsilon)kn'}$.

While this part on tightness of Johnson bound remains speculative at this stage, for the problem of list recovery itself, our work proves that RS codes are indeed sub-optimal, as we describe below. By our work Reed-Solomon codes for list recovery with input lists of size ℓ must have rate at most $1/\ell$. On the other hand, Guruswami and Indyk [52] prove that there exists a fixed $R > 0$ (in fact R can be close to 1) such that for every integer ℓ there are codes of rate R which are list recoverable given input lists of size ℓ (the alphabet size and output list size will necessarily grow with ℓ but the rate itself is independent of ℓ). Note that in Chapter 3, we showed that folded Reed-Solomon codes are explicit list recoverable codes with optimal rate.

6.2.2 Explicit "Bad" List Decoding Configurations

The result mentioned above presents an explicit bad list recovery configuration, i.e., an input instance to the list recovery problem with a super-polynomial number of solutions. To prove results on limitations of list decoding, such as the tightness of the Johnson bound, we need to demonstrate a received word $\mathbf{y}$ with super-polynomially many codewords that agree with $\mathbf{y}$ at t or more places. A simple counting argument establishes the *existence* of such received words that have agreement t with $\binom{n}{t}/q^{t-k}$ many codewords [70, 25]. In particular, this implies the following for $n = q$. For $k = n^\delta$ (in which case we say that the Reed-Solomon code has low rate), one can get $t = \frac{k}{2\delta}$ for any $\delta > 0$ and for k in $\Omega(n)$ (in which case we say that the Reed-Solomon code has high rate), one can get $t = k + O\left(\frac{n}{\log n}\right)$. In Section 6.4.2, we demonstrate an *explicit* construction of such a received word with super-polynomial number of codewords with agreement t up to $(2-\varepsilon)k$ (for any $\varepsilon > 0$), where $k = n^\delta$ for any $\delta > 0$. Note that such a construction is trivial for $t = k$ since we can interpolate degree k polynomials through any set of k points. In Section 6.4.3, we demonstrate an *explicit* construction of such a received word with super-polynomial number of codewords with agreement t up to $k + \frac{n}{\log^{\omega(1)} n}$, when k is in $\Omega(n)$.

In general, the quest for *explicit* constructions of this sort (namely small Hamming balls with several codewords) is well motivated. If achieved with appropriate parameters they will lead to a derandomization of the inapproximability result for computing the minimum distance of a linear code [32]. However, for this application it is important to get $2^{n^{\Omega(1)}}$ codewords in a ball of radius ρ times the distance of the code for some constant $\rho < 1$. Unfortunately, neither of our explicit constructions achieve ρ smaller than $1 - o(1)$.

As another motivation, we point out that the current *best* trade-off between rate and relative distance (for a code over constant sized alphabet) is achieved by a non-linear code comprising of precisely a bad list-decoding configuration in certain algebraic-geometric codes [107]. Unfortunately the associated received word is only shown to exist by a counting argument and its explicit specification will be required to get explicit codes with these parameters.

6.2.3 Proof Approach

We show our result on list recovering Reed-Solomon codes by proving a super-polynomial (in $n = q^m$) bound on the number of polynomials over $\mathbb{F}_{q^m}$ of degree k that take values in $\mathbb{F}_q$ at every point in $\mathbb{F}_{q^m}$, for any prime power q where k is roughly q^{m-1}. Note that this implies that there can be a super-polynomial number of solutions to list recovery when input list sizes are $\lceil \frac{n}{k} \rceil$. We establish this bound on the number of such polynomials by exploiting a folklore connection of such polynomials to a classic family of cyclic codes called BCH codes, followed by an (exact) estimation of the size of BCH codes with certain parameters. We also write down an explicit collection of polynomials, obtained by taking $\mathbb{F}_q$-linear combinations of translated norm functions, all of which take values only in $\mathbb{F}_q$. By the BCH bound, we conclude that this in fact is a precise description of the collection of all such polynomials.

Our explicit construction of a received word $\mathbf{y}$ with several RS codewords (for low rate RS codes) with non-trivial agreement with $\mathbf{y}$ is obtained using ideas from [25] relating to representations of elements in an extension finite field by products of distinct linear factors. Our explicit construction for high rate RS codes is obtained by looking at cosets of certain prime fields.

6.3 BCH Codes and List Recovering Reed-Solomon Codes

6.3.1 Main Result

We will work with polynomials over $\mathbb{F}_{q^m}$ of characteristic p where q is a power of p, and $m \geqslant 1$. Our goal in this section is to prove the following result, and in Section 6.3.2 we will use it to state corollaries on limits to list decodability of Reed-Solomon codes. (We will only need a lower bound on the number of polynomials with the stated property but the result below in fact gives an exact estimation, which in turn is used in Section 6.3.4 to give a precise characterization of the concerned polynomials.)

Theorem 6.1. *Let q be a prime power, and $m \geqslant 1$ be an integer. Then, the number of univariate polynomials in $\mathbb{F}_{q^m}[z]$ of degree at most $\frac{q^m-1}{q-1}$ which take values in $\mathbb{F}_q$ when evaluated at every point in $\mathbb{F}_{q^m}$ is exactly q^{2^m}. That is,*

$$\left| \{ P(z) \in \mathbb{F}_{q^m}[z] \mid \deg(P) \leqslant \frac{q^m - 1}{q-1} \text{ and } \forall \alpha \in \mathbb{F}_{q^m}, P(\alpha) \in \mathbb{F}_q \} \right| = q^{2^m}$$

In the rest of this section, we prove Theorem 6.1. The proof is based on a connection of polynomials with the stated property to a family of cyclic codes called BCH codes, followed by an estimation of the size (or dimension) of the associated BCH code. Now, the latter estimation itself uses basic algebra. In particular one can prove Theorem 6.1 using finite field theory and Fourier transform without resorting to coding terminology. However, the connection to BCH codes is well known and we use this body of prior work to modularize our presentation.

We begin with the definition of BCH codes[2]. We point the reader to [80], Ch. 7, Sec. 6, and Ch. 9, Secs. 1-3, for detailed background information on BCH codes.

Definition 6.2. *Let α be a primitive element of $\mathbb{F}_{q^m}$, and let $n = q^m - 1$. The BCH code* $\mathrm{BCH}_{q,m,d,\alpha}$ *of* designed distance d *is a linear code of block length n over $\mathbb{F}_q$ defined as:*

$$\mathrm{BCH}_{q,m,d,\alpha} = \{\langle c_0, c_1, \ldots, c_{n-1}\rangle \in \mathbb{F}_q^n \mid c(\alpha^i) = 0 \text{ for } i = 1, 2, \ldots, d-1, \text{ where } c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \in \mathbb{F}_q[x]\}.$$

We will omit one or more the subscripts in $\mathrm{BCH}_{q,m,d,\alpha}$ *for notational convenience when they are clear from the context.*

In our proof, we will use the following well-known result. For the sake of completeness, we present its proof here.

Lemma 6.1 (BCH codes are subfield subcodes of RS codes). *Let q be a prime power and $m \geqslant 1$ an integer. Let $n = q^m - 1$, d be an integer in the range $1 < d < n$, and α be a primitive element of $\mathbb{F}_{q^m}$. Then the set of codewords of* $\mathrm{BCH}_{q,m,d,\alpha}$ *maybe written as*

$$\{\langle P(\alpha^0), P(\alpha^1), \ldots, P(\alpha^{n-1})\rangle \in \mathbb{F}_q^n \mid P \in \mathbb{F}_{q^m}[z], \deg(P) \leqslant n-d, \text{ and } P(\gamma) \in \mathbb{F}_q \ \forall \gamma \in \mathbb{F}_{q^m}\}.$$

Proof. Our goal is to prove that the two sets

$$S_1 = \{\ \langle c_0, c_1, \ldots, c_{n-1}\rangle \mid c(\alpha^i) = 0 \text{ for } i = 1, 2, \ldots, d-1, \text{ where } c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \in \mathbb{F}_q[x]\ \},$$

$$S_2 = \left\{\langle P(\alpha^0), P(\alpha^1), \ldots, P(\alpha^{n-1})\rangle \mid P \in \mathbb{F}_{q^m}[z],\ \deg(P) \leqslant n-d,\ \text{and}\ P(\gamma) \in \mathbb{F}_q \ \forall \gamma \in \mathbb{F}_{q^m}\right\},$$

are identical. We will do so by showing both the inclusions $S_2 \subseteq S_1$ and $S_1 \subseteq S_2$.

We begin with showing $S_2 \subseteq S_1$. Let $P(z) = \sum_{j=0}^{n-d} a_j z^j \in \mathbb{F}_{q^m}[z]$ be a polynomial of degree at most $(n-d)$ that takes values in $\mathbb{F}_q$. Then, for $r = 1, 2, \ldots, d-1$, we have

$$\sum_{i=0}^{n-1} P(\alpha^i)(\alpha^r)^i = \sum_{i=0}^{n-1}\Big(\sum_{j=0}^{n-d} a_j \alpha^{ij}\Big)\alpha^{ri} = \sum_{j=0}^{n-d} a_j \sum_{i=0}^{n-1}(\alpha^{r+j})^i = 0\,,$$

where in the last step we use that $\sum_{i=0}^{n-1}\gamma^i = 0$ for every $\gamma \in \mathbb{F}_{q^m} \setminus \{1\}$ and $\alpha^{r+j} \neq 1$ since $1 \leqslant r + j \leqslant n-1$ and α is primitive. Therefore, $\langle P(\alpha^0), P(\alpha^1), \ldots, P(\alpha^{n-1})\rangle \in S_1$.

[2]What we define are actually referred to more specifically as *narrow-sense primitive* BCH codes, but we will just use the term BCH codes for them.

We next proceed to show the inclusion $S_1 \subseteq S_2$. Suppose $\langle c_0, c_1, \ldots, c_{n-1}\rangle \in S_1$. For $0 \leqslant j \leqslant n-1$, define (this is the "inverse Fourier transform")

$$a_j = \frac{1}{n}\sum_{i=0}^{n-1} c_i \alpha^{-ji} \ ,$$

where by $\frac{1}{n}$, we mean the multiplicative inverse of $n \cdot 1$ in the field $\mathbb{F}_{q^m}$. Note that $a_j = \frac{1}{n}c(\alpha^{-j}) = \frac{1}{n}c(\alpha^{n-j})$ where $c(x) = \sum_{i=0}^{n-1} c_i x^i$. So, by the definition of S_1, it follows that $a_j = 0$ for $j > n-d$. Therefore the polynomial $P(z) \in \mathbb{F}_{q^m}$ defined by

$$P(z) = \sum_{j=0}^{n-1} a_j z^j = \sum_{j=0}^{n-d} a_j z^j$$

has degree at most $(n-d)$.

We now claim that for $P(\alpha^s) = c_s$ for $0 \leqslant s \leqslant n-1$. Indeed,

$$\begin{aligned} P(\alpha^s) &= \sum_{j=0}^{n-1} a_j \alpha^{sj} = \sum_{j=0}^{n-1}\left(\frac{1}{n}\sum_{i=0}^{n-1} c_i \alpha^{-ji}\right)\alpha^{sj} \\ &= \sum_{i=0}^{n-1}\frac{c_i}{n}\sum_{j=0}^{n-1}(\alpha^{s-i})^j = c_s \ , \end{aligned}$$

where in the last step we used the fact that $\sum_{j=0}^{n-1}(\alpha^{s-i})^j = 0$ whenever $i \neq s$, and equals n when $i = s$. Therefore, $\langle c_0, c_1, \ldots, c_{n-1}\rangle = \langle P(\alpha^0), \ldots, P(\alpha^{n-1})\rangle$. We are pretty much done, except that we have to check also that $P(0) \in \mathbb{F}_q$ (since we wanted $P(\gamma) \in \mathbb{F}_q$ for all $\gamma \in \mathbb{F}_{q^m}$, including $\gamma = 0$). Note that $P(0) = a_0 = \frac{1}{n}\cdot\sum_{i=0}^{n-1} c_i$. Since $n = q^m - 1$, we have $n+1 = 0$ in $\mathbb{F}_{q^m}$ and so $\frac{1}{n} = -1 \in \mathbb{F}_q$. This together with the fact that $c_i \in \mathbb{F}_q$ for every i implies that $P(0) \in \mathbb{F}_q$ as well, completing the proof. □

In light of the above lemma, in order to prove Theorem 6.1, we have to prove that $|\mathrm{BCH}_{q,m,d,\alpha}| = q^{2^m}$ when $d = (q^m-1)(1-\frac{1}{q-1})$. We turn to this task next. We begin with the following bound on the size of BCH codes [15, Ch. 12]. For the sake of completeness, we also give a proof sketch.

Lemma 6.2 (Dimension of BCH Codes). *For integer i, n, let $\lfloor i \rfloor_n$ be a shorthand for $i \bmod n$. Then $|\mathrm{BCH}_{q,m,d,\alpha}| = q^{|I(q,m,d)|}$ where*

$$I(q,m,d) = \{i \mid 0 \leqslant i \leqslant n-1, \lfloor iq^j \rfloor_n \leqslant n-d \text{ for all } j, 0 \leqslant j \leqslant m-1\} \qquad (6.1)$$

for $n = q^m - 1$. (Note that for this value of n, if $i = i_0 + i_1 q + \cdots i_{m-1}q^{m-1}$, then $\lfloor iq \rfloor_n = i_{m-1} + i_0 q + i_1 q^2 + \cdots + i_{m-2}q^{m-1}$, and so $\lfloor iq \rfloor_n$ is obtained by a simple cyclic shift of the q-ary representation of i.)

Proof. It follows from Definition 6.2 that the BCH codewords are simply polynomials $c(x)$ over $\mathbb{F}_q$ of degree at most $(n-1)$ that vanish at α^i for $1 \leqslant i < d$. Note that if $c(x), c'(x)$ are two such polynomials, then so is $c(x)+c'(x)$. Moreover, since $\alpha^n = 1$, $xc(x) \mod (x^n-1)$ also vanishes at each designated α^i. It follows that if $c(x)$ is a codeword, then so is $r(x)c(x) \mod (x^n-1)$ for every polynomial $r(x) \in \mathbb{F}_q[x]$.

In other words $\mathrm{BCH}_{q,m,d}$ is an *ideal* in the quotient ring $R = \mathbb{F}_q[x]/(x^n-1)$. It is well known that R is a principal ideal ring, i.e., a ring in which every ideal is generated by one element [77, Chap. 1, Sec. 3]. Therefore there is a unique monic polynomial $g(x) \in \mathbb{F}_q[x]$ such that

$$\mathrm{BCH}_{q,m,d,\alpha} = \{g(x)h(x) \mid h(x) \in \mathbb{F}_q[x]; \deg(h) \leqslant n-1-\deg(g)\}$$

It follows that $|\mathrm{BCH}_{q,m,d,\alpha}| = q^{n-\deg(g)}$, and so it remains to prove that $\deg(g) = n - |I(q,m,d)|$ where $I(q,m,d)$ is defined as in (6.1).

It is easily argued that the polynomial $g(x)$ is the monic polynomial of lowest degree over $\mathbb{F}_q$ that has α^i for every i, $1 \leqslant i < d$, as roots. It is well known ([80, Chap. 7, Sec. 5]) that $g(x)$ is then given by

$$g(x) = \prod_{\beta \in M(\alpha) \cup M(\alpha^2) \cdots \cup M(\alpha^{d-1})} (x-\beta)\,,$$

where $M(\alpha^i)$ is the *cyclotomic coset*[3] of α^i. Further for the ease of notation, define $M_{d,\alpha} = M(\alpha) \cup M(\alpha^2) \cdots \cup M(\alpha^{d-1})$. To complete the proof we will show that

$$\mid M_{d,\alpha} \mid = n - \mid I(q,m,d) \mid . \tag{6.2}$$

To prove (6.2), we claim that for every $0 \leqslant i \leqslant n-1$, $\alpha^i \in M_{d,\alpha}$ if and only if $(n-i) \notin I(m,q,d)$. To see that this is true note that $(n-i) \notin I(q,m,d)$ if and only if there is a $0 \leqslant j_i < m$ such that $\lfloor (n-i)q^{j_i} \rfloor_n = n - i^* > n-d$. In other words, $\lfloor iq^{j_i} \rfloor_n = i^*$, where $0 \leqslant i^* < d$. This implies that $(n-i) \notin I(q,m,d)$ if and only if $\alpha^i \in M(\alpha^{i^*}) \subseteq M_{d,\alpha}$, which proves the claim. □

Let's now use the above to compute the size of $\mathrm{BCH}_{q,m,d,\alpha}$ where $d = (q^m-1) - \frac{q^m-1}{q-1}$. We need to compute the quantity $|I(q,m,d)|$, i.e., the number of i, $0 \leqslant i < q^m - 1$ such that $\lfloor iq^j \rfloor_{q^m-1} \leqslant \frac{q^m-1}{q-1} = 1 + q + \cdots + q^{m-1}$ for each $j = 0, 1, \ldots, m-1$. This condition is equivalent to saying that if $i = i_0 + i_1 q + \cdots + i_{m-1}q^{m-1}$ is the q-ary expansion of i, then all the m integers whose q-ary representations are cyclic shifts of $(i_0, i_1, \ldots, i_{m-1})$ are $\leqslant 1 + q + \cdots + q^{m-1}$. Clearly, this condition is satisfied if and only if for each $j = 0, 1, \ldots, m-1$, $i_j \in \{0, 1\}$. There are 2^m choices for i with this property, and hence we conclude $|I(q,m,d)| = 2^m$ when $d = (q^m-1) - \frac{q^m-1}{q-1}$.

[3] In other words $M(\alpha^i) = \{\alpha^i, \alpha^{\lfloor iq \rfloor_n}, \ldots, \alpha^{\lfloor iq^{m_i-1} \rfloor_n}\}$, where m_i is the smallest integer such that $\lfloor iq^{m_i} \rfloor_n = i$.

Together with Lemma 6.1, we conclude that the number of polynomials of degree at most $\frac{q^m-1}{q-1}$ over $\mathbb{F}_{q^m}$ which take on values only in $\mathbb{F}_q$ at every point in $\mathbb{F}_{q^m}$ is precisely q^{2^m}. This is exactly the claim of Theorem 6.1.

Before moving on to state implications of the above result for Reed-Solomon list decoding, we state the following variant of Theorem 6.1.

Theorem 6.2. *Let q be a prime power, and $m \geqslant 1$ be an integer. Then, for each s, $1 \leqslant s \leqslant m$, the number of univariate polynomials in $\mathbb{F}_{q^m}[z]$ of degree at most $\sum_{j=1}^{s} q^{m-j}$ which take values in $\mathbb{F}_q$ when evaluated at every point in $\mathbb{F}_{q^m}$ is at least $q^{\sum_{j=0}^{s}\binom{m}{j}}$. And the number of such polynomials of degree strictly less than q^{m-1} is exactly q (namely just the constant polynomials, so there are no polynomials with this property for degrees between 1 and $q^{m-1}-1$).*

Since the proof of the theorem above is similar to the proof of Theorem 6.1, we will just sketch it here. By Lemmas 6.1 and 6.2, to count the number of univariate polynomials in $\mathbb{F}_{q^m}[z]$ of degree at most $q^{m-1}+\cdots+q^{m-s}$ which take values in $\mathbb{F}_q$, we need to count the number of integers $i = i_0 + i_1 q + \cdots + i_{m-1}q^{m-1}$ such that all integers corresponding to cyclic shifts of $(i_0, \ldots, i_{m-1})$ are at most $q^{m-1}+\cdots+q^{m-s}$. It is easy to see all integers i such that $i_j \in \{0,1\}$ for all j and $i_j = 1$ for at most s values of j, satisfy the required condition. The number of such integers is $\sum_{j=0}^{s}\binom{m}{j}$, which implies the bound claimed in the theorem. The argument when degree is $< q^{m-1}$ is similar. In this case we have to count the number of integers $i_0 + i_1 q + \cdots + i_{m-1}q^{m-1}$ such that all integers corresponding to all cyclic shifts of $(i_0, \ldots, i_{m-1})$ is $< q^{m-1}$. Note that if $i_j \neq 0$ for some $0 \leqslant j \leqslant m-1$, then the $(m-1-j)$th shift with be at least q^{m-1}. Thus, only $i = 0$ satisfies the required condition, which implies claimed bound in the theorem.

6.3.2 Implications for Reed-Solomon List Decoding

In the result of Theorem 6.1, if we imagine keeping $q \geqslant 3$ fixed and letting m grow, then for the choice $n = q^m$ and $k = (q^m-1)/(q-1)$ (so that $\lceil \frac{n}{k} \rceil = q$), Theorem 6.1 immediately gives us the following "negative" result on polynomial reconstruction algorithms and Reed-Solomon list decoding.[4]

Theorem 6.3. *For every prime power $q \geqslant 3$, there exist infinitely many pairs of integers k, n such that $\lceil \frac{n}{k} \rceil = q$ for which there are Reed-Solomon codes of dimension $(k+1)$ and block length n, such that noiselessly list recovering them with input lists of size $\lceil \frac{n}{k} \rceil$ requires super-polynomial (in fact $q^{n^{1/\lg q}}$) output list size.*

The above result is exactly tight in the following sense. It is easy to argue combinatorially (via the "Johnson type" bounds, cf. [64]) that when $\ell < \lceil \frac{n}{k} \rceil$, the number of codewords

[4]We remark that we used the notation $n = q^m - 1$ in the previous subsection, but for this Subsection we will take $n = q^m$.

is polynomially bounded. Moreover [63] presents a polynomial time algorithm to recover all the solution codewords in this case. As was mentioned in the introduction, our results also show the tightness of noiselessly list recovering Reed-Solomon codes in the special setting of Ar, Lipton, Rubinfeld and Sudan [3]. One of the problems considered in [3] is that of noiselessly list recovering Reed-Solomon codes with list size ℓ, when the set S_i at every position i is the set of values of *fixed* ℓ codewords at position i. Note that our lower bound also works in this restricted model if one takes the q fixed codewords to be the q constant codewords.

The algorithm in [63] solves the more general problem of finding all polynomials of degree at most k which agree with at least t out of n' distinct pairs (β_i, γ_i) whenever $t > \sqrt{kn'}$. The following corollary states that, in light of Theorem 6.3, this is essentially the best possible trade-off one can hope for from such a general algorithm. We view this as providing the message that a list-decoding algorithm for Reed-Solomon codes that works with fractional agreement t/n that is less than $\sqrt{R}$ where R is the rate, must exploit the fact that the evaluation points β_i are distinct or almost distinct (by which we mean that no β_i is repeated too many times). Note that for small values of R (close to 0), our result covers even an improvement of the necessary fractional agreement by $O(R)$ which is substantially smaller than $\sqrt{R}$.

Corollary 6.4. *Suppose $\mathcal{A}$ is an algorithm that takes as input n' distinct pairs $(\beta_i, \gamma_i) \in \mathbb{F}^2$ for an arbitrary field $\mathbb{F}$ and outputs a list of all polynomials p of degree at most k for which $p(\beta_i) = \gamma_i$ for more than $\sqrt{kn'} - \frac{k}{2}$ pairs. Then, there exist inputs under which $\mathcal{A}$ must output a list of super-polynomial size.*

Proof. Note that in the list recovery setting of Theorem 6.3, the total number of pairs $n' = n\ell = n\lceil \frac{n}{k} \rceil < n(\frac{n}{k} + 1)$, and the agreement parameter $t = n$. Then

$$\sqrt{kn'} - \frac{k}{2} < \sqrt{kn\Big(\frac{n}{k} + 1\Big)} - \frac{k}{2} = n\sqrt{1 + \frac{k}{n}} - \frac{k}{2}$$

$$\leqslant n\Big(1 + \frac{k}{2n}\Big) - \frac{k}{2} = n = t\ .$$

Therefore there can be super-polynomially many candidate polynomials to output even when the agreement parameter t satisfies $t > \sqrt{kn'} - k/2$. □

6.3.3 Implications for List Recovering Folded Reed-Solomon Codes

In this subsection, we will digress a bit and see what the ideas in Section 6.3.1 imply about list recoverability of folded Reed-Solomon codes. Recall that a folded Reed-Solomon code with folding parameter m is just a Reed-Solomon code with m consecutive evaluation pointsbundled together (see Chapter 3). In particular, if we start with an $[n, k]$ Reed-Solomon code, then we get an $(N = n/m, K = k/m)$ folded Reed-Solomon code.

It is not too hard to check that the one can generalize Theorem 6.1 to show the following. Let $a \geqslant 1$ be an integer and q be a prime power. Then there are q^{2^a} codewords from an $\left(\frac{q^a}{m}, \frac{q^a-1}{m(q-1)}\right)$ folded Reed-Solomon code such that every symbol of such a codeword takes a value in $(\mathbb{F}_q)^m$. The set of q^{2^a} folded Reed-Solomon codewords are just the q^{2^a} BCH codewords from Theorem 6.1, with m consecutive positions in the BCH codeword "folded" into one symbol. Thus, this shows that an (N, K) folded Reed-Solomon code (with folding parameter m) cannot be noiselessly list recovered with input lists of size $\left(\frac{N}{K}\right)^m$.

Let us now recall the algorithmic results for (noiselessly) list recovering folded Reed-Solomon codes. From (3.6) it follows that an (N, K) folded Reed-Solomon code (with folding parameter m) can be noiselessly list recovered with input lists of size ℓ if

$$N \geqslant \left(1 + \frac{s}{r}\right) \left(\frac{m}{m-s+1}\right) \sqrt[s+1]{NK^s\ell},$$

where $1 \leqslant s \leqslant m$, and $r \geqslant s$ are parameters that we can choose. Thus for any $\varepsilon > 0$ if $r = \frac{s}{\varepsilon}$, then we can satisfy the above condition if

$$\ell \leqslant (1-\varepsilon)^{s+1} \left(\frac{N}{K}\right)^s \left(\frac{m-s+1}{m}\right)^{s+1}. \tag{6.3}$$

The bound above unfortunately is much smaller than the bound of $(N/K)^m$, unlike the case of Reed-Solomon codes where the two bounds were (surprisingly) tight. For the case when $K = o(N)$, however one can show that for any $\delta > 0$, the bound in (6.3) is at least $(N/K)^{m(1-\delta)}$. Indeed, one can choose $s = m(1-\delta/2)$, in which case the bound in (6.3) is $(N/K)^{m(1-\delta)} \cdot (N/K)^{m\delta/2}(\delta/2)^{m(1-\delta/2)+1}(1-\varepsilon)^{m(1-\delta/2)+1}$. The claimed expression follows by noting that $N/K = \omega(1)$ while δ, ε and m are all $O(1)$.

6.3.4 A Precise Description of Polynomials with Values in Base Field

We proved in Section 6.3.1, for $Q = \frac{q^m-1}{q-1}$, there are exactly q^{2^m} polynomials over $\mathbb{F}_{q^m}$ of degree Q or less that evaluate to a value in $\mathbb{F}_q$ at every point in $\mathbb{F}_{q^m}$. The proof of this obtains the coefficients of such polynomials using a "Fourier transform" of codewords of an associated BCH code, and as such gives little insight into the structure of these polynomials. One of the natural questions to ask is: Can we say something more concrete about the structure of these q^{2^m} polynomials? In this section, we answer this question by giving an exact description of the set of all these q^{2^m} polynomials.

We begin with the following well-known fact which simply states that the "Norm" function of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ takes only values in $\mathbb{F}_q$.

Lemma 6.3. *For all $x \in \mathbb{F}_{q^m}$, $x^{\frac{q^m-1}{q-1}} \in \mathbb{F}_q$.*

Theorem 6.5. *Let q be a prime power, and let $m \geqslant 1$. Let α be a primitive element of $\mathbb{F}_{q^m}$. Then, there are exactly q^{2^m} univariate polynomials in $\mathbb{F}_{q^m}[z]$ of degree at most $Q = \frac{q^m-1}{q-1}$*

that take values in $\mathbb{F}_q$ when evaluated at every point in $\mathbb{F}_{q^m}$, and these are precisely the polynomials in the set

$$N = \{ \sum_{i=0}^{2^m-1} \beta_i (z+\alpha^i)^Q \ \mid \ \beta_0, \beta_1, \ldots, \beta_{2^m-1} \in \mathbb{F}_q \} .$$

Proof. By Lemma 6.3, clearly every polynomial P in the set N satisfies $P(\gamma) \in \mathbb{F}_q$ for all $\gamma \in \mathbb{F}_{q^m}$. The claim that there are exactly q^{2^m} polynomials over $\mathbb{F}_{q^m}$ of degree Q or less that take values only in $\mathbb{F}_q$ was already established in Theorem 6.1. So the claimed result that N precisely describes the set of all these polynomials follows if we show that $|N| = q^{2^m}$.

Note that by definition, $|N| \leqslant q^{2^m}$. To show that $|N| \geqslant q^{2^m}$, it clearly suffices to show (by linearity) that if

$$\sum_{i=0}^{2^m-1} \beta_i (z+\alpha^i)^Q = 0 \tag{6.4}$$

as polynomials in $\mathbb{F}_{q^m}[z]$, then $\beta_0 = \beta_1 = \cdots = \beta_{2^m-1} = 0$. We will prove this by setting up a full rank homogeneous linear system of equations that the β_i's must satisfy. For this we need Lucas' theorem, stated below.

Lemma 6.4 (Lucas' Theorem, cf. [47]). *Let p be a prime. Let a and b be positive integers with p-ary expansions $a_0 + a_1 p + \cdots + a_r p^r$ and $b_0 + b_1 p + \cdots + b_r p^r$ respectively. Then $\binom{a}{b} = \binom{a_0}{b_0}\binom{a_1}{b_1}\cdots\binom{a_r}{b_r} \mod p$, which gives us $\binom{a}{b} \neq 0 \mod p$ if and only if $a_j \geqslant b_j$ for all $j \in \{0, 1, \cdots, r\}$.*

Define the set

$$T = \{ \sum_{j \in S} q^j \ \mid \ S \subseteq \{0, \cdots, m-1\} \} .$$

Applying Lemma 6.4 with p being the characteristic of the field $\mathbb{F}_q$, we note that when operating in the field $\mathbb{F}_{q^m}$, the binomial coefficient of z^j in the expansion of $(z+\alpha^i)^Q$ is 1 if $j \in T$ and 0 otherwise. It follows that (6.4) holds if and only if $\sum_{i=0}^{2^m-1} (\alpha^i)^{Q-j} \beta_i = 0$ for all $j \in T$, which by the definition of T and the fact that $Q = 1 + q + q^2 + \cdots + q^{m-1}$ is equivalent to

$$\sum_{i=0}^{2^m-1} (\alpha^j)^i \beta_i = 0 \ \text{ for all } j \in T. \tag{6.5}$$

Let us label the 2^m elements $\{\alpha^j \mid j \in T\}$ as $\alpha_0, \alpha_1, \ldots, \alpha_{2^m-1}$ (note that these are *distinct* elements of $\mathbb{F}_{q^m}$ since α is primitive in $\mathbb{F}_{q^m}$). The coefficient matrix of the homogeneous system of equations (6.5) with unknowns $\beta_0, \ldots, \beta_{2^m-1}$ is then the Vandermonde matrix

$$\begin{pmatrix} 1 & \alpha_0 & \alpha_0^2 & \cdots & \alpha_0^{2^m-1} \\ 1 & \alpha_1 & \alpha_1^2 & \cdots & \alpha_1^{2^m-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha_{2^m-1} & \alpha_{2^m-1}^2 & \cdots & \alpha_{2^m-1}^{2^m-1} \end{pmatrix},$$

which has full rank. Therefore, the only solution to the system (6.5) is $\beta_0 = \beta_1 = \cdots = \beta_{2^m-1} = 0$, as desired. □

6.3.5 Some Further Facts on BCH Codes

The results in the previous subsections show that a large number (q^{2^m}) of polynomials over $\mathbb{F}_{q^m}$ take on values in $\mathbb{F}_q$ at every evaluation point, and this proved the tightness of the "square-root" bound for agreement $t = n = q^m$ and total number of points $n' = nq$ (recall Corollary 6.4). It is a natural question whether similarly large list size can be shown at other points (t, n'), specifically for slightly smaller n' and t. For example, what if $n' = n(q-1)$ and we consider list recovery from lists of size $q-1$. In particular, how many polynomials of degree at most $Q = (q^m - 1)/(q-1)$ take on values in $\mathbb{F}_q \setminus \{0\}$ at t points in $\mathbb{F}_{q^m}$. It is easily seen that when $t = n = q^m$, there are precisely $(q-1)$ such polynomials, namely the constant polynomials that equal an element of $\mathbb{F}_q^*$. Indeed, by the Johnson bound, since $t > \sqrt{Qn'}$ for the choice $t = n$ and $n' = n(q-1)$, we should not expect a large list size. However, even for the slightly smaller amount of agreement $t = n - 1 = \lfloor \sqrt{Qn'} \rfloor$, there are only about a linear in n number of codewords, as Lemma 6.5 below shows. Hence obtaining super-polynomial number of codewords at other points on the square-root bound when the agreement t is less than the block length remains an interesting question, which perhaps the BCH code connection just by itself cannot resolve.

Lemma 6.5. *Let q be a prime power and let $m > 1$. For any polynomial $P(z)$ over $\mathbb{F}_{q^m}[z]$, let its Hamming weight be defined as $|\{\beta \in \mathbb{F}_{q^m} | P(\beta) \neq 0\}|$. Then, there are exactly $(q-1)q^m$ univariate polynomials in $\mathbb{F}_{q^m}[z]$ of degree at most $Q = \frac{(q^m-1)}{q-1}$ that take values in $\mathbb{F}_q$ when evaluated at every point in $\mathbb{F}_{q^m}$ and that have Hamming weight $(q^m - 1)$. Furthermore, these are precisely the polynomials in the set $W = \{\lambda(z+\beta)^Q \mid \beta \in \mathbb{F}_{q^m}, \lambda \in \mathbb{F}_q^*\}$.*

Proof. It is obvious that all the polynomials in W satisfy the required property and are distinct polynomials. We next show that any polynomial of degree at most Q that satisfies the required properties belongs to W completing the proof.

Let $P(z)$ be a polynomial of degree at most Q that satisfies the required properties. We must show that $P(z) \in W$. Let $\gamma \in \mathbb{F}_{q^m}$ be such that $P(\gamma) = 0$. Clearly, for each $\beta \in (\mathbb{F}_{q^m} \setminus \{\gamma\})$, $P(\beta)/(\beta - \gamma)^Q \in \mathbb{F}_q^*$. By a pigeonhole argument, there must exist some $\lambda \in \mathbb{F}_q^*$ such that $P(\beta) = \lambda(\beta - \gamma)^Q$ for at least $\frac{q^m-1}{q-1} = Q$ values of β in $\mathbb{F}_{q^m} \setminus \{\gamma\}$. Since $P(\gamma) = 0$, we have that the degree Q polynomials $P(z)$ and $\lambda(z-\gamma)^Q$ agree on at least $Q+1$ field elements, which means that they must be equal to each other. Thus the polynomial $P(z)$ belongs to W and the proof is complete. □

6.4 Explicit Hamming Balls with Several Reed-Solomon Codewords

Throughout this section, we will be concerned with an $[q, k+1]$ Reed-Solomon code $\mathrm{RS}[q, k+1]$ over $\mathbb{F}_q$. We will be interested in a received word $\mathbf{y} \in \mathbb{F}_q^q$ such that a super-polynomial number of codewords of $\mathrm{RS}[q, k+1]$ agree with $\mathbf{y}$ on t or more positions, and the aim would be to prove such a result for t non-trivially larger than k. We start with the existential result.

6.4.1 Existence of Bad List Decoding Configurations

It is easy to prove the *existence* of a received word $\mathbf{y}$ with at least $\binom{q}{t}/q^{t-k}$ codewords with agreement at least t with $\mathbf{y}$. One way to see this is that this quantity is the expected number of such codewords for a received word that is the evaluation of a *random* polynomial of degree t [70].[5]

We have the following lower bound on $\binom{q}{t}/q^{t-k}$:

$$\frac{\binom{q}{t}}{q^{t-k}} \geqslant \frac{q^t}{t^t q^{t-k}} = \frac{q^k}{t^t} = 2^{k \log q - t \log t}.$$

Now when $k = q^\delta$ for some $\delta > 0$ and $t = \frac{q^\delta}{2\delta}$, then $k \log q - t \log t$ is $\Omega(q^\delta \log q)$, which implies that the number of RS codewords with agreement t with the received word $\mathbf{r}$ is $q^{\Omega(q^\delta)}$.

On the other hand, if $k = \Omega(q)$ let $t = k + \Delta$, where $\Delta = \frac{k}{2 \log q}$ (we also assume $t \leqslant q/2$). Now, $k \log q - t \log t \geqslant k \log q - (k+\Delta)(\log q - 1) = k + \Delta - \Delta \log q \geqslant k/2$. Thus, we get $2^{\Omega(q)}$ RS codewords with agreement $t = k + O\left(\frac{q}{\log q}\right)$ with the received word $\mathbf{r}$.

In the remainder of the chapter, we will try to match these parameters with *explicit* received words. We will refer to Reed-Solomon codes with constant rate as *high* rate Reed-Solomon codes and to Reed-Solomon codes with inverse polynomial rate as *low* rate Reed-Solomon codes.

6.4.2 Low Rate Reed-Solomon Codes

Another argument for the existence of a bad list-decoding configuration (from the previous subsection), as suggested in [25], is based on an element β in $\mathbb{F}_{q^h} = \mathbb{F}_q(\alpha)$, for some positive integer h, that can be written as a product $\prod_{a \in T}(\alpha + a)$ for at least $\binom{q}{t}/q^h$ subsets $T \subset \mathbb{F}_q$ with $|T| = t$ — the *existence* of such a β again follows by a trivial counting argument. Here we use the result due to Cheng and Wan [25] that for certain settings of parameters and fields such a β can be explicitly specified with only a slight loss in the number of subsets T, and thereby get an *explicit received word* $\mathbf{y}$ with several close-by codewords from $\mathrm{RS}[q, k+1]$.

[5] The bound can be improved slightly to $\binom{q}{t}/q^{t-1-k}$ by using a random *monic* polynomial.

Theorem 6.6 ([25]). *Let $\varepsilon > 0$ be arbitrary. Let q be a prime power, h be a positive integer and α be such that $\mathbb{F}_q(\alpha) = \mathbb{F}_{q^h}$. For any $\beta \in \mathbb{F}_{q^h}^*$, let $N_t(\beta)$ denote the number of t-tuples $\langle a_1, a_2, \ldots, a_t\rangle$ of distinct $a_i \in \mathbb{F}_q$ such that $\beta = \prod_{i=1}^{t}(\alpha + a_i)$. If $t \geqslant (\frac{4}{\varepsilon} + 2)(h+1)$, $\varepsilon < t - 2$ and $q \geqslant \max(t^2, (h-1)^{\frac{(2+\varepsilon)t}{t-(2+\varepsilon)}})$, then for all $\beta \in \mathbb{F}_{q^h}^*$, $N_t(\beta) > (t-1)q^{t-h-1}$.*

Proof. From the proof of Theorem 3 in [25], we obtain $N_t(\beta) \geqslant E_1 - E_2$, where $E_1 = \frac{q^t - \binom{t}{2}q^{t-1}}{q^{h-1}}$ and $E_2 = (1 + \binom{t}{2})(h-1)^t q^{\frac{t}{2}}$. Observe that from the choice of q, $\binom{t}{2} = \frac{t^2}{2} - \frac{t}{2} \leqslant \frac{q-t}{2}$.

We first give a lower bound on E_1. Indeed, using $\binom{t}{2} \leqslant \frac{q-t}{2}$ and $q^h - 1 < q^h$, we have $E_1 > \frac{2q^t - (q-t)q^{t-1}}{2q^h} = \frac{q^{t-h}}{2} + \frac{t}{2}q^{t-h-1}$.

Note that from our choice of t, we have $t > (\frac{4}{\varepsilon} + 2)h$, that is, $t - h > (\frac{4+\varepsilon}{4+2\varepsilon})t$. Further, from our choice of q, $(h-1)^t \leqslant q^{\frac{t}{2+\varepsilon}-1}$. We now bound E_2 from above. From our bounds on $\binom{t}{2}$ and $(h-1)^t$, we have $E_2 \leqslant (1 + \frac{q-t}{2})q^{(\frac{4+\varepsilon}{4+2\varepsilon})t-1} < (1 + \frac{q-t}{2})q^{t-h-1} = \frac{q^{t-h}}{2} - (\frac{t}{2} - 1)q^{t-h-1}$, where the second inequality comes from our bound on $t - h$.

Combining the bounds on E_1 and E_2 proves the theorem. □

We now state the main result of this section concerning Reed-Solomon codes:

Theorem 6.7. *Let $\varepsilon > 0$ be arbitrary real, q a prime power, and h any positive integer. If $t \geqslant (\frac{4}{\varepsilon} + 2)(h+1)$ and $q \geqslant \max(t^2, (h-1)^{\frac{(2+\varepsilon)t}{t-(2+\varepsilon)}})$ then for every k in the range $t - h \leqslant k \leqslant t - 1$, there exists an explicit received word $\mathbf{y} \in \mathbb{F}_q^q$ such that there are at least $\frac{q^k}{t!\binom{k+h}{t}}$ codewords of $\mathrm{RS}[q, k+1]$ that agree with $\mathbf{y}$ in at least t positions.*

We will prove the above theorem at the end of this section. As $\varepsilon \to 0$, and $q, k, h \to \infty$ in the above, we can get super-polynomially many codewords with agreement $(1+\delta)k$ for some $\delta = \delta(\varepsilon) > 0$ for a Reed-Solomon code of dimension tending to $q^{1/2}$. As $\varepsilon \to \infty$, we can get super-polynomially many codewords with agreement tending to $2k$ with dimension still being $q^{\Omega(1)}$. We record these as two corollaries below (for the sake of completeness, we sketch the proofs). We note that the non-explicit bound $\binom{q}{t}/q^{t-k}$ gives a super-polynomial number of codewords for agreement $t \geqslant k/\delta$ for dimension about $k = q^{\delta - o(1)}$, where as our explicit construction can give agreement at most $2k$ (or dimension at most $\sqrt{q}$).

Corollary 6.8. *For all $0 < \gamma < 1$, and primes p, there exists $\delta > 0$ such that for any power of p (call it q) that is large enough, there exists an explicit $\mathbf{y} \in \mathbb{F}_q^q$ such that the Reed-Solomon code $\mathrm{RS}[q, k+1 = q^\delta + 1]$ contains a super-polynomial (in q) number of codewords with agreement at least $(2-\gamma)k$ with $\mathbf{y}$.*

Proof. For any integer h, choose ε, t and k such that $t = (\frac{4}{\varepsilon} + 2)(h+1)$, $k = t - h + 1$ and $t = (2-\gamma)k$. These relations imply that

$$\varepsilon = \frac{4}{(\frac{2-\gamma}{1-\gamma})(\frac{h-1}{h+1}) - 2}.$$

Note that in the limit as h goes to infinity, $\varepsilon = \frac{4(1-\gamma)}{\gamma}$. Further, choose q to be a prime power such that $pq_0 \geqslant q \geqslant q_0$, where $q_0 = (h-1)^{\frac{2+\varepsilon}{1-(2+\varepsilon)/t}}$. Finally note that as t goes to infinity, $q_0 = (h-1)^{\frac{2(2-\gamma)}{\gamma}}$. For the rest of the proof we will assume that h is large enough so that $\varepsilon \simeq \frac{4(1-\gamma)}{\gamma}$, $q_0 \simeq (h-1)^{\frac{2(2-\gamma)}{\gamma}}$ and $(h-1)^{\frac{2(2-\gamma)}{\gamma}} \geqslant t^2$. Note that now $\delta = \log_q(h-1) - \log_q(1-\gamma) \geqslant \frac{2(2-\gamma)}{\gamma} - \log_q p - \log_q(1-\gamma) > 0$. As all conditions of Theorem 6.7 are satisfied, we have that the relevant number of codewords is at least $\mathcal{B} = \frac{q^k}{(t+1)!}$. Now as $t \simeq \left(\frac{2-\gamma}{1-\gamma}\right)(h+1)$ and h is large enough, we can assume that $t \leqslant \left(\frac{2-\gamma}{1-\gamma}\right)(2h)$. Thus, $t^t \leqslant (2h)^{(\frac{2-\gamma}{1-\gamma})(2h)} \cdot (\frac{2-\gamma}{1-\gamma})^{(\frac{2-\gamma}{1-\gamma})(2h)}$. To finish the proof we will show that $\mathcal{B} \geqslant \frac{q^{ch}}{2^{dh}}$ where c and d are constants which depend on γ. Indeed as $(t+1)! \leqslant t^t$, and $k \geqslant h$, we have

$$\frac{q^k}{(t+1)!} \geqslant \frac{q^h}{(2h)^{(\frac{2-\gamma}{1-\gamma})(2h)} \cdot (\frac{2-\gamma}{1-\gamma})^{(\frac{2-\gamma}{1-\gamma})(2h)}}.$$

Since h is large enough, $q \geqslant (h/2)^{\frac{2(2-\gamma)}{\gamma}}$, which along with the above inequality implies that

$$\mathcal{B} \geqslant \frac{h^{h\left(\frac{2(2-\gamma)}{\gamma}\right)}}{h^{(\frac{2-\gamma}{1-\gamma})(2h)}} \cdot \frac{1}{2^{\frac{2(2-\gamma)h}{\gamma}} 2^{(\frac{2-\gamma}{1-\gamma})(2h)} (\frac{2-\gamma}{1-\gamma})^{(\frac{2-\gamma}{1-\gamma})(2h)}} \geqslant \frac{h^{h\left(\frac{2(2-\gamma)}{\gamma}\right)\left(1-\frac{\gamma}{1-\gamma}\right)}}{2^{dh}},$$

where d is chosen such that $2^{dh} \geqslant 2^{\frac{2(2-\gamma)h}{\gamma}} 2^{(\frac{2-\gamma}{1-\gamma})(2h)} (\frac{2-\gamma}{1-\gamma})^{(\frac{2-\gamma}{1-\gamma})(2h)}$. Note that such d exists and it only depends on γ. Finally, if $\gamma < 1/2$, then there exists a value c that depends only on γ such that $h^{h\left(\frac{2(2-\gamma)}{\gamma}\right)\left(1-\frac{\gamma}{1-\gamma}\right)} \geqslant q^{ch}$. Thus, we have proved the theorem for $0 < \gamma < 1/2$. Since having an agreement of $2-\gamma$ implies an agreement of $2-\gamma'$ for any $\gamma' \geqslant \gamma$, the proof of the theorem for $0 < \gamma < 1$ follows. □

Corollary 6.9. *For all $0 < \gamma < \frac{1}{2}$ and primes p, there exists $\delta > 0$, such that for any power of p (call it q) that is large enough, there is an explicit $\mathbf{y} \in \mathbb{F}_q^q$ such that the Reed-Solomon code $\mathrm{RS}[q, k+1 = q^{1/2-\gamma}+1]$ contains a super-polynomial (in q) number of codewords with agreement at least $(1+\delta)k$ with $\mathbf{y}$.*

Proof. The proof is similar to the proof of Corollary 6.8 and hence, most details are skipped. Choose t and k such that $t = (\frac{4}{\varepsilon}+3)(h-1)$ and $k = t-h+1$. Note that for $h > \frac{8}{\varepsilon}+5$, $t > (\frac{4}{\varepsilon}+2)(h+1)$. Also let q be a prime power such that $q_0 \leqslant q \leqslant pq_0$, where $q_0 = (h-1)^{\frac{2+\varepsilon}{1-(2+\varepsilon)/t}}$. As in Corollary 6.8, we consider h to be very large and we have $q_0 \simeq (h-1)^{\frac{2}{1-2\gamma}}$, $t \simeq \frac{1+\gamma}{\gamma}(h-1)$ and $k \simeq \frac{h-1}{\gamma}$. Recalling that $t = (1+\delta)k$, we have $\delta \simeq \gamma$. Again using arguments as in the proof of Corollary 6.8, we have a lower bound of $\Omega(\frac{q^h}{2^{dh}})$ where d is a constant which depends on γ. □

(**Proof of Theorem 6.7**). In what follows, we fix $E(x)$ to be a polynomial of degree h that is irreducible over $\mathbb{F}_q$. For the rest of this proof we will denote $\mathbb{F}_q[x]/(E(x))$ by $\mathbb{F}_{q^h}$. Also note that for any root α of E, $\mathbb{F}_q(\alpha) = \mathbb{F}_{q^h}$.

Pick any ℓ where $0 \leqslant \ell \leqslant h-1$ and note that q and t satisfy the conditions of Theorem 6.6. For any $B = (b_0, b_1, \cdots, b_\ell)$, where $b_i \in \mathbb{F}_q$ with at least one non zero b_j; define $L_B(x) \stackrel{\text{def}}{=} \sum_{i=0}^{\ell} b_i x^i$. Fix $r(x)$ to be an arbitrary non-zero polynomial of degree at most $h-1$. By their definitions, $r(\alpha)$ and $L_B(\alpha)$ are elements of $\mathbb{F}^*_{q^h}$.

We will set the received word $\mathbf{y}$ to be $\langle \frac{r(a)}{E(a)} \rangle_{a \in \mathbb{F}_q}$. Note that since $E(x)$ is an irreducible polynomial, $E(a) \neq 0$ for all $a \in \mathbb{F}_q$, and $\mathbf{y}$ is a well-defined element of $\mathbb{F}_q^q$.

We now proceed to bound from below the number of polynomials of degree $k \stackrel{\text{def}}{=} t + \ell - h$ that agree with $\mathbf{y}$ on t positions. For each non-zero tuple $B \in \mathbb{F}_q^{\ell+1}$, define $Q_B(x) = -\frac{r(x)}{L_B(x)}$. Clearly, $Q_B(\alpha) \in \mathbb{F}^*_{q^h}$. For notational convenience we will use N_B to denote $N_t(Q_B(\alpha))$. Then, for $j = 1, \cdots, N_B$ there exist $\mathcal{A}_{(B,j)}$ where $\mathcal{A}_{(B,j)} \subset \mathbb{F}_q$ and $|\mathcal{A}_{(B,j)}| = t$ such that $P_B^{(j)}(\alpha) \stackrel{\text{def}}{=} \prod_{a \in \mathcal{A}_{(B,j)}} (\alpha + a) = Q_B(\alpha)$. By Theorem 6.6, we have $N_B \geqslant (t-1)q^{t-h-1}$ for every B — let us denote by N this latter quantity. Recalling the definition of Q_B, we have that for any (B, j), $\frac{r(\alpha)}{L_B(\alpha)} = -P_B^{(j)}(\alpha)$, or equivalently $r(\alpha) + P_B^{(j)}(\alpha) L_B(\alpha) = 0$. Since E is the irreducible polynomial of α over $\mathbb{F}_q$, this implies that $E(x)$ divides $P_B^{(j)}(x) L_B(x) + r(x)$ in $\mathbb{F}_q[x]$.

Finally we define $T_B^{(j)}(x)$ to be a polynomial of degree $k = t + \ell - h$ such that

$$T_B^{(j)}(x) E(x) = P_B^{(j)}(x) L_B(x) + r(x). \tag{6.6}$$

Clearly $T_B^{(j)}(-a)$ equals $r(-a)/E(-a)$ for each $a \in \mathcal{A}_{(B,j)}$ and thus the polynomial $T_B^{(j)}$ agrees with $\mathbf{y}$ on at least t positions. To complete the proof we will give a lower bound on the number of *distinct* polynomials in the collection $\{T_B^{(j)}\}$. For a fixed B, out of the N_B choices for $P_B^{(j)}$, $t!$ choices of j would lead to the same[6] polynomial of degree t. Since $N_B \geqslant N$, there are at least $\frac{(q^{\ell+1}-1)N}{t!}$ choices of pairs (B, j). Clearly for $j_1 \neq j_2$ the polynomials $P_B^{(j_1)}(x)$ and $P_B^{(j_2)}(x)$ are distinct, however we could have $P_{B_1}^{(j_1)}(x) L_{B_1}(x) = P_{B_2}^{(j_2)}(x) L_{B_2}(x)$ (both are equal to say $S(x)$) leading to $T_{B_1}^{(j_1)}(x) = T_{B_2}^{(j_2)}(x)$. However the degree of S is at most $t + \ell = k + h$, and hence S can have at most $k + h$ roots, and therefore at most $\binom{k+h}{t}$ factors of the form $\prod_{a \in T}(x + a)$ with $|T| = t$. It follows that no single degree k polynomial is counted more than $\binom{k+h}{t}$ times in the collection $\{T_B^{(j)}\}$, and hence there must be at least

$$\frac{(q^{\ell+1}-1)N}{t!\binom{k+h}{t}} \geqslant \frac{q^k}{t!\binom{k+h}{t}}$$

distinct polynomials among them, where we used $N = (t-1)q^{t-h-1}$ and $(q^{\ell+1}-1)(t-1) \geqslant q^{\ell+1} = q^{k-t+h+1}$ since $k = t + \ell - h$. $\square$

---

[6] If $\langle a_1, \cdots, a_t \rangle$ is a solution of the equation $\beta = \prod_{i=1}^{t}(\alpha + a_i)$ then so is $\langle a_{\sigma(1)}, \cdots, a_{\sigma(t)} \rangle$ for any permutation σ on $\{1, \cdots, t\}$.

6.4.3 High Rate Reed-Solomon Codes

We now consider the case of constant rate Reed-Solomon codes. We start with the main result of this subsection.

Theorem 6.10. *Let $L \geqslant 2$ be an integer. Let $p = aL + 1$ be a prime and define $t = bL$ for any $1 < b < a - 1$. Let the received word $\mathbf{r}$ be the evaluation of $R(X) = X^t$ over $\mathbb{F}_p^*$. Then there are $\binom{a}{b}$ many codewords in $RS[n = p, k = (b-1)L + 1]_{\mathbb{F}_p}$ that agree with $\mathbf{r}$ in at least t places.*

To get some interesting numbers, let's instantiate the parameters in the above theorem. First we need the following result (we will prove this later in the subsection):

Lemma 6.6. *For every $0 < \varepsilon \leqslant 1/(c_L - 1)$, where $1 < c_L < 6$, there exists infinitely many L with prime $p = aL + 1$ such that a is $\Theta(L^{\varepsilon})$.*

Corollary 6.11. *Let p be a prime that satisfies Lemma 6.6 for some ε. Then there exists at least $2^{\Omega(n^{\varepsilon/(1+\varepsilon)})}$ codewords in $RS[n = p, k = \Omega(n), d = n - k + 1]_{\mathbb{F}_p}$ with agreement $t = k + \Theta(n^{1/(1+\varepsilon)})$.*

Proof. Set $b = \lfloor (1-\delta)a \rfloor + 1$ for some $\delta > 0$. Thus, $k = (b-1)L \geqslant \lfloor (1-\delta)aL \rfloor = \Theta(aL) = \Theta(n)$. Further, $t = bL = k + L = k + \Theta(n^{1/(1+\varepsilon)})$. The last part follows from the fact that $n = \Theta(L^{1+\varepsilon})$. Finally, the number of codewords is at least $\left(\frac{a}{b-1}\right)^{b-1} = 2^{\Omega(a)} = 2^{\Omega(n^{\varepsilon/(1+\varepsilon)})}$. □

If one is satisfied with super polynomially many codewords, say $2^{w(n)}$ for some $w(n) = \omega(\log n)$, then choosing $\varepsilon = \frac{c \log w(n)}{\log n - c \log w(n)}$ (for some suitable constant c), gives an agreement $t = k + \Theta\left(\frac{n}{(w(n))^c}\right)$.

Proof of Theorem 6.10. The basic idea is to find a "lot" of t-tuples $(y_1, y_2, \ldots, y_t) \in \mathbb{F}_p^t$, (where for every $i \neq j$, $y_i \neq y_j$) such that the polynomial $P_{(y_1,\ldots,y_t)}(X) = \prod_{i=1}^{t}(X - y_i)$ is actually of the form

$$X^t + \sum_{j=1}^{t-L} c_j X^j$$

where c_{t-L} can be 0.[7] The above is equivalent to showing that $(y_1, \ldots, y_t)$ satisfy the following equations

$$y_1^s + y_2^s + \cdots y_t^s \quad = 0 \quad s = 1, 2, \ldots L - 1 \tag{6.7}$$

We give an "explicit" description of at least $\binom{a}{b}$ distinct $(y_1, \ldots, y_t)$ such tuples.

[7] Then $R(X) - P_{(y_1,\ldots,y_t)}(X)$ is of degree $t - L = k - 1$ as needed.

Let $\mathbb{F}_p^*$ be generated by γ and set $\alpha = \gamma^a$. Note that the order of α is exactly L. Now consider the "orbits" in $\mathbb{F}_p^*$ under the action of α. It is not too hard to see that for $0 \leqslant i < a$, the i^{th} orbit is the set $\gamma^i \mathcal{A}$, where $\mathcal{A} = \{1, \alpha, \alpha^2, \ldots, \alpha^{L-1}\}$. We will call γ^i the "representative" of the i^{th} orbit. Consider all subsets $\{i_0, \ldots, i_{b-1}\} \subseteq \{0, 1, \ldots, a-1\}$ of size b. Each such subset corresponds to a tuple $(y_1, \ldots, y_t)$ in the following manner (recall that $t = bL$). For subset $\{i_0, \ldots, i_{b-1}\}$, define $y_{dL+r} = \gamma^{i_d} \alpha^r$, where $0 \leqslant d < b$ and $0 \leqslant r < L$. Note that each such subset $\{i_0, \ldots, i_{b-1}\}$ implies a distinct tuple $(y_1, \ldots, y_t)$. Thus, there are $\binom{a}{b}$ such distinct tuples.

To complete the proof, we will now verify that (6.7) holds for every such tuple $(y_1, \ldots, y_t)$. Indeed by construction, for $s = 1, \ldots, L-1$:

$$\sum_{j=1}^{t} y_j^s = \sum_{d=0}^{b-1} \gamma^{i_d s} \left(\sum_{r=0}^{L-1} \alpha^{sr} \right) = \sum_{d=0}^{b-1} \gamma^{i_d s} \left(\frac{\alpha^{Ls} - 1}{\alpha^s - 1} \right) = 0,$$

where the last inequality follows from the the fact that the order of α is L. $\square$

We now turn to the proof of Lemma 6.6. First we need the following result, which is a special case of Linnik's theorem:

Theorem 6.12 ([78]). *There exists a constant c_L, $1 < c_L < 6$, such that for all sufficiently large d, there exists a prime p such that $p < d^{c_L}$ and $p \equiv 1 \mod d$.*

Proof of Lemma 6.6. Fix any $0 < \varepsilon \leqslant \frac{1}{c_L - 1}$. The basic idea of the proof is to "re-distribute" the product bd as aL, where $a = \Theta(L^\varepsilon)$.

Let $d = 2^r$ be sufficiently large so that it satisfies the condition of Theorem 6.12. Thus, by Theorem 6.12, $p = bd + 1$ is prime for some $1 \leqslant b < 2^{r(c_L - 1)}$. Let $2^i \leqslant b < 2^{i+1}$ for some $i \in [0, r(c_L - 1) - 1]$. Now we consider two cases depending on whether $i \leqslant i_0 = \lfloor r\varepsilon \rfloor$ or not.

First consider the case when $i \leqslant i_0$. Here define $x_i = \lfloor \frac{r\varepsilon - i}{1+\varepsilon} \rfloor$. Finally, let $a = b2^{x_i}$ and $L = 2^{r - x_i}$. First note that $0 \leqslant x_i \leqslant r$ and thus, a and L are well defined. Also note that

$$\frac{a}{L^\varepsilon} = \frac{b2^{x_i}}{2^{\varepsilon(r - x_i)}} = b2^{(1+\varepsilon)x_i - r\varepsilon} \geqslant 2^i 2^{(1+\varepsilon)(\frac{r\varepsilon - i}{1+\varepsilon}) - r\varepsilon - 1} = \frac{1}{2},$$

where the inequality follows from the fact that for all positive reals $\lfloor y \rfloor \geqslant y - 1$ and $b \geqslant 2^i$. Similarly, one can show that $a/L^\varepsilon < 4$ and thus, $a = \Theta(L^\varepsilon)$ as required.

Now we consider the case when $i > i_0$. In this case define $x_i = \lfloor \frac{r - \varepsilon(i+1)}{1+\varepsilon} \rfloor$. Finally, let $a = 2^{r - x_i}$ and $L = b2^{x_i}$. Note that $x_i \leqslant r$. Also note that as $i + 1 < r(c_L - 1)$, $x_i \geqslant 0$ and thus, a and L are well defined. As before, we first lower bound

$$\frac{a}{L^\varepsilon} = \frac{2^{r - x_i}}{b^\varepsilon 2^{\varepsilon x_i}} > \frac{2^{r - x_i}}{2^{\varepsilon(i+1) + \varepsilon x_i}} = 2^{r - (1+\varepsilon)x_i - \varepsilon(i+1)} \geqslant 1,$$

where the first inequality follows from $b < 2^{i+1}$ and the second follows from the fact that for all positive y, $\lfloor y \rfloor \leqslant y$. Similarly one can show that $\frac{a}{L^\varepsilon} \leqslant 4$, which implies that $a = \Theta(L^\varepsilon)$ as required. $\square$

Smooth Variation of the Agreement

In this section, we will see how to get rid of the "restriction" that t has to be a multiple of L in Theorem 6.10.

Theorem 6.13. *Let $L \geqslant 2$ and $0 \leqslant e < L$ be integers. Let $p = aL+1$ be a prime and define $t = bL + e$ for any $1 < b < a-1$. Let the received word $\mathbf{r}$ be the evaluation of $R(X) = X^t$ over $\mathbb{F}_p^*$. Then there are $\binom{a-1}{b}$ many codewords in $RS[n = p, k = (b-1)L + 1 + e]_{\mathbb{F}_p}$ that agree with $\mathbf{r}$ in at least t places.*

Since the proof is very similar to that of Theorem 6.10, we will just sketch the main ideas here. The basic argument used earlier was that every t-tuple $(y_1, y_2, \ldots, y_t)$ was chosen such that the polynomials $P_{(y_1,\ldots,y_t)}(X)$ and $R(X)$ agreed on the first $t-k$ co-efficients and the RS codewords were simply the polynomials $R(X) - P_{(y_1,\ldots,y_t)}(X)$. Now the simple observation is that for any fixed polynomial $D(X)$ of degree e we can get RS codewords of dimension $k' = k + e$ by considering the polynomials $D(X)\left(R(X) - P_{(y_1,\ldots,y_t)}(X)\right)$. The new agreement t' is with the new received word $R'(X) = R(X)D(X)$. Now $t' - t$ is the number of roots of $D(X)$ that are not in the set $\{y_1, \ldots, y_t\}$.

Thus, we can now vary the values of k by picking the polynomial $D(X)$ of different degrees. However, the difference $t' - k'$ might go down (as an arbitrary polynomial $D(X)$ of degree e might not have e roots and even then, some of them might be in the set $\{y_1, \ldots, y_t\}$). To get around this, while choosing the tuples $(y_1, \ldots, y_t)$, we will not pick any elements from one of the a cosets (recall that the tuples $(y_1, \ldots, y_t)$ are just a collection of b out of the a cosets formed by the orbits of $\alpha = \gamma^a$, where γ generates $\mathbb{F}_p^*$). This reduces the number of tuples from $\binom{a}{b}$ to $\binom{a-1}{b}$. Now we pick an arbitrary subset of that coset of size $0 \leqslant e < L$– say the subset is $\{z_1, \ldots, z_e\}$. Finally, pick $D(X) = \prod_{i=1}^{e}(X - z_i)$. Note that this implies that $t' = t + e$ as desired.

6.5 Bibliographic Notes and Open Questions

Results in Section 6.3 and Section 6.4.2 appeared in [59] while those in Section 6.4.3 are from [62].

Our work, specifically the part that deals with precisely describing the collection of polynomials that take values only in $\mathbb{F}_q$, bears some similarity to [51] which also exhibited limits to list recoverability of codes. One of the simple yet powerful ideas used in [51], and also in the work on extractor codes [101], is that polynomials which are r'th powers of a lower degree polynomial take only values in a multiplicative subgroup consisting of the r'th powers in the field. Specifically, the construction in [101, 51] yields roughly $n^{\frac{\ell k}{n}}$ codewords for list recovery where ℓ is the size of the S_i's in Definition 6.1. Note that this gives super-polynomially many codewords only when the input lists are asymptotically bigger than n/k.

In our work, we also use r'th powers, but the value of r is such that the r'th powers form a subfield of the field. Therefore, one can also freely add polynomials which are r'th

powers and the sum still takes on values in the subfield. This lets us demonstrate a much larger collection of polynomials which take on only a small possible number of values at every point in the field. Proving bounds on the size of this collection of polynomials used techniques that were new to this line of study.

The technique behind our results in Section 6.4.2 is closely related to that of the result of Cheng and Wan [25] on connections between Reed-Solomon list decoding and the discrete logarithm problem over finite fields. However, our aim is slightly different compared to theirs in that we want to get a large collection of codewords close by to a received word. In particular in Theorem 6.6, we get an estimate on $N_t(\beta)$ while Cheng and Wan only require $N_t(\beta) > 0$. Also Cheng and Wan consider equation (6.6) only with the choice $L_B(x) = 1$.

Ben-Sasson, Kopparty and Radhakrishnan in [12], exploiting the sparsity of *linearized polynomials*, have shown the following. For every $\delta \in (0, 1)$ there exits Reed-Solomon code of block length n and dimension $n^\delta + 1$, which contains super-polynomial many codewords that agree with a received word in at least $n^{\sqrt{\delta}}$ positions. Also they show for constant rate Reed-Solomon codes (where the rate is $R > 0$), there exists a received word that has agreement $R'N$ (where $R' > R$) with roughly $N^{\Omega(\log(1/R)}$ codewords. The received word in the above constructions, however, is not explicit. Ben-Sasson et al. also construct an explicit received word that agrees with super-polynomially many Reed-Solomon codewords in $\omega(k)$ many places, where $k = n^\delta + 1$ is the dimension of the code. However, their results do not give an explicit bad list decoding configurations for constant rate Reed-Solomon codes. The results in [12] do not work for prime fields while the results on explicit received words in this chapter do work for prime fields.

We conclude with some open questions.

Open Question 6.1. *We have shown that RS codes of rate $1/\ell$ cannot be list recovered with input lists of size ℓ in polynomial time when ℓ is a prime power. Can one show a similar result for other values of ℓ?*

Using the density of primes and our work, we can bound the rate by $O(1/\ell)$, but if it is true it will be nice to show it is at most $1/\ell$ for every ℓ.

We have shown that the $\sqrt{kn'}$ bound for polynomial reconstruction is the best possible given n' general pairs $(\beta_i, \gamma_i) \in \mathbb{F}^2$ as input. It remains a big challenge to determine whether this is the case also when the β_i's are all distinct, or equivalently

Open Question 6.2. *Is the Johnson bound is the true list decoding radius of RS codes?*

We conjecture this to be the case in the following sense: there exists a field $\mathbb{F}$ and a subset of evaluations points S such that for the Reed-Solomon code defined over $\mathbb{F}$ and S, the answer to the question above is yes. One approach that might give at least partial results would be to use some of our ideas (in particular those using the norm function, possibly extended to other symmetric functions of the automorphisms of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$) together with ideas in the work of Justesen and Hoholdt [70] who used the Trace function to demonstrate that a linear number of codewords could occur at the Johnson bound. Further, the work of

Ben-Sasson et al. [12] gives evidence for this for RS codes of rate $n^{-\varepsilon}$ for constant ε close to 0.

Open Question 6.3. *Can one show an analog of Theorem 6.6 on products of linear factors for the case when t is linear in the field size q (the currently known results work only for t up to $q^{1/2}$)?*

This is an interesting field theory question in itself, and furthermore might help towards showing the existence of super-polynomial number of Reed-Solomon codewords with agreement $t \geqslant (1+\varepsilon)k$ for some $\varepsilon > 0$ for constant rate (i.e. when k is linear in n)? It is important for the latter, however, that we show that $N_t(\beta)$ is very large for some *special* field element β in an extension field, since by a trivial counting argument it follows that there exist $\beta \in \mathbb{F}^*_{q^h}$ for which $N_t(\beta) \leqslant \binom{q}{t}/(q^h - 1)$.

Chapter 7

LOCAL TESTING OF REED-MULLER CODES

From this chapter onwards, we will switch gears and talk about property testing of codes.

7.1 Introduction

A *low degree tester* is a probabilistic algorithm which, given a degree parameter t and oracle access to a function f on n arguments (which take values from some finite field $\mathbb{F}$), has the following behavior. If f is the evaluation of a polynomial on n variables with total degree at most t, then the low degree tester must accept with probability one. On the other hand, if f is "far" from being the evaluation of some polynomial on n variables with degree at most t, then the tester must reject with constant probability. The tester can query the function f to obtain the evaluation of f at any point. However, the tester must accomplish its task by using as few probes as possible.

Low degree testers play an important part in the construction of Probabilistically Checkable Proofs (or PCPs). In fact, different parameters of low degree testers (for example, the number of probes and the amount of randomness used) directly affect the parameters of the corresponding PCPs as well as various inapproximability results obtained from such PCPs ([36, 5]). Low degree testers also form the core of the proof of MIP = NEXPTIME in [9].

Blum, Luby, and Rubinfeld designed the first low degree tester, which handled the linear case, i.e., $t = 1$ ([21]), although with a different motivation. This was followed by a series of works that gave low degree testers that worked for larger values of the degree parameter ([93, 42, 7]). However, these subsequent results as well as others which use low degree testers ([9, 43]) only work when the degree is smaller than size of the field $\mathbb{F}$. Alon et al. proposed a low degree tester for any nontrivial degree parameter over the binary field $\mathbb{F}_2$ [1].

A natural open problem was to give a low degree tester for all degrees for finite fields of size between two and the degree parameter. In this chapter we (partially) solve this problem by presenting a low degree test for multivariate polynomials over any prime field $\mathbb{F}_p$.

7.1.1 Connection to Coding Theory

The evaluations of polynomials in n variables of degree at most t are well known *Reed-Muller codes* (note that when $n = 1$, we have the Reed-Solomon codes, which we considered in Chapter 6). In particular, the evaluation of polynomials in n variables of degree

at most t over $\mathbb{F}_q$ is the Reed-Muller code or $\mathrm{RM}_q(t,n)$ with parameters t and n. These codes have length q^n and dimension $\binom{n+t}{n}$ (see [28, 29, 69] for more details). Therefore, a function has degree t if and only if (the vector of evaluations of) the function is a valid codeword in $\mathrm{RM}_q(n,t)$. In other words, low degree testing is equivalent to locally testing Reed-Muller codes.

7.1.2 Overview of Our Results

It is easier to define our tester over $\mathbb{F}_3$. To test if f has degree at most t, set $k = \lceil \frac{t+1}{2} \rceil$, and let $i = (t+1) \pmod 2$. Pick k-vectors $y_1, \cdots, y_k$ and b from $\mathbb{F}_3^n$, and test if

$$\sum_{c \in \mathbb{F}_3^k; c=(c_1,\cdots,c_k)} c_1^i f(b + \sum_{j=1}^{k} c_j y_j) = 0,$$

where for notational convenience we use $0^0 = 1$ (and we will stick to this convention throughout this chapter). We remark here that a polynomial of degree at most t always passes the test, whereas a polynomial of degree greater than t gets caught with non-negligible probability α. To obtain a constant rejection probability we repeat the test $\Theta(1/\alpha)$ times.

The analysis of our test follows a similar general structure developed by Rubinfeld and Sudan in [93] and borrows techniques from [93, 1]. The presence of a doubly-transitive group suffices for the analysis given in [93]. Essentially we show that the presence of a doubly-transitive group acting on the coordinates of the dual code does indeed allow us to localize the test. However, this gives a weaker result. We use techniques developed in [1] for better results, although the adoption is not immediate. In particular the interplay between certain geometric objects described below and their polynomial representations plays a pivotal role in getting results that are only about a quadratic factor away from optimal query complexity.

In coding theory terminology, we show that Reed-Muller codes over prime fields are locally testable. We further consider a new basis of Reed-Muller code over prime fields that in general differs from the minimum weight basis. This allows us to present a novel exact characterization of the multivariate polynomials of degree t in n variables over prime fields. Our basis has a clean geometric structure in terms of *flats* [69], and unions of parallel flats but with different weights assigned to different parallel flats[1]. The equivalent polynomial and geometric representations allow us to provide an almost optimal test.

Main Result

Our results may be stated quantitatively as follows. For a given integer $t \geqslant (p-1)$ and a given real $\varepsilon > 0$, our testing algorithm queries f at $O\left(\frac{1}{\varepsilon} + t \cdot p^{\frac{2t}{p-1}+1}\right)$ points to determine

[1]The natural basis given in [28, 29] assigns the same weight to each parallel flat.

whether f can be described by a polynomial of degree at most t. If f is indeed a polynomial of degree at most t, our algorithm always accepts, and if f has a relative Hamming distance at least ε from every degree t polynomial, then our algorithm rejects f with probability at least $\frac{1}{2}$. (In the case $t < (p-1)$, our tester still works but more efficient testers are known). Our result is almost optimal since any such testing algorithm must query f in at least $\Omega(\frac{1}{\varepsilon} + p^{\frac{t+1}{p-1}})$ many points (see Corollary 7.5).

We extend our analysis also to obtain a *self-corrector* for f (as defined in [21]), in case the function f is reasonably close to a degree t polynomial. Specifically, we show that the value of the function f at any given point $x \in \mathbb{F}_p^n$ may be obtained with good probability by querying f on $\Theta(p^{t/p})$ random points. Using pairwise independence we can achieve even higher probability by querying f on $p^{O(t/p)}$ random points and using majority logic decoding.

7.1.3 Overview of the Analysis

The design of our tester and its analysis follows the following general paradigm first formalized by Rubinfeld and Sudan [93]. The analysis also uses additional ideas used in [1]. In this section, we review the main steps involved.

The first step is coming up with an *exact characterization* for functions that have low degree. The characterization identifies a collection of subsets of points and a predicate such that an input function is of low degree if and only if for every subset in the collection, the predicate is satisfied by the evaluation of the function at the points in the subset. The second step entails showing that the characterization is a *robust characterization*, that is, the following natural tester is indeed a local tester (see section 2.3 for a formal definition): Pick one of the subsets in the collection uniformly at random and check if the predicate is satisfied by the evaluation of the function on the points in the chosen subset. Note that the number of queries made by the tester is bounded above by the size of the largest subset in the collection.

There is a natural characterization for polynomials of low degree using their alternative interpretation as a RM code. As RM code is a linear code, a function is of low degree if and only if it is orthogonal to every codeword in the dual of the corresponding RM code. The problem with the above characterization is that the resulting local tester will have to make as many queries as the maximum number of non-zero position in any dual codeword, which can be large. To get around this problem, instead of considering all codewords in the dual of the RM code, we consider a collection of dual codewords that have few non-zero positions. To obtain an exact characterization, note that this collection has to generate the dual code.

We use the well known fact that the dual of a RM code is a RM code (with different parameters). Thus, to obtain a collection of dual codewords with low weight that generate the dual of a RM code it is enough to find low weight codewords that generate every RM code. To this end we show that the characteristic vector of any affine subspace (also called a

flat in RM terminology [69]) generates certain RM codes. To complete the characterization, we show that any RM code can be generated by flats and certain weighted characteristic vectors of affine subspaces (which we call *pseudoflats*). To prove these we look at the affine subspaces as the intersection of (a fixed number of) hyperplanes and alternatively represent the characteristic vectors as polynomials.

To prove that the above exact characterization is robust we use the *self-correcting* approach ([21, 93]). Given an input f we define a related function g as follows. The value of $g(x)$ is defined to be the most frequently occurring value, or *plurality*, of f at correlated random points. The major part of the analysis is to show that if f disagrees from all low degree polynomials in a lot of places then the tester rejects with high probability.

The analysis proceeds by first showing that f and g agree on most points. Then we show that if the tester rejects with low enough probability then g is a low degree polynomial. In other words, if f is far enough from all low degree polynomials, then the tester rejects with high probability. To complete the proof, we take care of the case when f is close to some low degree polynomial separately.

7.2 Preliminaries

Throughout this chapter, we use p to denote a prime and q to denote a prime power (p^s for some positive integer s) to be a prime power. In this chapter, we will mostly deal with prime fields. We therefore restrict most definitions to the prime field setting.

For any $t \in [n(q-1)]$, let $\mathcal{P}_t$ denote the family of all functions over $\mathbb{F}_q^n$ that are polynomials of total degree at most t (and w.l.o.g. individual degree at most $q-1$) in n variables. In particular $f \in \mathcal{P}_t$ if there exists coefficients $a_{(e_1,\cdots,e_n)} \in \mathbb{F}_q$, for every $i \in [n]$, $e_i \in \{0, \cdots, q-1\}$, $\sum_{i=1}^n e_i \leqslant t$, such that

$$f = \sum_{(e_1,\cdots,e_n)\in\{0,\cdots,q-1\}^n; 0\leqslant\sum_{i=1}^n e_i\leqslant t} a_{(e_1,\cdots,e_n)} \prod_{i=1}^n x_i^{e_i}. \tag{7.1}$$

The codeword corresponding to a function will be the evaluation vector of f. We recall the definition of the (Primitive) Reed-Muller code as described in [69, 29].

Definition 7.1. *Let $V = \mathbb{F}_q^n$ be the vector space of n-tuples, for $n \geqslant 1$, over the field $\mathbb{F}_q$. For any k such that $0 \leqslant k \leqslant n(q-1)$, the k^{th} order Reed-Muller code $\mathrm{RM}_q(k,n)$ is the subspace of $\mathbb{F}_q^{|V|}$ of all n-variable polynomial functions (reduced modulo $x_i^q - x_i$) of degree at most k.*

This implies that the code corresponding to the family of functions $\mathcal{P}_t$ is $\mathrm{RM}_q(t,n)$. Therefore, a characterization for one will simply translate into a characterization for the other.

We will be using terminology defined in Section 2.3. We now briefly review the definitions that are relevant to this chapter. For any two functions $f, g : \mathbb{F}_q^n \to \mathbb{F}_q$, the relative

distance $\delta(f,g) \in [0,1]$ between f and g is defined as $\delta(f,g) \stackrel{def}{=} \Pr_{x\in\mathbb{F}_q^n}[f(x) \neq g(x)]$. For a function g and a family of functions F (defined over the same domain and range), we say g is ε- *close* to F, for some $0 < \varepsilon < 1$, if, there exists an $f \in F$, where $\delta(f,g) \leqslant \varepsilon$. Otherwise it is ε- *far* from F.

A one sided testing algorithm (*one-sided tester*) for $\mathcal{P}_t$ is a probabilistic algorithm that is given query access to a function f and a distance parameter ε, $0 < \varepsilon < 1$. If $f \in \mathcal{P}_t$, then the tester should always accept f (perfect completeness), and if f is ε-*far* from $\mathcal{P}_t$, then with probability at least $\frac{1}{2}$ the tester should reject f.

For vectors $x, y \in \mathbb{F}_p^n$, the dot (scalar) product of x and y, denoted $x \cdot y$, is defined to be $\sum_{i=1}^{n} x_i y_i$, where w_i denotes the i^{th} co-ordinate of w.

To motivate the next notation which we will use frequently, we give a definition.

Definition 7.2. *For any $k \geqslant 0$, a k-*flat *in $\mathbb{F}_p^n$ is a k-dimensional affine subspace. Let $y_1, \cdots, y_k \in \mathbb{F}_p^n$ be linearly independent vectors and $b \in \mathbb{F}_p^n$ be a point. Then the subset*

$$L = \{\sum_{i=1}^{k} c_i y_i + b | \forall i \in [k]\ c_i \in \mathbb{F}_p\}$$

is a k-dimensional flat. We will say that L is generated by $y_1, \cdots, y_k$ at b. The incidence vector of the points in a given k-flat will be referred to as the codeword corresponding to the given k-flat.

Given a function $f : \mathbb{F}_p^n \to \mathbb{F}_p$, for $y_1, \cdots, y_l, b \in \mathbb{F}_p^n$ we define

$$T_f^0(y_1, \cdots, y_l, b) \stackrel{def}{=} \sum_{c=(c_1,\cdots,c_l)\in\mathbb{F}_p^l} f(b + \sum_{i\in[l]} c_i y_i), \tag{7.2}$$

which is the sum of the evaluations of function f over an l-flat generated by $y_1, \cdots, y_l$, at b. Alternatively, as we will see later in Observation 7.4, this can also be interpreted as the dot product of the codeword corresponding to the l-flat generated by $y_1, \cdots, y_l$ at b and that corresponding to the function f.

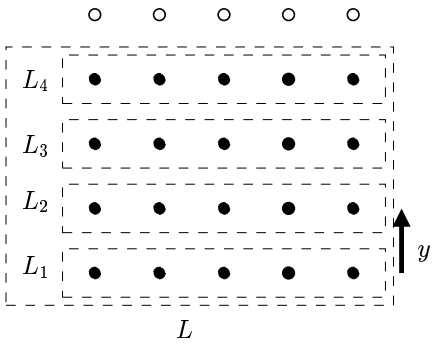
While k-flats are well-known, we define a new geometric object, called a pseudoflat. A k-pseudoflat is a union of $(p-1)$ parallel $(k-1)$-flats.

Definition 7.3. *Let $L_1, L_2, \cdots, L_{p-1}$ be parallel $(k-1)$-flats ($k \geqslant 1$), such that for some $y \in \mathbb{F}_p^n$ and all $t \in [p-2]$, $L_{t+1} = y + L_t$, where for any set $S \subseteq \mathbb{F}_p^n$ and $y \in \mathbb{F}_p^n$, $y + S \stackrel{def}{=} \{x + y | x \in S\}$. We define a k-*pseudoflat *to be the union of the set of points L_1 to L_{p-1}. Further, given an r (where $1 \leqslant r \leqslant p-2$) and a k-pseudoflat, we define a (k,r)-*pseudoflat vector *as follows. Let I_j be the incidence vector of L_j for $j \in [p-1]$. Then the (k,r)-pseudoflat vector is defined to be $\sum_{j=1}^{p-1} j^r I_j$. We will also refer to the (k,r)-pseudoflat vector as a codeword.*

Let L be a k-pseudoflat. Also, for $j \in [p-1]$, let L_j be the $(k-1)$-flat generated by $y_1, \cdots, y_{k-1}$ at $b + j \cdot y$, where $y_1, \cdots, y_{k-1}$ are linearly independent. Then we say that the

(k, r)*-pseudoflat vector corresponding to* L *as well as the pseudoflat* L*, are* generated by $y, y_1, \cdots, y_{k-1}$ *at* b exponentiated along y.

See Figure 7.1 for an illustration of the Definition 7.3.



0	0	0	0	0
4	4	4	4	4
3	3	3	3	3
2	2	2	2	2
1	1	1	1	1

$(2, 1)$-pseudoflat vector corresponding to L

Figure 7.1: Illustration of a k-pseudoflat L defined over $\mathbb{F}_p^n$ with $k = 2, p = 5$ and $n = 5$. Picture on the left shows the points in L (recall that each of $L_1, \ldots, L_4$ are 1-flats or lines). Each L_i (for $1 \leqslant i \leqslant 4$) has $p^{k-1} = 5$ points in it. The points in L are shown by filled circles and the points in $\mathbb{F}_5^5 \setminus L$ are shown by unfilled circles. The picture on the right is the $(2, 1)$-pseudoflat corresponding to L.

Given a function $f : \mathbb{F}_p^n \to \mathbb{F}_p$, for $y_1, \cdots, y_l, b \in \mathbb{F}_p^n$, for all $i \in [p-2]$, we define

$$T_f^i(y_1, \cdots, y_l, b) \stackrel{def}{=} \sum_{c=(c_1,\cdots,c_l)\in\mathbb{F}_p^l} c_1^i \cdot f(b + \sum_{j\in[l]} c_j y_j). \tag{7.3}$$

As we will see later in Observation 7.5, the above can also be interpreted as the dot product of the codeword corresponding to the (l, r)-pseudoflat vector generated by $y_1, \cdots, y_l$ at b exponentiated along y_1 and the codeword corresponding to the function f.

7.2.1 Facts from Finite Fields

In this section we spell out some facts from finite fields which will be used later. We begin with a simple lemma.

Lemma 7.1. *For any* $t \in [q-1]$, $\sum_{a\in\mathbb{F}_q} a^t \neq 0$ *if and only if* $t = q-1$.

Proof. First note that $\sum_{a\in\mathbb{F}_q} a^t = \sum_{a\in\mathbb{F}_q^*} a^t$. Observing that for any $a \in \mathbb{F}_q^*$, $a^{q-1} = 1$, it follows that $\sum_{a\in\mathbb{F}_q^*} a^{q-1} = \sum_{a\in\mathbb{F}_q^*} 1 = -1 \neq 0$.

Next we show that for all $t \neq q-1$, $\sum_{a \in \mathbb{F}_q^*} a^t = 0$. Let α be a generator of $\mathbb{F}_q^*$. The sum can be re-written as $\sum_{i=0}^{q-2} \alpha^{it} = \frac{\alpha^{t(q-1)}-1}{\alpha^t-1}$. The denominator is non-zero for $t \neq q-1$ and thus, the fraction is well defined. The proof is complete by noting that $\alpha^{t(q-1)} = 1$. □

This immediately implies the following lemma.

Lemma 7.2. *Let $t_1, \cdots, t_l \in [q-1]$. Then*

$$\sum_{(c_1,\cdots,c_l)\in(\mathbb{F}_q)^l} c_1^{t_1} c_2^{t_2} \cdots c_l^{t_l} \neq 0 \text{ if and only if } t_1 = t_2 = \cdots = t_l = q-1. \tag{7.4}$$

Proof. Note that the left hand side can be rewritten as $\prod_{i\in[l]} \left(\sum_{c_i \in \mathbb{F}_q} c_i^{t_i}\right)$. □

We will need to transform products of variables to powers of linear functions in those variables. With this motivation, we present the following identity.

Lemma 7.3. *For each k, s.t. $0 < k \leqslant (p-1)$ there exists $c_k \in \mathbb{F}_p^*$ such that*

$$c_k \prod_{i=1}^{k} x_i = \sum_{i=1}^{k} (-1)^{k-i} S_i \quad \textit{where} \quad S_i = \sum_{\emptyset \neq I \subseteq [k]; |I|=i} \left(\sum_{j\in I} x_j\right)^k. \tag{7.5}$$

Proof. Consider the right hand side of the (7.5). Note that all the monomials are of degree exactly k. Also note that $\prod_{i=1}^{k} x_i$ appears only in the S_k and nowhere else. Now consider any other monomial of degree k that has a support of size j, where $0 < j < k$: w.l.o.g. assume that this monomial is $M = x_1^{i_1} x_2^{i_2} \cdots x_j^{i_j}$ such that $i_1 + \cdots + i_j = k$. Now note that for any $I \supseteq [j]$, M appears with a coefficient of $\binom{k}{i_1, i_2, \ldots, i_j}$ in the expansion of $(\sum_{\ell \in I} x_\ell)^k$. Further for every $i \geqslant j$, the number of choices of $I \supseteq [j]$ with $|I| = i$ is exactly $\binom{k-j}{k-i}$. Therefore, summing up the coefficients of M in the various summands S_i (along with the $(-1)^{k-i}$ factor), we get that the coefficient of M in the right hand side of (7.5) is

$$\begin{aligned}
\binom{k}{i_1, i_2, \ldots, i_j} \left(\sum_{i=j}^{k} (-1)^{k-i} \binom{k-j}{k-i}\right) &= \binom{k}{i_1, i_2, \ldots, i_j} \left(\sum_{\ell=0}^{k-j} (-1)^{k-j-\ell} \binom{k-j}{k-j-\ell}\right) \\
&= \binom{k}{i_1, i_2, \ldots, i_j} (1-1)^{k-j} \\
&= 0.
\end{aligned}$$

Moreover, it is clear that $c_k = \binom{k}{1,1,\ldots,1} = k! \pmod p$ and $c_k \neq 0$ for the choice of k. □

7.3 Characterization of Low Degree Polynomials over $\mathbb{F}_p$

In this section we present an exact characterization for the family $\mathcal{P}_t$ over prime fields. Specifically we prove the following:

Theorem 7.1. *Let $t = (p-1)\cdot k + r$. (Note $0 \leqslant r \leqslant p-2$.) Let $i = p-2-r$. Then a function f belongs to $\mathcal{P}_t$, if and only if for every $y_1, \cdots, y_{k+1}, b \in \mathbb{F}_p^n$, we have*

$$T_f^i(y_1, \cdots, y_{k+1}, b) = 0 \tag{7.6}$$

As mentioned previously, a characterization for the family $\mathcal{P}_t$ is equivalent to a characterization for $\mathrm{RM}_p(t,n)$. It turns out that it is easier to characterize $\mathcal{P}_t$ when viewed as $\mathrm{RM}_p(t,n)$. Therefore our goal is to determine whether a given word belongs to the RM code. Since we deal with a linear code, a simple strategy will then be to check whether the given word is orthogonal to all the codewords in the dual code. Though this yields a characterization, this is computationally inefficient. Note however that the dot product is linear in its input. Therefore checking orthogonality with a basis of the dual code suffices. To make it computationally efficient, we look for a basis with small weights. The above theorem essentially is a clever restatement of this idea.

We recall the following useful lemma which can be found in [69].

Lemma 7.4. $\mathrm{RM}_q(k,n)$ *is a linear code with block length q^n and minimum distance $(R+1)q^Q$ where R is the remainder and Q the quotient resulting from dividing $(q-1)\cdot n - k$ by $(q-1)$. Then* $\mathrm{RM}_q(k,n)^\perp = \mathrm{RM}_q((q-1)\cdot n - k - 1, n)$.

Since the dual of a RM code is again a RM code (of appropriate order), we therefore need the generators of RM code (of arbitrary order). We first establish that flats and pseudoflats (of suitable dimension and exponent) indeed generate the Reed-Muller code. We then end the section with a proof of Theorem 7.1 and a few remarks.

We begin with few simple observations about flats. Note that an l-flat L is the intersection of $(n-l)$ hyperplanes in general position. Equivalently, it consists of all points v that satisfy $(n-l)$ linear equations over $\mathbb{F}_p$ (i.e., one equation for each hyperplane): $\forall i \in [n-l] \quad \sum_{j=1}^n c_{ij}x_j = b_i$ where c_{ij}, b_i defines the i^{th} hyperplane (i.e., v satisfies $\sum_{j=1}^n c_{ij}v_j = b_i$). General position means that the matrix $\{c_{ij}\}$ has rank $(n-l)$. Note that then the characteristic function (and by abuse of notation the incidence vector) of L can be written as

$$\prod_{i=1}^{n-l}(1 - (\sum_{j=1}^{n} c_{ij}x_j - b_i)^{p-1}) = \begin{cases} 1 & \text{if } (v_1, \cdots, v_l) \in L \\ 0 & \text{otherwise} \end{cases} \tag{7.7}$$

We now record a lemma here that will be used later in this section.

Lemma 7.5. *For $k \geqslant l$, the incidence vector of any k-flat is a linear sum of the incidence vectors of l-flats.*

Proof. Let $k = l + r$ and let W be an k-flat. We want to show that it is generated by a linear combination of l flats.

Let W be generated by $y_1, \cdots, y_{l-1}, w_1, \cdots, w_{r+1}$ at b. For each non-zero vector $c_i = \langle c_{i1}, \ldots, c_{i(r+1)} \rangle$ in $\mathbb{F}_p^{r+1}$ define:

$$v_i = \sum_{j=1}^{r+1} c_{ij} w_j.$$

Clearly there are $(p^{r+1} - 1)$ such v_i. Now for each $i \in [p^{r+1} - 1]$, define an l-flat L_i generated by $y_1, \cdots, y_{l-1}, v_i$ at b. Denote the incidence vector of a flat V by 1_V, then we claim that

$$1_W = (p-1) \sum_{i=1}^{p^{r+1}-1} 1_{L_i}. \tag{7.8}$$

Since the vectors $y_1, \ldots, y_{l-1}, w_1, \ldots, w_{r+1}$ are all linearly independent, we can divide the proof in three sub cases:

- $v \in W$ is of the form $b + \sum_{i=1}^{l-1} e_i y_i$, for some $e_1, \ldots, e_{l-1} \in \mathbb{F}_p$: Then each flat L_i contributes 1 to the right hand side of (7.8), and therefore, the right hand side is $(p-1)(p^{r+1} - 1) = 1$ in $\mathbb{F}_p$.

- $v \in W$ is of the form $b + \sum_{i=1}^{r+1} d_i w_i$ for some $d_1, \ldots, d_{r+1} \in \mathbb{F}_p$: Then the flats L_j that contribute have $V_j = a \cdot \sum_{i=1}^{r+1} d_i w_i$, for $a = 1, \ldots, p-1$. Therefore, the right hand side of (7.8) is $(p-1)^2 = 1$ in $\mathbb{F}_p$.

- $v \in W$ is of the form $b + \sum_{i=1}^{l-1} e_i y_i + \sum_{i=1}^{r+1} d_i w_i$: Then the flats L_j that contribute have $V_j = a \cdot \sum_{i=1}^{r+1} d_i w_i$, for $a = 1, \ldots, p-1$. Therefore, the right hand side of (7.8) is $(p-1)^2 = 1$ in $\mathbb{F}_p$.

□

As mentioned previously, we give an explicit basis for $\mathrm{RM}_p(r, n)$. For the special case of $p = 3$, our basis coincides with the min-weight basis given in [29].[2] However, in general, our basis differs from the min-weight basis provided in [29].

The following Proposition shows that the incidence vectors of flats form a basis for the Reed-Muller code of orders that are multiples of $(p-1)$.

Proposition 7.6. $\mathrm{RM}_p((p-1)(n-l), n)$ *is generated by the incidence vectors of the* l*-flats.*

[2]The equations of the hyperplanes are slightly different in our case; nonetheless, both of them define the same basis generated by the min-weight codewords.

Proof. We first show that the incidence vectors of the l-flats are in $\mathrm{RM}_p((p-1)(n-l), n)$. Recall that L is the intersection of $(n-l)$ independent hyperplanes. Therefore using (7.7), L can be represented by a polynomial of degree at most $(n-l)(p-1)$ in $x_1, \cdots, x_n$. Therefore the incidence vectors of l-flats are in $\mathrm{RM}_p((p-1)(n-l), n)$.

We prove that $\mathrm{RM}_p((p-1)(n-l), n)$ is generated by l-flats by induction on $n-l$. When $n-l=0$, the code consists of constants, which is clearly generated by n-flats i.e., the whole space.

To prove for an arbitrary $(n-l) > 0$, we show that any monomial of total degree $d \leqslant (p-1)(n-l)$ can be written as a linear sum of the incidence vectors of l-flats. Let the monomial be $x_1^{e_1} \cdots x_s^{e_s}$. Rewrite the monomials as $\underbrace{x_1 \cdots x_1}_{e_1 \text{ times}} \cdots \underbrace{x_s \cdots x_t}_{e_s \text{ times}}$. Group into products of $(p-1)$ (not necessarily distinct) variables as much as possible. Rewrite each group using (7.5) with $k = (p-1)$. For any incomplete group of size d', use the same equation by setting the last $(p-1-d')$ variables to the constant 1. After expansion, the monomial can be seen to be a sum of products of at most $(n-l)$ linear terms raised to the power of $p-1$. We can add to it a polynomial of degree at most $(p-1)(n-l-1)$ so as to represent the resulting polynomial as a sum of polynomials, each polynomial as in (7.7). Each such non-zero polynomial is generated by a t flat, $t \geqslant l$. By induction, the polynomial we added is generated by $(l+1)$ flats. Thus, by Lemma 7.5 our given monomial is generated by l-flats. □

This leads to the following observation:

Observation 7.4. *Consider an l-flat generated by $y_1, \cdots, y_l$ at b. Denote the incidence vector of this flat by I. Then the right hand side of (7.2) may be identified as $I \cdot f$, where I and f denote the vector corresponding to respective codewords and $\cdot$ is the dot (scalar) product.*

To generate a Reed-Muller code of any arbitrary order, we need pseudoflats. Note that the points in a k-pseudoflat may alternatively be viewed as the space given by the union of intersections of $(n-k-1)$ hyperplanes, where the union is parameterized by another hyperplane that does not take one particular value. Concretely, it is the set of points v which satisfy the following constraints over $\mathbb{F}_p$:

$$\forall i \in [n-k-1] \sum_{j=1}^{n} c_{ij} x_j = b_i; \text{ and } \quad \sum_{j=1}^{n} c_{n-k,j} x_j \neq b_{n-k}.$$

Thus the values taken by the points of a k-pseudoflat in its corresponding (k, r)-pseudoflat vector is given by the polynomial

$$\prod_{i=1}^{n-k-1} (1 - (\sum_{j=1}^{n} c_{ij} x_j - b_i)^{p-1}) \cdot (\sum_{j=1}^{n} c_{n-k,j} x_j - b_{n-k})^r \tag{7.9}$$

Remark 7.1. *Note the difference between (7.9) and the basis polynomial in [29] that (along with the action of the affine general linear group) yields the min-weight codewords:*

$$h(x_1, \cdots, x_m) = \prod_{i=1}^{k-1}(1 - (x_i - w_i)^{p-1}) \prod_{j=1}^{r}(x_k - u_j),$$

where $w_1, \cdots, w_{k-1}, u_1, \cdots, u_r \in \mathbb{F}_p$.

The next lemma shows that the code generated by the incidence vectors of l-flats is a subcode of the code generated by the (l, r)-pseudoflats vectors.

Claim 7.7. *The* (l, r)*-pseudoflats vectors, where* $l \geqslant 1$ *and* $r \in [p-2]$*, generate a code containing the incidence vectors of* l*-flats.*

Proof. Let W be the incidence vector of an l-flat generated by $y_1, \cdots, y_l$ at b. Since pseudoflat vector corresponding to an l-pseudoflat (as well as a flat) assigns the same value to all points in the same $(l-1)$-flat, we can describe W (as well as any $(l, \cdot)$-pseudoflat vector) by giving its values on each of its p $l-1$-flats. In particular, $W = \langle 1, \ldots, 1 \rangle$. Let L_j be a pseudoflat generated by $y_1, \cdots, y_l$ exponentiated along y_1 at $b + j \cdot y_1$, for each $j \in \mathbb{F}_p$, and let V_j be the corresponding (l, r)- pseudoflat vector. By Definition 7.3, V_j assigns a value i^r to the $(l-1)$-flat generated by $y_2, \cdots, y_l$ at $b + (j+i)y$. Rewriting them in terms of the values on its $l-1$-flats yields that $V_j = \langle (p-j)^r, (p-j+1)^r, \cdots, (p-j+i)^r, \cdots, (p-j-1)^r \rangle \in \mathbb{F}_p^p$. Let λ_j denote p variables for $j = 0, 1, \cdots, p-1$, each taking values in $\mathbb{F}_p$. Then a solution to the following system of equations

$$1 = \sum_{j \in \mathbb{F}_p} \lambda_j (i-j)^r \quad \text{for every } 0 \leqslant l \leqslant p-1$$

implies that $W = \sum_{j=0}^{p-1} \lambda_j V_j$, which suffices to establish the claim. Consider the identity

$$1 = (-1) \sum_{j \in \mathbb{F}_p} (j+i)^r j^{p-1-r}$$

which may be verified by expanding and applying Lemma 7.1. Setting λ_j to $(-1)(-j)^{p-1-r}$ establishes the claim. □

The next Proposition complements Proposition 7.6. Together they say that by choosing pseudoflats appropriately, Reed-Muller codes of any given order can be generated. This gives an equivalent representation of Reed-Muller codes. An exact characterization then follows from this alternate representation.

Proposition 7.8. *For every* $r \in [p-2]$*, the linear code generated by* (l, r)*-pseudoflat vectors is equivalent to* $\mathrm{RM}_p((p-1)(n-l)+r, n)$.

Proof. For the forward direction, consider an l-pseudoflat L. Its corresponding (l, r)-pseudoflat vector is given by an equation similar to (7.9). Thus the codeword corresponding to the evaluation vector of this flat can be represented by a polynomial of degree at most $(p-1)(n-l)+r$. This completes the forward direction.

Since monomials of degree at most $(p-1)(n-l)$ are generated by the incidence vectors of l-flats, Claim 7.7 will establish the proposition for such monomials. Thus, to prove the other direction of the proposition, we restrict our attention to monomials of degree at least $(p-1)(n-l)+1$ and show that these monomials are generated by (l, r)-pseudoflats vectors. Now consider any such monomial. Let the degree of the monomial be $(p-1)(n-l)+r'$ ($1 \leqslant r' \leqslant r$). Rewrite it as in Proposition 7.6. Since the degree of the monomial is $(p-1)(n-l)+r'$, we will be left with an incomplete group of degree r'. We make any incomplete group complete by adding 1's (as necessary) to the product. We then use Lemma 7.3 to rewrite each (complete) group as a linear sum of r^{th} powered terms. After expansion, the monomial can be seen to be a sum of product of at most $(n-l)$ degree $(p-1)^{th}$ powered linear terms and a r^{th} powered linear terms. Each such polynomial is generated either by an (l, r)-pseudoflat vector or an l-flat. Claim 7.7 completes the proof. □

The following is analogous to Observation 7.4.

Observation 7.5. *Consider an l-pseudoflat, generated by $y_1, \cdots, y_l$ at b exponentiated along y_1. Let E be its corresponding (l, r)-pseudoflat vector. Then the right hand side of (7.3) may be interpreted as $E \cdot f$.*

Now we prove the exact characterization.

Proof of Theorem 7.1: The proof directly follows from Lemma 7.4, Proposition 7.6, Proposition 7.8 and Observation 7.4 and Observation 7.5. Indeed by Observation 7.4, Observation 7.5 and (7.6) are essentially tests to determine whether the dot product of the function with every vector in the dual space of $\mathrm{RM}_p(t, n)$ evaluates to zero. □

Remark 7.2. *One can obtain an alternate characterization from Remark 7.1 which we state here without proof.*

Let $t = (p-1) \cdot k + R$ (note $0 < R \leqslant (p-2)$). Let $r = p - R - 2$. Let $W \subseteq \mathbb{F}_p$ with $|W| = r$. Define the polynomial $g(x) \stackrel{def}{=} \prod_{\alpha \in W}(x - \alpha)$ if W is non-empty; and $g(x) = 1$ otherwise. Then a function belong to $\mathcal{P}_t$ if and only if for every $y_1, \cdots, y_{k+1}, b \in \mathbb{F}_p^n$, we have

$$\sum_{c_1 \in \mathbb{F}_p \setminus W} g(c_1) \sum_{(c_2, \cdots, c_{k+1}) \in \mathbb{F}_p^k} f(b + \sum_{i=1}^{k+1} c_i \cdot y_i) = 0.$$

Moreover, this characterization can also be extended to certain degrees for more general fields, i.e., $\mathbb{F}_{p^s}$ (see the next remark).

Remark 7.3. *The exact characterization of low degree polynomials as claimed in [42] may be proved using duality. Note that their proof works as long as the dual code has a min-weight basis (see [29]). Suppose that the polynomial has degree $d \leqslant q - q/p - 1$, then*

the dual of $\mathrm{RM}_q(d,n)$ *is* $\mathrm{RM}_q((q-1)n-d-1,n)$ *and therefore has a min-weight basis. Note that then the dual code has min-weight* $(d+1)$*. Therefore, assuming the minimum weight codewords constitute a basis (that is, the span of all codewords with the minimum Hamming weight is the same as the code), any* $d+1$ *evaluations of the original polynomial on a line are dependent and vice-versa.*

7.4 A Tester for Low Degree Polynomials over $\mathbb{F}_p^n$

In this section we present and analyze a one-sided tester for $\mathcal{P}_t$. The analysis of the algorithm roughly follows the proof structure given in [93, 1]. We emphasize that the generalization from [1] to our case is not straightforward. As in [93, 1] we define a self-corrected version of the (possibly corrupted) function being tested. The straightforward adoption of the analysis given in [93] gives reasonable bounds. However, a better bound is achieved by following the techniques developed in [1]. In there, they show that the self-corrector function can be interpolated with overwhelming probability. However their approach appears to use special properties of $\mathbb{F}_2$ and it is not clear how to generalize their technique for arbitrary prime fields. We give a clean formulation which relies on the flats being represented through polynomials as described earlier. In particular, Claims 7.14, 7.15 and their generalizations appear to require our new polynomial based view.

7.4.1 Tester in $\mathbb{F}_p$

In this subsection we describe the algorithm when underlying field is $\mathbb{F}_p$.
Algorithm Test-$\mathcal{P}_t$ in $\mathbb{F}_p$

0. Let $t=(p-1)\cdot k+R$, $0\leqslant R<p-1$. Denote $r=p-2-R$.
1. Uniformly and independently at random select $y_1,\cdots,y_{k+1},b\in\mathbb{F}_p^n$.
2. If $T_f^r(y_1,\cdots,y_{k+1},b)\neq 0$, then *reject*, else *accept*.

Theorem 7.2. *The algorithm* **Test-$\mathcal{P}_t$ in $\mathbb{F}_p$** *is a one-sided tester for $\mathcal{P}_t$ with a success probability at least min$(c(p^{k+1}\varepsilon),\frac{1}{2(k+7)p^{k+2}})$ for some constant $c>0$.*

Corollary 7.3. *Repeating the algorithm* **Test-$\mathcal{P}_t$ in $\mathbb{F}_p$** *for $\Theta(\frac{1}{p^{k+1}\varepsilon}+kp^k)$ times, the probability of error can be reduced to less than $1/2$.*

We will provide a general proof framework. However, for the ease of exposition we prove the main technical lemmas for the case of $\mathbb{F}_3$. The proof idea in the general case is similar and the details are omitted. Therefore we will essentially prove the following.

Theorem 7.4. *The algorithm* **Test-$\mathcal{P}_t$ in $\mathbb{F}_3$** *is a one-sided tester for $\mathcal{P}_t$ with success probability at least min$(c(3^{k+1}\varepsilon),\frac{1}{2(t+7)3^{t/2+1}})$ for some constant $c>0$.*

7.4.2 *Analysis of Algorithm* Test-$\mathcal{P}_t$

In this subsection we analyze the algorithm described in Section 7.4.1. From Claim 7.1 it is clear that if $f \in \mathcal{P}_t$, then the tester accepts. Thus, the bulk of the proof is to show that if f is ε-*far* from $\mathcal{P}_t$, then the tester rejects with significant probability. Our proof structure follows that of the analysis of the test in [1]. In particular, let f be the function to be tested for membership in $\mathcal{P}_t$. Assume we perform Test T_f^i for an appropriate i as required by the algorithm described in Section 7.4.1. For such an i, we define $g_i : \mathbb{F}_p^n \to \mathbb{F}_p$ as follows: For $y \in F_p^n, \alpha \in \mathbb{F}_p$, denote $p_{y,\alpha} = \Pr_{y_1,\cdots,y_{k+1}}[f(y) - T_f^i(y - y_1, y_2, \cdots, y_{k+1}, y_1) = \alpha]$. Define $g_i(y) = \alpha$ such that $\forall \beta \neq \alpha \in \mathbb{F}_p, p_{y,\alpha} \geqslant p_{y,\beta}$ with ties broken arbitrarily. With this meaning of plurality, for all $i \in [p-2] \cup \{0\}$, g_i can be written as:

$$g_i(y) \;=\; \text{plurality}_{y_1,\cdots,y_{k+1}} \left[f(y) - T_f^i(y - y_1, y_2, \cdots, y_{k+1}, y_1) \right]. \tag{7.10}$$

Further we define

$$\eta_i \stackrel{def}{=} Pr_{y_1,\cdots,y_{k+1},b}[T_f^i(y_1, \cdots, y_{k+1}, b) \neq 0] \tag{7.11}$$

The next lemma follows from a Markov-type argument.

Lemma 7.9. *For a fixed $f : \mathbb{F}_p^n \to \mathbb{F}_p$, let g_i, η_i be defined as above. Then, $\delta(f, g_i) \leqslant 2\eta_i$.*

Proof. If for some $y \in \mathbb{F}_p^n$, $\Pr_{y_1,\cdots,y_{k+1}}[f(y) = f(y) - T_f^i(y - y_1, y_2, \cdots, y_{k+1}, y_1)] > 1/2$, then $g(y) = f(y)$. Thus, we only need to worry about the set of elements y such that $\Pr_{y_1,\cdots,y_{k+1}}[f(y) = f(y) - T_f^i(y - y_1, y_2, \cdots, y_{k+1}, y_1)] \leqslant 1/2$. If the fraction of such elements is more than $2\eta_i$ then that contradicts the condition that

$$\begin{aligned}
\eta_i &= \Pr_{y_1,\cdots,y_{k+1},b}[T_f^i(y_1, \cdots, y_{k+1}, b) \neq 0] \\
&= \Pr_{y_1,y_2,\cdots,y_{k+1},b}[T_f^i(y_1 - b, y_2, \cdots, y_{k+1}, b) \neq 0] \\
&= \Pr_{y,y_1,\cdots,y_{k+1}}[f(y) \neq f(y) - T_f^i(y - y_1, y_2, \cdots, y_{k+1}, y_1)].
\end{aligned}$$

Therefore, we obtain $\delta(f, g_i) \leqslant 2\eta_i$. □

Note that $\Pr_{y_1,\cdots,y_{k+1}}[g_i(y) = f(y) - T_f^i(y - y_1, y_2, \cdots, y_{k+1}, y_1)] \geqslant \frac{1}{p}$. We now show that this probability is actually much higher. The next lemma gives a weak bound in that direction following the analysis in [93]. For the sake of completeness, we present the proof.

Lemma 7.10. $\forall y \in \mathbb{F}_p^n$, $\Pr_{y_1,\cdots,y_{k+1} \in \mathbb{F}_p^n}[g_i(y) = f(y) - T_f^i(y - y_1, y_2, \cdots, y_{k+1}, y_1)] \geqslant 1 - 2p^{k+1}\eta_i$.

Proof. We will use I, J, I', J' to denote $(k+1)$ dimensional vectors over $\mathbb{F}_p$. Now note that

$$\begin{aligned} -g_i(y) &= \text{Plurality}_{y_1,\cdots,y_{k+1}\in\mathbb{F}_p^n}[\sum_{I\in\mathbb{F}_p^{k+1};I\neq\langle 1,0,\cdots,0\rangle} I_1^i f(I_1(y-y_1)+\sum_{t=2}^{k+1} I_t y_t + y_1)] \\ &= \text{Plurality}_{y-y_1,y_2,\cdots,y_{k+1}\in\mathbb{F}_p^n}[\sum_{I\in\mathbb{F}_p^{k+1};I\neq\langle 0,\cdots,0\rangle} (I_1+1)^i f(I_1(y-y_1) \\ &\quad + \sum_{t=2}^{k+1} I_t y_t + y)] \\ &= \text{Plurality}_{y_1,\cdots,y_{k+1}\in\mathbb{F}_p^n}[\sum_{I\in\mathbb{F}_p^{k+1};I\neq\langle 0,\cdots,0\rangle} (I_1+1)^i f(\sum_{t=1}^{k+1} I_t y_t + y)] \end{aligned} \tag{7.12}$$

Let $Y = \langle y_1, \cdots, y_{k+1}\rangle$ and $Y' = \langle y'_1, \cdots, y'_{k+1}\rangle$. Also we will denote $\langle 0, \cdots, 0\rangle$ by $\mathbf{0}$. Now note that

$$\begin{aligned} 1-\eta_i \leqslant\ & \Pr_{y_1,\cdots,y_{k+1},b}[T_f^i(y_1,\cdots,y_{k+1},b)=0] \\ =\ & \Pr_{y_1,\cdots,y_{k+1},b}[\sum_{I\in\mathbb{F}_p^{k+1}} I_1^i f(b+I\cdot Y)=0] \\ =\ & \Pr_{y_1,\cdots,y_{k+1},b}[f(b+y_1)+\sum_{I\in\mathbb{F}_p^{k+1};I\neq\langle 1,0,\cdots,0\rangle} I_1^i f(b+I\cdot Y)=0] \\ =\ & \Pr_{y_1,\cdots,y_{k+1},y}[f(y)+\sum_{I\in\mathbb{F}_p^{k+1};I\neq\langle 1,0,\cdots,0\rangle} I_1^i f(y-y_1+I\cdot Y)=0] \\ =\ & \Pr_{y_1,\cdots,y_{k+1},y}[f(y)+\sum_{I\in\mathbb{F}_p^{k+1};I\neq\langle 0,\cdots,0\rangle} (I_1+1)^i f(y+I\cdot Y)=0] \end{aligned}$$

Therefore for any given $I \neq \mathbf{0}$ we have the following:

$$\Pr_{Y,Y'}[f(y+I\cdot Y) = \sum_{J\in\mathbb{F}_p^{k+1};J\neq\mathbf{0}} -(J_1+1)^i f(y+I\cdot Y+J\cdot Y')] \geqslant 1-\eta_i$$

and for any given $J \neq \mathbf{0}$,

$$\Pr_{Y,Y'}[f(y+J\cdot Y') = \sum_{I\in\mathbb{F}_p^{k+1};I\neq\mathbf{0}} -(I_1+1)^i f(y+I\cdot Y+J\cdot Y')] \geqslant 1-\eta_i.$$

Combining the above two and using the union bound we get,

$$\begin{aligned}
&\Pr_{Y,Y'}\left[\sum_{I\in\mathbb{F}_p^{k+1};I\neq\mathbf{0}}(I_1+1)^i f(y+I\cdot Y)\right] \\
&\quad=\sum_{I\in\mathbb{F}_p^{k+1};I\neq\mathbf{0}}\ \sum_{J\in\mathbb{F}_p^{k+1};J\neq\mathbf{0}} -(I_1+1)^i(J_1+1)^i f(y+I\cdot Y+J\cdot Y') \\
&\quad=\sum_{J\in\mathbb{F}_p^{k+1};J\neq\mathbf{0}}(J_1+1)^i f(y+J\cdot Y')] \\
&\quad\geqslant 1-2(p^{k+1}-1)\eta\geqslant 1-2p^{k+1}\eta_i
\end{aligned} \tag{7.13}$$

The lemma now follows from the observation that the probability that the same object is drawn from a set in two independent trials lower bounds the probability of drawing the most likely object in one trial: Suppose the objects are ordered so that p_i is the probability of drawing object i, and $p_1 \geqslant p_2 \geqslant \cdots$. Then the probability of drawing the same object twice is $\sum_i p_i^2 \leqslant \sum_i p_1 p_i \leqslant p_1$. □

However, when the degree being tested is larger than the field size, we can improve the above lemma considerably. The following lemma strengthens Lemma 7.10 when $t \geqslant p-1$ or equivalently $k \geqslant 1$. We now focus on the $\mathbb{F}_3$ case. The proof appears in Section 7.4.3.

Lemma 7.11. $\forall y \in \mathbb{F}_3^n$, $\Pr_{y_1,\cdots,y_{k+1}\in\mathbb{F}_3^n}[g_i(y) = f(y) - T_f^i(y-y_1,y_2,\cdots,y_{k+1},y_1)] \geqslant 1-(4k+14)\eta_i$.

Lemma 7.11 will be instrumental in proving the next lemma, which shows that sufficiently small η_i implies g_i is the self-corrected version of the function f (the proof appears in Section 7.4.4).

Lemma 7.12. *Over $\mathbb{F}_3$, if $\eta_i < \frac{1}{(4k+14)3^{k+1}}$, then the function g_i belongs to $\mathcal{P}_t$ (assuming $k \geqslant 1$).*

By combining Lemma 7.9 and Lemma 7.12 we obtain that if f is $\Omega(1/(k3^k))$-far from $\mathcal{P}_t$ then η_i is at least $\Omega(1/(k3^k))$. We next consider the case in which η_i is small. By Lemma 7.9, in this case, the distance $\delta = \delta(f,g)$ is small. The next lemma shows that in this case the test rejects f with probability that is close to $3^{k+1}\delta$. This follows from the fact that in this case, the probability over the selection of $y_1,\cdots,y_{k+1},b$, that among the 3^{k+1} points $\sum_i c_i y_i + b$ (where $c_1,\ldots,c_{k+1}\in\mathbb{F}_3$), the functions f and g differ in precisely one point, is close to $3^{k+1}\cdot\delta$. Observe that if they do, then the test rejects.

Lemma 7.13. *Suppose $0 \leqslant \eta_i \leqslant \frac{1}{(4k+14)3^{k+1}}$. Let δ denote the relative distance between f and g and $\ell = 3^{k+1}$. Then, when $y_1,\cdots,y_{k+1},b$ are chosen randomly, the probability that for exactly one point v among the ℓ points $\sum_i c_i y_i + b$ (where $(c_1,\ldots,c_{k+1})\in\mathbb{F}_3^{k+1}$), $f(v)\neq g(v)$ is at least $\left(\frac{1-\ell\delta}{1+\ell\delta}\right)\ell\delta$.*

Observe that η_i is at least $\Omega(3^{k+1}\delta)$. The proof of Lemma 7.13 is deferred to Section 7.4.5.

Proof of Theorem 7.4: Clearly if f belongs to $\mathcal{P}_t$, then by Claim 7.1 the tester accepts f with probability 1.

Therefore let $\delta(f, \mathcal{P}_t) \geqslant \varepsilon$. Let $d = \delta(f, g_r)$, where r is as in algorithm **Test-$\mathcal{P}_t$**. If $\eta < \frac{1}{(4k+14)3^{k+1}}$ then by Lemma 7.12 $g_r \in \mathcal{P}_t$ and, by Lemma 7.13, η_i is at least $\Omega(3^{k+1} \cdot d)$, which by the definition of ε is at least $\Omega(3^{k+1}\varepsilon)$. Hence $\eta_i \geqslant min\left(c(3^{k+1}\varepsilon), \frac{1}{(4k+14)3^{k+1}}\right)$, for some fixed constant $c > 0$. □

Remark 7.4. *Theorem 7.2 follows from a similar argument.*

7.4.3 Proof of Lemma 7.11

Observe that the goal of Lemma 7.11 is to show that at any fixed point y, if g_i is interpolated from a random hyperplane, then w.h.p. the interpolated value is the most popular vote. To ensure this we show that if g_i is interpolated on two independently random hyperplanes, then the probability that these interpolated values are the same, that is the collision probability, is large. To estimate this collision probability, we show that the difference of the interpolation values can be rewritten as a sum of T_f^i on small number of random hyperplanes. Thus if the test passes often (that is, T_f^i evaluates to zero w.h.p.), then this sum (by a simple union bound) evaluates to zero often, which proves the high collision probability.

The improvement will arise because we will express differences involving $T_f^i(\cdots)$ as a telescoping series to essentially reduce the number of events in the union bound. To do this we will need the following claims. We note that a similar claim for $p = 2$ was proven by expanding the terms on both sides in [1]. However, the latter does not give much insight into the general case i.e., for $\mathbb{F}_p$. We provide proofs that have a much cleaner structure based on the underlying geometric structure, i.e., flats or pseudoflats.

Claim 7.14. *For every $l \in \{2, \cdots, k+1\}$, for every $y(= y_1), z, w, b, y_2, \cdots, y_{l-1}, y_{l+1}, \cdots, y_{k+1} \in \mathbb{F}_p^n$, let let*

$$S_f^l(y, z) \stackrel{def}{=} T_f^0(y, y_2, \cdots, y_{l-1}, z, y_{l+1}, \cdots, y_{k+1}, b).$$

The the following holds:

$$S_f^l(y, w) - S_f^l(y, z) = \sum_{e \in \mathbb{F}_p^*} \left[S_f^l(y + ew, z) - S_f^l(y + ez, w)\right].$$

Proof. Assume y, z, w are independent. If not then both sides are equal to 0 and hence the equality is trivially satisfied. To see why this claim is true for the left hand side, recall the definition of $T_f^0(\cdot)$ and note that the sets of points in the flat generated by $y, y_2, \cdots, y_{l-1}, w$, $y_{l+1}, \cdots, y_{k+1}$ at b and the flat generated by $y, y_2, \cdots, y_{l-1}, z, y_{l+1}, \cdots, y_{k+1}$ at b are the same. A similar argument works for the expression on the right hand side of the equality.

We claim that it is enough to prove the result for $k = 1$ and $b = \mathbf{0}$. A linear transform (or renaming the co-ordinate system appropriately) reduces the case of $k = 1$ and $b \neq \mathbf{0}$ to the case of $k = 1$ and $b = \mathbf{0}$. We now show how to reduce the case of $k > 1$ to the $k = 1$ case. Fix some values $c_2, \cdots, c_{l-1}, c_{l+1}, \cdots, c_{k+1}$ and note that one can write $c_1 y + c_2 y_2 + \cdots c_{l-1} y_{l-1} + c_l w + c_{l+1} y_{l+1} + c_{k+1} y_{k+1} + b$ as $c_1 y + c_l w + b'$, where $b' = \sum_{j \in \{2,\cdots,l-1,l+1,\cdots,k+1\}} c_j y_j + b$. Thus,

$$S_f^l(y, w) = \sum_{(c_2,\cdots,c_{l-1},c_{l+1},\cdots,c_{k+1}) \in \mathbb{F}_p}^{k-1} \sum_{(c_1,c_l) \in \mathbb{F}_p^2} f(c_1 y + c_l w + b').$$

One can rewrite the other $S_f^l(\cdot)$ terms similarly. Note that for a fixed vector $(c_2, \cdots, c_{l-1}, c_{l+1}, \cdots, c_{k+1})$, the value of b' is the same. Finally note that the equality in the general case is satisfied if p^{k-1} similar equalities hold in the $k = 1$ case.

Now consider the space $\mathcal{H}$ generated by y, z and w at $\mathbf{0}$. Note that $S_f^l(y, w)$ (with $b = \mathbf{0}$) is just $f \cdot 1_L$, where 1_L is the incidence vector of the flat given by the equation $z = 0$. Therefore 1_L is equivalent to the polynomial $(1 - z^{p-1})$ over $\mathbb{F}_p$. Similarly $S_f^l(y, z) = f \cdot 1_{L'}$ where L' is given by the polynomial $(1 - w^{p-1})$ over $\mathbb{F}_p$. We use the following polynomial identity (in $\mathbb{F}_p$)

$$w^{p-1} - z^{p-1} = \sum_{e \in \mathbb{F}_p^*} \left[[1 - (ew + y)^{p-1}] - [1 - (ez + y)^{p-1}] \right] \tag{7.14}$$

Now observe that the polynomial $(1 - (ew + y)^{p-1})$ is the incidence vector of the flat generated by $y - e^{-1}w$ and z. Similarly, the polynomial $(1 - (ez + y)^{p-1})$ is the incidence vector of the flat generated by $y - e^{-1}z$ and w. Therefore, interpreting the above equation in terms of incidence vectors of flats, Observation 7.4 completes the proof assuming (7.14) is true.

We complete the proof by proving (7.14). Consider the sum: $\sum_{e \in \mathbb{F}_p^*} (ew + y)^{p-1}$. Expanding the terms and rearranging the sums we get $\sum_{j=0}^{p-1} \binom{p-1}{j} w^{p-1-j} y^j \sum_{e \in \mathbb{F}_p^*} e^{p-1-j}$. By Lemma 7.1 the sum evaluates to $(-w^{p-1} - y^{p-1})$. Similarly, $\sum_{e \in \mathbb{F}_p^*} (ez + y)^{p-1} = (-z^{p-1} - y^{p-1})$ which proves (7.14). □

We will also need the following claim.

Claim 7.15. *For every $i \in \{1, \cdots, p-2\}$, for every $l \in \{2, \cdots, k+1\}$ and for every $y(= y_1), z, w, b, y_2, \cdots, y_{l-1}, y_{l+1}, \cdots, y_{k+1} \in \mathbb{F}_p^n$, denote*

$$S_f^{i,l}(y, w) \stackrel{def}{=} T_f^i(y, y_2, \cdots, y_{l-1}, w, y_{l+1}, \cdots, y_{k+1}, b).$$

Then there exists c_i such that

$$S_f^{i,l}(y, w) - S_f^{i,l}(y, z) = c_i \sum_{e \in \mathbb{F}_p^*} \left[S_f^{i,l}(y + ew, z) - S_f^{i,l}(y + ez, w) \right].$$

Proof. As in the proof of Claim 7.14, we only need to prove the claim for $k = 1$ and $b = \mathbf{0}$. Observe that $S_f^{i,l}(y,z) = f \cdot E_{L_i}$, where E_{L_i} denotes the $(2,i)$-pseudoflat vector of the pseudoflat L generated by y, z at b exponentiated along y. Note that the polynomial defining E_{L_i} is just $y^i(w^{p-1}-1)$. Similarly we can identify the other terms with polynomials over $\mathbb{F}_p$. To complete the proof, we need to prove the following identity (which is similar to the one in (7.14)):

$$y^i(w^{p-1} - z^{p-1}) = c_i \sum_{e \in \mathbb{F}_p^*} \left[(y+ew)^i[1-(y-ew)^{p-1}] - (y+ez)^i[1-(y-ez)^{p-1}]\right]. \tag{7.15}$$

where $c_i = 2^i$. Before we prove the identity, note that $(-1)^j\binom{p-1}{j} = 1$ in $\mathbb{F}_p$. This is because for $1 \leqslant m \leqslant j$, $m = (-1)(p-m)$. Therefore $j! = (-1)^j \frac{(p-1)!}{(p-j-1)!}$ holds in $\mathbb{F}_p$. Substitution yields the desired result. Also note that $\sum_{e\in\mathbb{F}_p^*}(y+ew)^i = -y^i$ (expand and apply Lemma 7.1). Now consider the sum

$$\begin{aligned}
\sum_{e\in\mathbb{F}_p^*}(y+ew)^i(y-ew)^{p-1} &= \sum_{e\in\mathbb{F}_p^*}\sum_{\substack{0\leqslant j\leqslant i;\\ 0\leqslant m\leqslant p-1}} (-1)^m\binom{i}{j}\binom{p-1}{m} y^{p-1+i-j-m}w^{j+m}e^{j+m} \\
&= \sum_{\substack{0\leqslant j\leqslant i;\\ 0\leqslant m\leqslant p-1}} (-1)^m\binom{i}{j}\binom{p-1}{m} y^{p-1+i-j-m}w^{j+m}\sum_{e\in\mathbb{F}_p^*} e^{j+m} \\
&= -y^{p-1+i} + (-1)^p\sum_{j=0}^{i}\binom{i}{j}\underbrace{\binom{p-1}{p-1-j}(-1)^j}_{=1} y^i w^{p-1}] \\
&= (-1)[y^i + y^i w^{p-1}2^i] \qquad (7.16)
\end{aligned}$$

Similarly one has $\sum_{e\in\mathbb{F}_p^*}(y+ez)^i(y-ez)^{p-1} = (-1)[y^i + y^i z^{p-1}2^i]$. Substituting and simplifying one gets (7.15). □

Finally, we will also need the following claim.

Claim 7.16. *For every $i \in \{1,\cdots,p-2\}$, for every $l \in \{2,\cdots,l+1\}$ and for every $y(=y_l), z, w, b, y_2, \cdots, y_{l-1}, y_{l+1}, \cdots, y_{k+1} \in \mathbb{F}_p^n$, there exists $c_i \in \mathbb{F}_p^*$ such that*

$$\begin{aligned}
S_f^{i,l}(w,y) - S_f^{i,l}(z,y) = \sum_{e\in\mathbb{F}_p^*}\Big[&S_f^{i,l}(y+ew, y-ew) - S_f^{i,l}(w+ey, w-ey) + \\
&S_f^{i,l}(z+ey, z-ey) - S_f^{i,l}(y+ez, y-ez) \\
&+c_i\left[S_f^{i,l}(y+ew, z) - S_f^{i,l}(y+ez, w)\right]\Big]
\end{aligned}$$

Proof. As in the proof of Claim 7.15, the proof boils down to proving a polynomial identity over $\mathbb{F}_p$. In particular, we need to prove the following identity over $\mathbb{F}_p$:

$$w^i(1-z^{p-1}) - z^i(1-w^{p-1}) = (w^i-y^i)(1-z^{p-1}) - (z^i-y^i)(1-w^{p-1}) + y^i(w^{p-1}-z^{p-1}).$$

We also use that $\sum_{e\in\mathbb{F}_p^*}(w+ey)^i = -w^i$ and Claim 7.15 to expand the last term. Note that $c_i = 2^i$ as before. □

We need one more simple fact before we can prove Lemma 7.11. For a probability vector $v \in [0,1]^n$, $\left\|v\right\|_\infty = \text{Max}_{i\in[n]}\{v_i\} \geqslant \text{Max}_{i\in[n]}\{v_i\}\cdot(\sum_{i=1}^n v_i) = \sum_{i=1}^n v_i \cdot \text{Max}_{i\in[n]}\{v_i\} \geqslant \sum_{i=1}^n v_i^2 = \left\|v\right\|_2^2$.

Proof of Lemma 7.11: We first prove the lemma for $g_0(y)$. We fix $y \in \mathbb{F}_3^n$ and let $\gamma \stackrel{def}{=} \Pr_{y_1,\cdots,y_{k+1}\in\mathbb{F}_3^n}[g_0(y) = f(y) - T_f^0(y-y_1, y_2, \cdots, y_{k+1}, y_1)]$. Recall that we want to lower bound γ by $1-(4k+14)\eta_0$. In that direction, we bound a slightly different but related probability. Define

$$\mu = \Pr_{y_1,\cdots,y_{k+1},z_1,\cdots,z_{k+1}\in\mathbb{F}_3^n}[T_f^0(y-y_1,y_2,\cdots,y_{k+1},y_1) = T_f^0(y-z_1,z_2,\cdots,z_{k+1},z_1)]$$

Denote $Y = \langle y_1,\cdots,y_{k+1}\rangle$ and similarly Z. Then by the definitions of μ and γ we have, $\gamma \geqslant \mu$. Note that we have

$$\mu = \Pr_{y_1,\cdots,y_{k+1},z_1,\cdots,z_{k+1}\in\mathbb{F}_3^n}[T_f^0(y-y_1,y_2,\cdots,y_{k+1},y_1) - T_f^0(y-z_1,z_2,\cdots,z_{k+1},z_1) = 0].$$

We will now use a hybrid argument. Now, for any choice of $y_1,\cdots,y_{k+1}$ and $z_1,\cdots,z_{k+1}$ we have:

$$\begin{aligned}
&T_f^0(y-y_1,y_2,\cdots,y_{k+1},y_1) - T_f^0(y-z_1,z_2,\cdots,z_{k+1},z_1)\\
&\quad= T_f^0(y-y_1,y_2,\cdots,y_{k+1},y_1) - T_f^0(y-y_1,y_2,\cdots,y_k,z_{k+1},y_1)\\
&\qquad+ T_f^0(y-y_1,y_2,\cdots,y_k,z_{k+1},y_1) - T_f^0(y-y_1,y_2,\cdots,y_{k-1},z_k,z_{k+1},y_1)\\
&\qquad+ T_f^0(y-y_1,\cdots,y_{k-1},z_k,z_{k+1},y_1) - T_f^0(y-y_1,\cdots,y_{k-2},z_{k-1},z_k,z_{k+1},y_1)\\
&\qquad\qquad\vdots\\
&\qquad+ T_f^0(y-y_1,z_2,z_3,\cdots,z_{k+1},y_1) - T_f^0(y-z_1,z_2,\cdots,z_{k+1},y_1)\\
&\qquad+ T_f^0(y-z_1,z_2,z_3,\cdots,z_{k+1},y_1) - T_f^0(y-y_1,z_2,\cdots,z_{k+1},z_1)\\
&\qquad+ T_f^0(y-y_1,z_2,z_3,\cdots,z_{k+1},z_1) - T_f^0(y-z_1,z_2,\cdots,z_{k+1},z_1)
\end{aligned}$$

Consider any pair $T_f^0(y-y_1,y_2,\cdots,y_l,z_{l+1},\cdots,z_{k+1},y_1) - T_f^0(y-y_1,y_2,\cdots,y_{l-1},z_l,\cdots,z_{k+1},y_1)$ that appears in the first k "rows" in the sum above. Note that $T_f^0(y-y_1,y_2,\cdots,y_l,z_{l+1},\cdots,z_{k+1},y_1)$ and $T_f^0(y-y_1,y_2,\cdots,y_{l-1},z_l,\cdots,z_{k+1},y_1)$ differ only in a single parameter. We apply Claim 7.14 and obtain:

$T_f^0(y-y_1,y_2,\cdots,y_l,z_{l+1},\cdots,z_{k+1},y_1) - T_f^0(y-y_1,y_2,\cdots,y_{l-1},z_l,\cdots,z_{k+1},y_1) = T_f^0(y-y_1+y_l,y_2,\cdots,y_{l-1},z_l,\cdots,z_{k+1},y_1) + T_f^0(y-y_1-y_l,y_2,\cdots,y_{l-1},z_l,\cdots,z_{k+1},y_1) - T_f^0(y-y_1+z_l,y_2,\cdots,y_l,z_{l+1},\cdots,z_{k+1},y_1) - T_f^0(y-y_l-z_l,y_2,\cdots,y_l,z_{l+1},\cdots,z_{k+1},y_1)$.

Recall that y is fixed and $y_2,\cdots,y_{k+1},z_2,\cdots,z_{k+1} \in \mathbb{F}_3^n$ are chosen uniformly at random, so all the parameters on the right hand side of the equation are independent and

uniformly distributed. Similarly one can expand the pairs $T_f^0(y-y_1, z_2, z_3, \cdots, z_{k+1}, y_1) - T_f^0(y-z_1, z_2, \cdots, z_{k+1}, y_1)$ and $T_f^0(y-y_1, z_2, z_3, \cdots, z_{k+1}, z_1) - T_f^0(y-z_1, z_2, \cdots, z_{k+1}, z_1)$ into four T_f^0 with all parameters being independent and uniformly distributed[3]. Finally notice that the parameters in both $T_f^0(y-z_1, z_2, z_3, \cdots, z_{k+1}, y_1)$ and $T_f^0(y-z_1, z_2, \cdots, z_{k+1}, y_1)$ are independent and uniformly distributed. Further recall that by the definition of η_0, $\Pr_{r_1,\cdots,r_{k+1}}[T_f^0(r_1, \cdots, r_{k+1}) \neq 0] \leqslant \eta_0$ for independent and uniformly distributed r_is. Thus, by the union bound, we have:

$$\Pr_{y_1,\cdots,y_{k+1},z_1,\cdots,z_{k+1}\in\mathbb{F}_3^n}[T_f^0(y_1, \cdots, y_{k+1}) - T_f^0(z_1, \cdots, z_{k+1}) \neq 0] \leqslant (4k+10)\eta_0. \quad (7.17)$$

Therefore $\gamma \geqslant \mu \geqslant 1-(4k+10)\eta_0$. A similar argument proves the lemma for $g_1(y)$. The only catch is that $T_{f_1}(.)$ is not symmetric– in particular in its first argument. Thus, we use another identity as given in Claim 7.16 to resolve the issue and get four extra terms than in the case of g_0, which results in the claimed bound of $(4k+14)\eta_i$. □

Remark 7.5. *Analogously, in the case $\mathbb{F}_p$ we have: for every $y \in \mathbb{F}_p^n$, $\Pr_{y_1,y_2,\cdots,y_{k+1}\in\mathbb{F}_p^n}[g_i(y) = f(y) - T_f^i(y-y_1, y_2, \cdots, y_{k+1}, y_1) + f(y)] \geqslant 1-2((p-1)k+6(p-1)+1)\eta_i$.*
The proof is similar to that of Lemma 7.11 where it can be shown $\mu_i \geqslant 1-2((p-1)k+6(p-1)+1)\eta_i$, for each μ_i defined for $g_i(y)$.

7.4.4 Proof of Lemma 7.12

From Theorem 7.1, it suffices to prove that if $\eta_i < \frac{1}{(4k+14)3^{k+1}}$ then $T_{g_i}^i(y_1, \cdots, y_{k+1}, b) = 0$ for every $y_1, \cdots, y_{k+1}, b \in \mathbb{F}_3^n$. Fix the choice of $y_1, \cdots, y_{k+1}, b$. Define $Y = \langle y_1, \cdots, y_{k+1}\rangle$. We will express $T_{g_i}^i(Y, b)$ as the sum of $T_f^i(\cdot)$ with random arguments. We uniformly select $(k+1)^2$ random variables $z_{i,j}$ over $\mathbb{F}_3^n$ for $1 \leqslant i \leqslant k+1$, and $1 \leqslant j \leqslant k+1$. Define $Z_i = \langle z_{i,1}, \cdots, z_{i,k+1}\rangle$. We also select uniformly $(k+1)$ random variables r_i over $\mathbb{F}_3^n$ for $1 \leqslant i \leqslant k+1$. We use $z_{i,j}$ and r_i's to set up the random arguments. Now by Lemma 7.11, for every $I \in \mathbb{F}_3^{k+1}$ (i.e. think of I as an ordered $(k+1)$-tuple over $\{0,1,2\}$), with probability at least $1-(4k+14)\eta_i$ over the choice of $z_{i,j}$ and r_i,

$$g_i(I\cdot Y+b) = f(I\cdot Y+b) - T_f^i(I\cdot Y+b-I\cdot Z_1-r_1, I\cdot Z_2+r_2, \cdots, I\cdot Z_{k+1}+r_{k+1}, I\cdot Z_1+r_1), \quad (7.18)$$

where for vectors $X, Y \in \mathbb{F}_3^{k+1}$, $Y\cdot X = \sum_{i=1}^{k+1} Y_i X_i$, holds.

Let E_1 be the event that (7.18) holds for all $I \in \mathbb{F}_3^{k+1}$. By the union bound:

$$\Pr[E_1] \geqslant 1 - 3^{k+1}\cdot(4k+14)\eta_i. \quad (7.19)$$

Assume that E_1 holds. We now need the following claims. Let $J = \langle J_1, \cdots, J_{k+1}\rangle$ be a $(k+1)$ dimensional vector over $\mathbb{F}_3$, and denote $J' = \langle J_2, \cdots, J_{k+1}\rangle$.

---

[3]Since $T_f^0(\cdot)$ is symmetric in all but its last argument.

Claim 7.17. *If (7.18) holds for all $I \in \mathbb{F}_3^{k+1}$, then*

$$
\begin{aligned}
T^0_{g_0}(Y,b) = & \sum_{0\neq J'\in\mathbb{F}_3^k}\left[-T^0_f(y_1+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,y_{k+1}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},b+\sum_{t=2}^{k+1}J_tr_t)\right]\\
& +\sum_{J'\in\mathbb{F}_3^k}\Bigg[-T^0_f(2y_1-z_{1,1}+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,2y_{k+1}-z_{1,(k+1)}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},\\
& \qquad 2b-r_1+\sum_{t=2}^{k+1}J_tr_t)\\
& \qquad +T^0_f\ (z_{1,1}+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,z_{1,k+1}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},r_1+\sum_{t=2}^{k+1}J_tr_t)\Bigg]
\end{aligned}
\tag{7.20}
$$

Proof.

$$
\begin{aligned}
T^0_{g_0}(Y,b) &= \sum_{I\in\mathbb{F}_3^{k+1}} g_0(I\cdot Y+b)\\
&= \sum_{I\in\mathbb{F}_3^{k+1}}\big[-T^0_f(I\cdot Y+b-I\cdot Z_1-r_1,I\cdot Z_2+r_2,\cdots,I\cdot Z_{k+1}+r_{k+1},\\
&\qquad I\cdot Z_1+r_1)+f(I\cdot Y+b)\big]\\
&= -\sum_{I\in\mathbb{F}_3^{k+1}}\Bigg[\Bigg[\sum_{\emptyset\neq J'\in\mathbb{F}_3^k} f(I\cdot Y+b+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\Bigg]\\
&\quad +\Bigg[\sum_{J'\in F_3^k}\Bigg(f(2I\cdot Y+2b-I\cdot Z_1-r_1+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\\
&\quad +f(I\cdot Z_1+r_1+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\Bigg)\Bigg]\Bigg]\\
&= -\sum_{0\neq J'\in\mathbb{F}_3^k}\Bigg[\sum_{I\in\mathbb{F}_3^{k+1}} f(I\cdot Y+b+\sum_{t=2}^{k+1}J_tr_t+\sum_{t=2}^{k+1}J_tI\cdot Z_t)\Bigg]\\
&\quad -\sum_{J'\in\mathbb{F}_3^k}\Bigg[\Bigg[\sum_{I\in\mathbb{F}_3^{k+1}} f(2I\cdot Y+2b-I\cdot Z_1-r_1+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\Bigg]\\
&\quad +\Bigg[\sum_{I\in\mathbb{F}_3^{k+1}} f(I\cdot Z_1+r_1+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\Bigg]\Bigg]
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{0\neq J'\in\mathbb{F}_3^k}\left[-T_f^0(y_1+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,y_{k+1}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},b+\sum_{t=2}^{k+1}J_tr_t)\right]\\
&+ \sum_{J'\in\mathbb{F}_3^k}\Bigg[-T_f^0(2y_1-z_{1,1}+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,2y_{k+1}-z_{1,(k+1)}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},\\
&\qquad 2b-r_1+\sum_{t=2}^{k+1}J_tr_t)\\
&+ T_f^0(z_{1,1}+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,z_{1,k+1}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},r_1+\sum_{t=2}^{k+1}J_tr_t)\Bigg]
\end{aligned}
$$

□

Claim 7.18. *If (7.18) holds for all $I\in\mathbb{F}_3^{k+1}$, then*

$$
\begin{aligned}
T_{g_1}^1(Y,b) = &\sum_{0\neq J'\in\mathbb{F}_3^k}\left[-T_f^1(y_1+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,y_{k+1}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},b+\sum_{t=2}^{k+1}J_tr_t)\right]\\
&+\sum_{J'\in\mathbb{F}_3^k}\Bigg[T_f^1(2y_1-z_{1,1}+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,2y_{k+1}-z_{1,(k+1)}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},\\
&\qquad 2b-r_1+\sum_{t=2}^{k+1}J_tr_t)\Bigg]. \qquad (7.21)
\end{aligned}
$$

Proof.

$$
\begin{aligned}
T_{g_1}^1(Y,b) &= \sum_{I\in\mathbb{F}_3^{k+1}} I_1g_1(I\cdot Y+b)\\
&= \sum_{I\in\mathbb{F}_3^{k+1}} I_1\left[-T_f^1(I\cdot Y+b-I\cdot Z_1-r_1,I\cdot Z_2+r_2,\cdots,I\cdot Z_{k+1}+r_{k+1},\right.\\
&\qquad \left. I\cdot Z_1+r_1)+f(I\cdot Y+b)\right]\\
&= -\sum_{I\in\mathbb{F}_3^{k+1}} I_1\Bigg[\Bigg[\sum_{\emptyset\neq J'\in\mathbb{F}_3^k} f(I\cdot Y+b+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\Bigg]\\
&\quad +\Bigg[\sum_{J'\in F_3^k} f(2I\cdot Y+2b-I\cdot Z_1-r_1+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\Bigg]\Bigg]\\
&= -\sum_{0\neq J'\in\mathbb{F}_3^k}\left[\sum_{I\in\mathbb{F}_3^{k+1}} I_1f(I\cdot Y+b+\sum_{t=2}^{k+1}J_tr_t+\sum_{t=2}^{k+1}J_tI\cdot Z_t)\right]
\end{aligned}
$$

$$
\begin{aligned}
& - \sum_{J' \in \mathbb{F}_3^k} \left[\sum_{I \in \mathbb{F}_3^{k+1}} I_1 f(2I \cdot Y + 2b - I \cdot Z_1 - r_1 + \sum_{t=2}^{k+1} J_t I \cdot Z_t + \sum_{t=2}^{k+1} J_t r_t) \right] \\
= & \sum_{0 \neq J' \in \mathbb{F}_3^k} \left[-T_f^1(y_1 + \sum_{t=2}^{k+1} J_t z_{t,1}, \cdots, y_{k+1} + \sum_{t=2}^{k+1} J_t z_{t,(k+1)}, b + \sum_{t=2}^{k+1} J_t r_t) \right] \\
+ & \sum_{J' \in \mathbb{F}_3^k} \Bigg[T_f^1(2y_1 - z_{1,1} + \sum_{t=2}^{k+1} J_t z_{t,1}, \cdots, 2y_{k+1} - z_{1,(k+1)} + \sum_{t=2}^{k+1} J_t z_{t,(k+1)}, \\
& \qquad 2b - r_1 + \sum_{t=2}^{k+1} J_t r_t) \Bigg]
\end{aligned}
$$

□

Let E_2 be the event that for every $J' \in \mathbb{F}_3^k$, $T_f^i(y_1 + \sum_t J_t z_{t,1}, \cdots, y_{k+1} + \sum_t J_t z_{t,(k+1)}, b + \sum_{t=2} k+1 J_t r_t) = 0$, $T_f^i(2y_1 - z_{1,1} + \sum_{t=2}^{k+1} J_t z_{t,1}, \cdots, 2y_{k+1} - z_{1,k+1} + \sum_{t=2}^{k+1} J_t z_{t,(k+1)}, 2b - r_1 + \sum_{t=2}^{k+1} J_t r_t) = 0$, and $T_f^0(z_{1,1} + \sum_{t=2}^{k+1} J_t z_{t,1}, \cdots, z_{1,k+1} + \sum_{t=2}^{k+1} J_t z_{t,k+1}, r_1 + \sum_{t=2}^{k+1} J_t r_t) = 0$. By the definition of η_i and the union bound, we have:

$$\Pr[E_2] \geqslant 1 - 3^{k+1}\eta_i. \tag{7.22}$$

Suppose that $\eta_i \leqslant \frac{1}{(4k+14)3^{k+1}}$ holds. Then by (7.19) and (7.22), the probability that E_1 and E_2 hold is strictly positive. In other words, there exists a choice of the $z_{i,j}$'s and r_i's for which all summands in either Claim 7.17 or in Claim 7.18, whichever is appropriate, is 0. This implies that $T_{g_i}^i(y_1, \cdots, y_{k+1}, b) = 0$. In other words, if $\eta_i \leqslant \frac{1}{(4k+14)3^{k+1}}$, then g_i belongs to $\mathcal{P}_t$. □

Remark 7.6. *Over $\mathbb{F}_p$ we have: if $\eta_i < \frac{1}{2((p-1)k+6(p-1)+1)p^{k+1}}$, then g_i belongs to $\mathcal{P}_t$ (if $k \geqslant 1$).*

In case of $\mathbb{F}_p$, we can generalize (7.18) in a straightforward manner. Let E_1' denote the event that all such events holds. We can similarly obtain

$$\Pr[E_1'] \geqslant 1 - p^{k+1} \cdot 2((p-1)k + 6(p-1) + 1)\eta_i. \tag{7.23}$$

Claim 7.19. *Assume equivalent of (7.18) holds for all $I \in \mathbb{F}_p^{k+1}$, then*

$$\begin{aligned} T^i_{g_i}(Y,b) &= \sum_{0\neq J'\in\mathbb{F}_p^k}\left[-T^i_f(y_1+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,y_{k+1}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},b+\sum_{t=2}^{k+1}J_tr_t)\right] \\ &+ \sum_{J'\in\mathbb{F}_p^k}\left[\sum_{J_1\in\mathbb{F}_p;J_1\neq 1}J_1^i\left[-T^i_f(J_1y_1-(J_1-1)z_{1,1}+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,J_1y_{k+1}-\right.\right. \\ &\qquad \left.\left.(J_1-1)z_{1,(k+1)}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},J_1b-(J_1-1)r_1+\sum_{t=2}^{k+1}J_tr_t)\right]\right] \end{aligned} \tag{7.24}$$

Proof.

$$\begin{aligned} T^i_{g_i}(Y,b) &= \sum_{I\in\mathbb{F}_p^{k+1}} I_1^i g_i(I\cdot Y+b) \\ &= \sum_{I\in\mathbb{F}_p^{k+1}} I_1^i\left[-T^i_f(I\cdot Y+b-I\cdot Z_1-r_1,I\cdot Z_2+r_2,\cdots,I\cdot Z_{k+1}+r_{k+1},\right. \\ &\qquad \left. I\cdot Z_1+r_1)+f(I\cdot Y+b)\right] \\ &= -\sum_{I\in\mathbb{F}_p^{k+1}} I_1^i\left[\left[\sum_{\emptyset\neq J'\in\mathbb{F}_p^k} f(I\cdot Y+b+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\right]\right. \\ &+ \left[\sum_{J_1\in\mathbb{F}_p,J_1\neq 1}J_1^i\left[\sum_{J'\in\mathbb{F}_p^k} f(J_1I\cdot Y+J_1b-(J_1-1)I\cdot Z_1-(J_1-1)r_1\right.\right. \\ &\qquad \left.\left.\left.+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\right]\right] \\ &= -\sum_{0\neq J'\in\mathbb{F}_p^k}\left[\sum_{I\in\mathbb{F}_p^{k+1}} I_1^i f(I\cdot Y+b+\sum_{t=2}^{k+1}J_tr_t+\sum_{t=2}^{k+1}J_tI\cdot Z_t)\right] \\ &- \sum_{J'\in\mathbb{F}_p^k}\left[\sum_{J_1\in\mathbb{F}_p;J_1\neq 1}J_1^i\left[\sum_{I\in\mathbb{F}_p^{k+1}} I_1^i f(J_1I\cdot Y+J_1b-(J_1-1)I\cdot Z_1-(J_1-1)r_1\right.\right. \\ &\qquad \left.\left.\left.+\sum_{t=2}^{k+1}J_tI\cdot Z_t+\sum_{t=2}^{k+1}J_tr_t)\right]\right] \\ &= \sum_{0\neq J'\in\mathbb{F}_p^k}\left[-T^i_f(y_1+\sum_{t=2}^{k+1}J_tz_{t,1},\cdots,y_{k+1}+\sum_{t=2}^{k+1}J_tz_{t,(k+1)},b+\sum_{t=2}^{k+1}J_tr_t)\right] \end{aligned}$$

$$+ \sum_{J' \in \mathbb{F}_p^k} \left[\sum_{J_1 \in \mathbb{F}_p; J_1 \neq 1} J_1^i \left[-T_f^i(J_1 y_1 - (J_1 - 1) z_{1,1} + \sum_{t=2}^{k+1} J_t z_{t,1}, \cdots, \quad J_1 y_{k+1} \right. \right.$$
$$\left. \left. -(J_1 - 1) z_{1,(k+1)} + \sum_{t=2}^{k+1} J_t z_{t,(k+1)}, J_1 b - (J_1 - 1) r_1 + \sum_{t=2}^{k+1} J_t r_t \right) \right] \right]$$

□

Let E_2' be the event analogous to the event E_2 in Claim 7.18. Then by the definition of η_i and the union bound, we have

$$\Pr[E_2'] \geqslant 1 - 2p^{k+1}\eta_i. \tag{7.25}$$

Then if we are given that $\eta_i < \frac{1}{2((p-1)k+6(p-1)+1)p^{k+1}}$, then the probability that E_1' and E_2' hold is strictly positive. Therefore, this implies $T_{g_i}^i(y_1, \cdots, y_{k+1}, b) = 0$.

7.4.5 Proof of Lemma 7.13

For each $C \in \mathbb{F}_3^{k+1}$, let X_C be the indicator random variable whose value is 1 if and only if $f(C \cdot Y + b) \neq g(C \cdot Y + b)$, where $Y = \langle y_1, \ldots, y_{k+1} \rangle$. Clearly, $\Pr[X_C = 1] = \delta$ for every C. It follows that the random variable $X = \sum_C X_C$ which counts the number of points v of the required form in which $f(v) \neq g(v)$ has expectation $\mathbb{E}[X] = 3^{k+1}\delta = \ell \cdot \delta$. It is not difficult to check that the random variables X_C are pairwise independent, since for any two distinct $C_1 = (C_{1,1}, \ldots, C_{i,k+1})$ and $C_2 = (C_{2,1}, \ldots, C_{2,k+1})$, the sums $\sum_{i=1}^{k+1} C_{1,i} y_i + b$ and $\sum_{i=1}^{k+1} C_{2,i} y_i + b$ attain each pair of distinct values in $\mathbb{F}_3^n$ with equal probability when the vectors are chosen randomly and independently. Since X_C's are pairwise independent, $\mathrm{Var}[X] = \sum_C \mathrm{Var}[X_C]$. Since X_C's are boolean random variables, we note

$$\mathrm{Var}[X_C] = \mathbb{E}[X_C^2] - (\mathbb{E}[X_C])^2 = \mathbb{E}[X_C] - (\mathbb{E}[X_C])^2 \leqslant \mathbb{E}[X_C].$$

Thus we obtain $\mathrm{Var}[X] \leqslant \mathbb{E}[X]$, so $\mathbb{E}[X^2] \leqslant \mathbb{E}[X]^2 + \mathbb{E}[X]$. Next we use the following well known inequality which holds for a random variable X taking nonnegative, integer values,

$$\Pr[X > 0] \geqslant \frac{(\mathbb{E}[X])^2}{\mathbb{E}[X^2]}.$$

Indeed if X attains value i with probability p_i, then we have

$$(\mathbb{E}[X])^2 = \left(\sum_{i>0} i p_i \right)^2 = \left(\sum_{i>0} i\sqrt{p_i}\sqrt{p_i} \right)^2 \leqslant \left(\sum_{i>0} i p_i \right) \left(\sum_{i>0} p_i \right) = \mathbb{E}[X] \cdot \Pr[X > 0],$$

where the inequality follows by the Cauchy-Schwartz inequality. In our case, this implies

$$\Pr[X > 0] \geqslant \frac{(\mathbb{E}[X])^2}{\mathbb{E}[X^2]} \geqslant \frac{(\mathbb{E}[X])^2}{\mathbb{E}[X] + (\mathbb{E}[X])^2} = \frac{\mathbb{E}[X]}{1 + \mathbb{E}[X]}.$$

Therefore,

$$\mathbb{E}[X] \geqslant \Pr[X=1] + 2\Pr[X \geqslant 2] = \Pr[X=1] + 2\left(\frac{\mathbb{E}[X]}{1+\mathbb{E}[X]} - \Pr[X=1]\right)$$
$$= \frac{2\mathbb{E}[X]}{1+\mathbb{E}[X]} - \Pr[X=1].$$

After simplification we obtain,

$$\Pr[X=1] \geqslant \frac{1-\mathbb{E}[X]}{1+\mathbb{E}[X]} \cdot \mathbb{E}[X].$$

The proof is complete by recalling that $\mathbb{E}[X] = \ell \cdot \delta$. □

7.5 A Lower Bound and Improved Self-correction

7.5.1 A Lower Bound

The next theorem is a simple modification of a theorem in [1] and essentially implies that our result is almost optimal.

Proposition 7.20. *Let $\mathcal{F}$ be any family of functions $f : \mathbb{F}_p^n \to \mathbb{F}_p$ that corresponds to a linear code $\mathcal{C}$. Let d denote the minimum distance of the code $\mathcal{C}$ and let $\bar{d}$ denote the minimum distance of the dual code of $\mathcal{C}$.*
Every one-sided testing algorithm for the family $\mathcal{F}$ must perform $\Omega(\bar{d})$ queries, and if the distance parameter ε is at most d/p^{n+1}, then $\Omega(1/\varepsilon)$ is also a lower bound for the necessary number of queries.

Lemma 7.4 and Proposition 7.20 gives us the following corollary.

Corollary 7.5. *Every one-sided tester for testing $\mathcal{P}_t$ with distance parameter ε must perform $\Omega(max(\frac{1}{\varepsilon}, (1 + ((t+1) \bmod (p-1)))p^{\frac{t+1}{p-1}}))$ queries.*

7.5.2 Improved Self-correction

From Lemmas 7.9, 7.11 and 7.12 the following corollary is immediate:

Corollary 7.6. *Consider a function $f : \mathbb{F}_3^n \to \mathbb{F}_3$ that is ε-close to a degree-t polynomial $g : \mathbb{F}_3^n \to \mathbb{F}_3$, where $\varepsilon < \frac{1}{(4k+14)3^{k+1}}$. (Assume $k \geqslant 1$.) Then the function f can be self-corrected. That is, for any given $x \in \mathbb{F}_3^n$, it is possible to obtain the value $g(x)$ with probability at least $1 - 3^{k+1}\varepsilon$ by querying f on 3^{k+1} points on $\mathbb{F}_3^n$.*

An analogous result may be obtained for the general case. We, however, improve the above corollary slightly. The above corrector does not allow any error in the 3^{k+1} points it queries. We obtain a stronger result by querying on a slightly larger flat H, but allowing some errors. Errors are handled by decoding the induced Reed-Muller code on H.

Proposition 7.21. *Consider a function $f : \mathbb{F}_p^n \to \mathbb{F}_p$ that is ε-close to a degree-t polynomial $g : \mathbb{F}_p^n \to \mathbb{F}_p$. Then the function f can be self-corrected. That is, assume $K > (k+1)$, then for any given $x \in \mathbb{F}_p^n$, the value of $g(x)$ can be obtained with probability at least $1 - \frac{\varepsilon}{(1-\varepsilon \cdot p^{k+1})^2} \cdot p^{-(K-2k-3)}$ with p^K queries to f.*

Proof. Our goal is to correct the $\mathrm{RM}_p(t,n)$ at the point x. Assume $t = (p-1)\cdot k + R$, where $0 \leqslant R \leqslant (p-2)$. Then the relative distance of the code δ is $(1 - R/p)p^{-k}$. Note that $2p^{-k-1} \leqslant \delta \leqslant p^{-k}$. Recall that the local testability test requires a $(k+1)$-flat, i.e., it tests $\sum_{c_1,\cdots,c_{k+1}\in\mathbb{F}_p} c_1^{p-2-R} f(y_0 + \sum_{i=1}^{k+1} c_i y_i) = 0$, where $y_i \in \mathbb{F}_p^n$.

We choose a slightly larger flat, i.e., a K-flat with $K > (k+1)$ to be chosen later. We consider the code restricted to this K-flat with point x being the origin. We query f on this K-flat. It is known that a majority logic decoding algorithm exists that can decode Reed-Muller codes up to half the minimum distance for any choice of parameters (see [99]). Thus if the number of errors is small we can recover $g(x)$.

Let the relative distance of f from the code be ε and let S be the set of points where it disagrees with the closest codeword. Let the random K-flat be $H = \{x + \sum_{i=1}^{K} t_i u_i | t_i \in \mathbb{F}, u_i \in_R \mathbb{F}_p^n\}$. Let the random variable $Y_{\langle t_1,\cdots,t_K\rangle}$ take the value 1 if $x + \sum_{i=1}^{K} u_i t_i \in S$ and 0 otherwise. Let $D = \mathbb{F}^K \setminus \{0\}$ and $U = \langle u_1, \cdots, u_K\rangle$. Define $Y = \sum_{\langle t_1,\cdots,t_K\rangle \in D} Y_{\langle t_1,\cdots,t_K\rangle}$ and $\ell = (p^K - 1)$. We would like to bound the probability

$$\Pr_U[|Y - \varepsilon\ell| \geqslant (\delta/2 - \varepsilon)\ell].$$

Since $\Pr_U[Y_{t_1,\cdots,t_K} = 1] = \varepsilon$, by linearity we get $\mathbb{E}_U[Y] = \varepsilon\ell$. Let $T = \langle t_1, \cdots, t_K\rangle$. Now

$$\begin{aligned} Var[Y] = & \sum_{T\in\mathbb{F}^K - \{0\}} Var[Y_T] + \sum_{T\neq T'} Cov[Y_T, Y_{T'}] \\ = & \ \ell(\varepsilon - \varepsilon^2) + \sum_{T\neq \lambda T'} Cov[Y_T, Y_{T'}] + \sum_{T=\lambda T'; 1\neq\lambda\in\mathbb{F}^*} Cov[Y_T, Y_{T'}] \\ \leqslant & \ \ell(\varepsilon - \varepsilon^2) + \ell\cdot(p-2)(\varepsilon - \varepsilon^2) \\ = & \ \ell(\varepsilon - \varepsilon^2)(p-1) \end{aligned}$$

The above follows from the fact that when $T \neq \lambda T'$ then the corresponding events Y_T and $Y_{T'}$ are independent and therefore $Cov[Y_T, Y_{T'}] = 0$. Also, when Y_T and $Y_{T'}$ are dependent then $Cov[Y_T, Y_{T'}] = \mathbb{E}_U[Y_T Y_{T'}] - \mathbb{E}_U[Y_T]\mathbb{E}_U[Y_{T'}] \leqslant \varepsilon - \varepsilon^2$.
Therefore, by Chebyshev's inequality we have (assuming $\varepsilon < p^{-(k+1)}$)

$$\Pr_U[|Y - \varepsilon\ell| \geqslant (\delta/2 - \varepsilon)\ell] \leqslant \frac{\ell\varepsilon(1-\varepsilon)(p-1)}{(\delta/2-\varepsilon)^2\ell^2}$$

Now note $(\delta/2 - \varepsilon) \geqslant (p^{-k-1} - \varepsilon) = (1 - \varepsilon \cdot p^{k+1})p^{-k-1}$. We thus have

$$\begin{aligned}\Pr_U[|Y - \varepsilon\ell| \geqslant (\delta/2 - \varepsilon)\ell] &\leqslant \frac{\varepsilon(1-\varepsilon)(p-1)}{(1-\varepsilon\cdot p^{k+1})^2 p^{-2k-2}\ell} \\ &\leqslant \frac{\varepsilon p}{(1-\varepsilon\cdot p^{k+1})^2 p^{-2k-2}(\ell+1)} \\ &= \frac{\varepsilon}{(1-\varepsilon\cdot p^{k+1})^2}\cdot p^{-(K-2k-3)}.\end{aligned}$$

□

7.6 Bibliographics Notes

The results presented in this chapter appear in [72].

As was mentioned earlier, the study of low degree testing (along with self-correction) dates back to the work of Blum, Luby and Rubinfeld ([21]), where an algorithm was required to test whether a given function is linear. The approach in [21] later naturally extended to yield testers for low degree polynomials over fields larger than the total degree. Roughly, the idea is to project the given function on to a random line and then test if the projected univariate polynomial has low degree. Specifically, for a purported degree t function $f : \mathbb{F}_q^n \to \mathbb{F}_q$, the test works as follows. Pick vectors y and b from $\mathbb{F}_q^n$ (uniformly at random), and distinct $s_1, \cdots, s_{t+1}$ from $\mathbb{F}_q$ arbitrarily. Query the oracle representing f at the $t+1$ points $b + s_i y$ and extrapolate to a degree t polynomial $P_{b,y}$ in one variable s. Now test for a random $s \in \mathbb{F}_p$ if

$$P_{b,y}(s) = f(b + sy)$$

(for details see [93],[42]). Similar ideas are also employed to test whether a given function is a low degree polynomial in each of its variable (see [36, 8, 6]).

Alon et al. give a tester over field $\mathbb{F}_2$ for any degree up to the number of inputs to the function (i.e., for any non-trivial degree) [1]. In other words, their work shows that Reed-Muller codes are locally testable. Under the coding theory interpretation, their tester picks a random minimum-weight codeword from the dual code and checks if it is orthogonal to the input vector. It is important to note that these minimum-weight code words generate the Reed-Muller code.

Specifically their test works as follows: given a function $f : \{0,1\}^n \to \{0,1\}$, to test if the given function f has degree at most t, pick $(t+1)$-vectors $y_1, \cdots, y_{t+1} \in \{0,1\}^n$ and test if

$$\sum_{\emptyset \neq S \subseteq [t+1]} f(\sum_{i \in S} y_i) = 0.$$

Independent of [72], Kaufman and Ron, generalizing a characterization result of [42], gave a tester for low degree polynomials over general finite fields (see [74]). They show that a given polynomial is of degree at most t *if and only if* the restriction of the polynomial to every affine subspace of suitable dimension is of degree at most t. Following this

idea, their tester chooses a random affine subspace of a suitable dimension, computes the polynomial restricted to this subspace, and verifies that the coefficients of the higher degree terms are zero[4]. To obtain constant soundness, the test is repeated many times. An advantage of the approach presented in this chapter is that in one round of the test (over the prime field) we test only one linear constraint, whereas their approach needs to test multiple linear constraints.

A basis of RM consisting of minimum-weight codewords was considered in [28, 29]. We extend their result to obtain a different exact characterization for low-degree polynomials. Furthermore, it seems likely that their exact characterization can be turned into a robust characterization following analysis similar to our robust characterization. However, our basis is cleaner and yields a simpler analysis. We point out that for degree smaller than the field size, the exact characterization obtained from [28, 29] coincides with [21, 93, 42]. This provides an alternate proof to the exact characterization of [42] (for more details, see Remark 7.3 and [42]).

In an attempt to generalize our result to more general fields, we obtain an exact characterization of low degree polynomials over general finite fields [71] (see [86] for more details). This provides an alternate proof to the result of Kaufman and Ron [74] described earlier. Specifically the result says that a given polynomial is of degree at most t *if and only if* the restriction of the polynomial to every affine subspace of dimension $\lceil \frac{t+1}{q-q/p} \rceil$ (and higher) is of degree at most t.

Recently Kaufman and Litsyn ([73]) show that the dual of BCH codes are locally testable. They also give a sufficient condition for a code to be locally testable. The condition roughly says that if the number of fixed length codewords in the dual of the union of the code and its ε-far coset is suitably smaller than the same in the dual of the code, then the code is locally testable. Their argument is more combinatorial in nature and needs the knowledge of weight-distribution of the code and thus differs from the self-correction approach used in this work.

[4]Since the coefficients can be written as linear sums of the evaluations of the polynomial, this is equivalent to check several linear constraints

Chapter 8

TOLERANT LOCALLY TESTABLE CODES

In this chapter, we revisit the notion of local testers (as defined in Section 2.3) that was the focus of Chapter 7.

8.1 Introduction

In the definition of LTCs, there is no requirement on the tester for input strings that are very close to a codeword (it has to reject "far" away received words). This "asymmetry" in the way the tester accepts and rejects an input reflects the way Probabilistically Checkable Proofs (or PCPs) [6, 5] are defined, where we only care about accepting perfectly correct proofs with high probability. However, the crux of error-correcting codes is to tolerate and *correct* a few errors that could occur during transmission of the codeword (and not just be able to detect errors). In this context, the fact that a tester can reject received words with few errors is not satisfactory. A more desirable (and stronger) requirement in this scenario would be the following– we would like the tester to make a quick decision on whether or not the purported codeword is close to any codeword. If the tester declares that there is probably a close-by codeword, we then use a decoding algorithm to decode the received word. If on the other hand, the tester rejects, then we assume with high confidence that the received word is far away from all codewords and not run our expensive decoding algorithm.

In this chapter, we introduce the concept of *tolerant testers*. These are testers which reject (w.h.p) received words far from every codeword (like the "standard" local testers) and accept (w.h.p) close-by received words (unlike the "standard" ones which only need to accept codewords). We will refer to codes that admit a tolerant tester as tolerant LTCs. In particular we get tolerant testers that (i) make $O(1)$ queries and work with codes of near constant rate codes and (ii) make sub-linear number of queries and work with codes of constant rate.

8.2 Preliminaries

Recall that for any two vectors $u, v \in [q]^n$, $\delta(u, v)$ denotes the (relative) Hamming distance between them. We will abuse the notation a bit and for any $S \subseteq [q]^n$, use $\delta(u, S)$ to denote $\min_{v \in S} \delta(u, v)$. We now formally define a *tolerant* tester.

Definition 8.1. *For any linear code $\mathcal{C}$ over $\mathbb{F}_q$ of block length n and distance d, and $0 \leqslant c_1 \leqslant c_2 \leqslant 1$, a (c_1, c_2)-tolerant tester T for $\mathcal{C}$ with query complexity $p(n)$ (or simply p when*

the argument is clear from the context) is a probabilistic polynomial time oracle Turing machine such that for every vector $v \in \mathbb{F}_q^n$:

1. *If $\delta(v, \mathcal{C}) \leqslant \frac{c_1 d}{n}$, T upon oracle access to v accepts with probability at least $\frac{2}{3}$ (*tolerance*),*
2. *If $\delta(v, \mathcal{C}) > \frac{c_2 d}{n}$, T rejects with probability at least $\frac{2}{3}$ (*soundness*),*
3. *T makes $p(n)$ probes into the string (oracle) v.*

A code is said to be (c_1, c_2, p)-testable if it admits a (c_1, c_2)-tolerant tester of query complexity $p(\cdot)$.

A tester has *perfect completeness* if it accepts any codeword with probability 1. As pointed out earlier, local testers are just $(0, c_2)$-tolerant testers with perfect completeness. We will refer to these as *standard* testers henceforth. Note that our definition of tolerant testers is per se not a generalization of standard testers since we do not require perfect completeness for the case when the input v is a codeword. However, all our constructions will inherit this property from the standard testers we obtain them from.

Recall one of the applications of tolerant testers mentioned earlier: a tolerant tester is used to decide if the expensive decoding algorithm should be used. In this scenario, one would like to set the parameters c_1 and c_2 such that the tester is tolerant up to the decoding radius. For example, if we have an unique decoding algorithm which can correct up to $\frac{d}{2}$ errors, a particularly appealing setting of parameters would be $c_1 = \frac{1}{2}$ and c_2 as close to $\frac{1}{2}$ as possible. However, we would not be able to achieve such large c_1. In general we will aim for positive constants c_1 and c_2 with $\frac{c_2}{c_1}$ being as small as possible while minimizing $p(n)$.

One might hope that the existing standard testers could also be tolerant testers. We give a simple example to illustrate the fact that this is not the case in general. Consider the tester for the Reed-Solomon (RS) codes of dimension $k+1$: pick $k+2$ points uniformly at random and check if the degree k univariate polynomial obtained by interpolating on the first $k+1$ points agrees with the input on the last point. It is well known that this is a standard tester [96]. However, this is not a tolerant tester. Assume we have an input which differs from a degree k polynomial in only one point. Thus, for $\binom{n-1}{k+1}$ choices of $k+2$ points, the tester would reject, that is, the rejection probability is $\frac{\binom{n-1}{k+1}}{\binom{n}{k+2}} = \frac{k+2}{n}$ which is greater than $\frac{1}{3}$ for high rate RS codes.

Another pointer towards the inherent difficulty in coming up with a tolerant tester is the work of Fischer and Fortnow [39] which shows that there are certain boolean properties which have a standard tester with constant number of queries but for which every tolerant tester requires at least $n^{\Omega(1)}$ queries.

In this chapter, we examine existing standard testers and convert some standard testers into tolerant ones. In Section 8.3 we record a few general facts which will be useful in

performing this conversion. The ultimate goal, if this can be realized at all, would be to construct tolerant LTCs of constant rate which can be tested using $O(1)$ queries (we remark that such a construction has not been obtained even without the requirement of tolerance). In this work, we show that we can achieve either constant number of queries with slightly sub-constant rate (Section 8.4) as well as constant rate with sub-linear number of queries (Section 8.5.1). That is, something non-trivial is possible in both the domains: (a) constant rate, and (b) constant number of queries. Specifically, in Section 8.4 we discuss binary codes which encode k bits into codewords of length $n = k \cdot \exp(\log^{\varepsilon} k)$ for any $\varepsilon > 0$, and can be tolerant tested using $O(1/\varepsilon)$ queries. In Section 8.5.1, following [14], we will study the simple construction of LTCs using products of codes — this yields asymptotically good codes which are tolerant testable using a sub-linear number n^{γ} of queries for any desired $\gamma > 0$. An interesting common feature of the codes in Section 8.4 and 8.5.1 is that they can be constructed from any code that has good distance properties and which in particular need not admit a local tester with sub-linear query complexity. In Section 8.6 we discuss the tolerant testability of Reed-Muller codes, which were considered in Chapter 7.

The overall message from this chapter is that a lot of the work on locally testable code constructions extends fairly easily to also yield tolerant locally testable codes. However, there does not seem to be a generic way to "compile" a standard tester to a tolerant tester for an arbitrary code.

8.3 General Observations

In this section we will spell out some general properties of tolerant testers and subsequently use them to design tolerant testers for some existing codes. All the testers we refer to are *non-adaptive testers* which decide on the locations to query all at once based only on the random choices. The motivation for the definition below will be clear in Section 8.4.

Definition 8.2. *Let $0 < \alpha \leqslant 1$. A tester T is* $(\langle s_1, q_1\rangle, \langle s_2, q_2\rangle, \alpha)$-smooth *if there exists a set $A \subseteq [n]$ where $|A| = \alpha n$ with the following properties:*

- *T queries at most q_1 points in A, and for every $x \in A$, the probability that each of these queries equals location x is at most $\frac{s_1}{|A|}$, and*
- *T queries at most q_2 points in $[n] \setminus A$, and for every $x \in [n] \setminus A$, the probability that each of these queries equals location x is at most $\frac{s_2}{n-|A|}$.*

As a special case a $(\langle 1, q\rangle, \langle 0, 0\rangle, 1)$-smooth tester makes a total of q queries each of them distributed uniformly among the n possible probe points. The following lemma follows easily by an application of the union bound.

Lemma 8.1. *For any $0 < \alpha < 1$, a $(\langle s_1, q_1\rangle, \langle s_2, q_2\rangle, \alpha)$-smooth $(0, c_2)$-tolerant tester T with perfect completeness is a (c_1, c_2)-tolerant tester T', where $c_1 = \frac{n\alpha(1-\alpha)}{3d \max\{q_1 s_1(1-\alpha),\, q_2 s_2 \alpha\}}$.*

Proof. The soundness follows from the assumption on T. Assume $\delta(v, \mathcal{C}) \leqslant \frac{c_1 d}{n}$ and let $f \in \mathcal{C}$ be the closest codeword to v. Suppose that f differs from v in a set A' of yd places among locations in A, and a set B' of $(\beta - y)d$ places among locations in $[n] \setminus A$, where we have $\beta \leqslant c_1$ and $0 \leqslant y \leqslant \beta$. The probability that any of the at most q_1 (resp. q_2) queries of T into A (resp. $[n] \setminus A$) falls in A' (resp. B') is at most $\frac{s_1 yd}{\alpha n}$ (resp. $\frac{s_2(\beta - y)d}{(1-\alpha)n}$). Clearly, whenever T does not query a location in $A' \cup B'$, it accepts (since T has perfect completeness). Thus, an easy calculation shows that the probability that T rejects v is at most

$$\frac{c_1 d}{n} \max\{\frac{s_1 q_1}{\alpha}, \frac{s_2 q_2}{1-\alpha}\}$$

which is $1/3$ for the choice of c_1 stated in the lemma. □

The above lemma is not useful for us unless the relative distance and the number of queries are constants. Next we sketch how to design tolerant testers from existing *robust* testers with certain properties. We first recall the definition of robust testers from [14].

A standard tester T has two inputs: an oracle for the received word v and a random string s. Depending on s, T generates q query positions $i_1, \cdots, i_q$, fixes a circuit C_s and then accepts if $C_s(v_f(s)) = 1$ where $v_f(s) = \langle v_{i_1}, \cdots, v_{i_q} \rangle$. The robustness of T on inputs v and s, denoted by $\rho^T(v, s)$, is defined to be the minimum, over all strings y such that $C_s(y) = 1$, of $\delta(v_f(s), y)$. The expected robustness of T on v is the expected value of $\rho^T(v, s)$ over the random choices of s and would be denoted by $\rho^T(v)$.

A standard tester T is said to be c-robust for $\mathcal{C}$ if for every $v \in \mathcal{C}$, the tester accepts with probability 1, and for every $v \in \mathbb{F}_q^n$, $\delta(v, \mathcal{C}) \leqslant c \cdot \rho^T(v)$.

The tolerant version T' of the standard c-robust tester T is obtained by accepting an oracle v on random input s, if $\rho^T(v, s) \leqslant \tau$ for some threshold τ. (Throughout the chapter τ will denote the threshold.) We will sometimes refer to such a tester as one with threshold τ. Recall that a standard tester T accepts if $\rho^T(v, s) = 0$. We next show that T' is sound.

The following lemma follows from the fact that T is c-robust:

Lemma 8.2. *Let $0 \leqslant \tau \leqslant 1$, and let $c_2 = \frac{(\tau+2)cn}{3d}$. For any $v \in \mathbb{F}_q^n$, if $\delta(v, \mathcal{C}) > \frac{c_2 d}{n}$, then the tolerant tester T' with threshold τ rejects v with probability at least $\frac{2}{3}$.*

Proof. Let $v \in \mathbb{F}_q^n$ be such that $\delta(v, \mathcal{C}) > \frac{c_2 d}{n}$. By the definition of robustness, the expected robustness, $\rho^T(v)$ is at least $\frac{c_2 d}{nc}$, and thus at least $(\tau + 2)/3$ by the choice of c_2. By the standard averaging argument, we can have $\rho^T(v, s) \leqslant \tau$ on at most a fraction $1/3$ of the of the random choices of s for T (and hence T'). Therefore, $\rho^T(v, s) > \tau$ with probability at least $2/3$ over the choice of s and thus T' rejects v with probability at least $2/3$. □

We next mention a property of the query pattern of T which would make T' tolerant. Let S be the set of all possible choices for the random string s. Further for each s, let $p^T(s)$ be the set of positions queried by T.

Definition 8.3. *A tester T has a* partitioned *query pattern if there exists a partition $s_1 \cup \cdots \cup S_m$ of the random choices of T for some m, such that for every i,*

- $\cup_{s \in S_i} p^T(s) = \{1, 2, \cdots, n\}$, *and*
- *For all* $s, s' \in S_i$, $p^T(s) \cap p^T(s') = \emptyset$ *if* $s \neq s'$.

Lemma 8.3. *Let T have a partitioned query pattern. For any $v \in \mathbb{F}_q^n$, if $\delta(v, \mathcal{C}) \leqslant \frac{c_1 d}{n}$, where $c_1 = \frac{n\tau}{3d}$, then the tolerant test T' with threshold τ rejects with probability at most $\frac{1}{3}$.*

Proof. Let $S_1, \cdots, S_m$ be the partition of S, the set of all random choices of the tester T. For each j, by the properties of S_j, $\sum_{s \in S_j} \rho^T(v, s) \leqslant \delta(v, \mathcal{C})$. By an averaging argument and by the assumption on $\delta(v, \mathcal{C})$ and the value of c_1, at least $\frac{2}{3}$ fraction of the choices of s in S_j have $\rho^T(v, s) \leqslant \tau$ and thus, T' accepts. Recalling that $S_1, \cdots, S_m$ was a partition of S, for at least $\frac{2}{3}$ of the choices of s in S, T' accepts. This completes the proof. □

8.4 Tolerant Testers for Binary Codes

One of the natural goals in the study of tolerant codes is to design explicit tolerant binary codes with constant relative distance and as large a rate as possible. In the case of standard testers, Ben-Sasson et al [11] give binary locally testable codes which map k bits to $k \cdot \exp(\log^{\varepsilon} k)$ bits for any $\varepsilon > 0$ and which are testable with $O(1/\varepsilon)$ queries. Their construction uses objects called PCPs of Proximity (PCPP) which they also introduce in [11]. In this section, we show that a simple modification to their construction yields tolerant testable binary codes which map k bits to $k \cdot \exp(\log^{\varepsilon} k)$ bits for any $\varepsilon > 0$. We note that a similar modification is used by Ben-Sasson et al to give a relaxed locally decodable codes [11] but with worse parameters (specifically they gives codes with block length $k^{1+\varepsilon}$).

8.4.1 PCP of Proximity

We start with the definition[1] of of a Probabilistic Checkable proof of Proximity (PCPP). A pair language is simply a language whose elements are naturally a pair of strings, i.e., it is some collection of strings (x, y). A notable example is $\text{CIRCUITVAL} = \{\langle C, a \rangle \mid$ Boolean circuit C evaluates to 1 on assignment $a\}$.

Definition 8.4. *Fix $0 \leqslant \gamma \leqslant 1$. A probabilistic verifier V is a PCPP for a pair language L with proximity parameter γ and query complexity $q(\cdot)$ if the following conditions hold:*

- *(Completeness) If $(x, y) \in L$ then there exists a proof π such that V accepts by accessing the oracle $y \circ \pi$ with probability* 1.
- *(Soundness) If y is γ-far from $L(x) = \{y | (x, y) \in L\}$, then for all proofs π, V accepts by accessing the oracle $y \circ \pi$ with probability strictly less than $\frac{1}{4}$.*

[1] The definition here is a special case of the general PCPP defined in [11] which would be sufficient for our purposes.

- *(Query complexity) For any input x and proof π, V makes at most $q(|x|)$ queries in $y \circ \pi$.*

Note that a PCPP differs from a standard PCP in that it has a more relaxed soundness condition but its queries into part of the input y are also counted in its query complexity.

Ben-Sasson et. al. give constructions of PCPPs with the following guarantees:

Lemma 8.4 ([11]). *Let $\varepsilon > 0$ be arbitrary. There exists a PCP of proximity for the pair language* $\text{CIRCUITVAL} = \{(C, x) | C$ *is a boolean circuit and* $C(x) = 1\}$ *whose proof length, for inputs circuits of size s, is at most $s \cdot \exp(\log^{\varepsilon/2} s)$ and for $t = \frac{2 \log\log s}{\log\log\log s}$ the verifier of proximity has query complexity $O(\max\{\frac{1}{\gamma}, \frac{1}{\varepsilon}\})$ for any proximity parameter γ that satisfies $\gamma \geqslant \frac{1}{t}$. Furthermore, the queries of the verifier are non-adaptive and each of the queries which lie in the input part x are uniformly distributed among the locations of x.*

The fact that the queries to the input part are uniformly distributed follows by an examination of the verifier construction in [11]. In fact, in the extended version of that paper, the authors make this fact explicit and use it in their construction of relaxed locally decodable codes (LDCs). To achieve a tolerant LTC using the PCPP, we will need all queries of the verifier to be somewhat uniformly or smoothly distributed. We will now proceed to make the queries of the PCPP verifier that fall into the "proof part" π near-uniform. This will follow a fairly general method suggested in [11] to smoothen out the query distribution, which the authors used to obtain relaxed locally decodable codes from the PCPP. We will obtain tolerant LTCs instead, and in fact will manage to do so without a substantial increase in the encoding length (i.e., the encoding length will remain $k \cdot 2^{\log^{\varepsilon} k}$). On the other hand, the best encoding length achieved for relaxed LDCs in [11] is $k^{1+\varepsilon}$ for constant $\varepsilon > 0$. We begin with the definition of a mapping that helps smoothen out the query distribution.

Definition 8.5. *Given any $v \in \mathbb{F}_q^n$ and $\vec{p} = \langle p_i \rangle_{i=1}^n$ with $p_i \geqslant 0$ for all $i \in [n]$ and $\sum_{i=1}^n p_i = 1$, we define the mapping* $\mathsf{Repeat}(\cdot, \cdot)$ *as follows:* $\mathsf{Repeat}(v, \vec{p}) \in \mathbb{F}_q^{n'}$ *such that v_i is repeated $\lfloor 4np_i \rfloor$ times in* $\mathsf{Repeat}(v, \vec{p})$ *and $n' = \sum_{i=1}^n \lfloor 4np_i \rfloor$.*

We now show why the mapping is useful. A similar fact appears in [11], but for the sake of completeness we present its proof here.

Lemma 8.5. *For any $v \in \mathbb{F}_q^n$ let a non-adaptive verifier T (with oracle access to v) make $q(n)$ queries and let p_i be the probability that each of these queries probes location $i \in [n]$. Let $c_i = \frac{1}{2n} + \frac{p_i}{2}$ and $\vec{c} = \langle c_i \rangle_{i=1}^n$. Consider the map* $\mathsf{Repeat}(v, \vec{c}) : \mathbb{F}_q^n \to \mathbb{F}_q^{n'}$. *Then there exists another tester T' for strings of length n' with the following properties:*

1. *T' makes $2q(n)$ queries on $v' =$* $\mathsf{Repeat}(v, \vec{c})$ *each of which probes location j, for any $j \in [n']$, with probability at most $\frac{2}{n'}$, and*

2. *for every $v \in \mathbb{F}_q^n$, the decision of T' on v' is identical to that of T on v. Further, $3n < n' \leqslant 4n$.*

Proof. We first add q dummy queries to T each of which are uniformly distributed, and then permute the $2q$ queries in a random order. Note that each of the $2q$ queries is now identically distributed. Moreover, any position in v is probed with probability at least $\frac{1}{2n}$ for each of the $2q$ queries. For the rest of the proof we will assume that T makes $2q$ queries for each of which any $i \in [n]$ is probed with probability $c_i = \frac{p_i}{2} + \frac{1}{2n}$. Let $r_i = \lfloor 4nc_i \rfloor$. Note that $r_i \leqslant 4nc_i$ and $r_i > 4nc_i - 1$. Recalling that $n' = \sum_{i=1}^{n} r_i$ and $\sum_{i=1}^{n} c_i = 1$, we have $3n < n' \leqslant 4n$.

T' just simulates T in the following manner: if T queries v_i for any $i \in [n]$, T' queries one of the r_i copies of v_i in v' uniformly at random. It is clear that the decision of T' on $v' = \mathsf{Repeat}(v, \vec{c})$ is identical to that of T on v. We now look at the query distribution of T'. T' queries any $j \in [n']$, where $v'_j = v_i$, with probability $p'_j = c_i \cdot \frac{1}{r_i}$. Recalling the lower bound on r_i, we have $p'_j \leqslant \frac{c_i}{4nc_i - 1}$ which is at most $\frac{1}{2n}$ since clearly $c_i \geqslant \frac{1}{2n}$. We showed earlier that $n' \leqslant 4n$ which implies $p'_j \leqslant \frac{2}{n'}$ as required. □

One might wonder if we can use Lemma 8.5 to smoothen out the queries made by the verifier of an arbitrary LTC to obtain a tolerant LTC. That is, whether the above allows one to compile the verifier for any LTC in a black-box manner to obtain a tolerant verifier. We will now argue (informally) that this technique alone will not work. Let C_1 be an $[n, k, d]_q$ LTC with a standard tester T_1 that makes q identically distributed queries with distribution p_i, $1 \leqslant i \leqslant n$, such that $p_i \geqslant 1/2n$ for each i. Create a new $[n+1, k, d]_q$ code C_2 whose $(n+1)$'th coordinate is just a copy of the n'th coordinate, i.e., corresponding to each codeword $(c_1, c_2, \ldots, c_n) \in \mathbb{F}_q^n$ of C_1, we will have a codeword $(c_1, c_2, \ldots, c_n, c_n) \in \mathbb{F}_q^{n+1}$ of C_2. Consider the following tester T_2 for C_2: Given oracle access to $v \in \mathbb{F}_q^{n+1}$, with probability $1/2$ check whether $v_n = v_{n+1}$, and with probability $1/2$ run the tester T_1 on the first n coordinates of v. Clearly, T_2 is a standard tester for C_2.

Now, consider what happens in the conversion procedure of Lemma 8.5 to get (C', T') from (C_2, T_2). Note that by Lemmas 8.5 and 8.3, T' is tolerant. Let $\vec{q} = (q_1, \ldots, q_{n+1})$ be the query distribution of T_2. Since T_2 queries (v_n, v_{n+1}) with probability $1/2$, the combined number of locations of $v' = \mathsf{Repeat}(v, \vec{q})$ corresponding to v_n, v_{n+1} will be about $1/2$ of the total length n'. Now let v' be obtained from a codeword of C' by corrupting just these locations. The tester T' will accept such a v' with probability at least $1/2$, which contradicts the soundness requirement since v' is $1/2$-far from C'. Therefore, using the behavior of the original tester T_2 as just a black-box, we cannot in general argue that the construction of Lemma 8.5 maintains good soundness.

Applying the transformation of Lemma 8.5 to the proximity verifier and proof of proximity of Lemma 8.4, we conclude the following.

Proposition 8.6. *Let $\varepsilon > 0$ be arbitrary. There exists a PCP of proximity for the pair language* $\mathrm{CIRCUITVAL} = \{(C, x) | C$ *is a boolean circuit and* $C(x) = 1\}$ *with the following properties:*

1. *The proof length, for inputs circuits of size s, is at most $s \cdot \exp(\log^{\varepsilon/2} s)$, and*

2. *for* $t = \frac{2 \log \log s}{\log \log \log s}$ *the verifier of proximity has query complexity* $O(\max\{\frac{1}{\gamma}, \frac{1}{\varepsilon}\})$ *for any proximity parameter* γ *that satisfies* $\gamma \geqslant \frac{1}{t}$.

Furthermore, the queries of the verifier are non-adaptive with the following properties:

1. *Each query made to one of the locations of the input* x *is uniformly distributed among the locations of* x*, and*

2. *each query to one of the locations in the proof of proximity* π *probes each location with probability at most* $2/|\pi|$ *(and thus is distributed* nearly uniformly *among the locations of* π*).*

8.4.2 The Code

We now outline the construction of the locally testable code from [11]. The idea behind the construction is to make use of a PCPP to aid in checking if the received word is a codeword is far away from being one. Details follow.

Suppose we have a binary code $C_0 : \{0,1\}^k \to \{0,1\}^m$ of distance d defined by a parity check matrix $H \in \{0,1\}^{(m-k)\times m}$ that is sparse, i.e., each of whose rows has only an absolute constant number of 1's. Such a code is referred to as a low-density parity check code (LDPC). For the construction below, we will use any such code which is asymptotically good (i.e., has rate k/m and relative distance d/m both positive as $m \to \infty$). Explicit constructions of such codes are known using expander graphs [95]. Let V be a verifier of a PCP of proximity for membership in C_0; more precisely, the proof of proximity of an input string $w \in \{0,1\}^m$ will be a proof that $\tilde{C}_0(w) = 1$ where $\tilde{C}_0$ is a linear-sized circuit which performs the parity checks required by H on w (the circuit will have size $O(m) = O(k)$ since H is sparse and C_0 has positive rate). Denote by $\pi(x)$ be the proof of proximity guaranteed by Proposition 8.6 for the claim that the input $C_0(x)$ is a member of C_0 (i.e., satisfies the circuit $\tilde{C}_0$). By Proposition 8.6 and fact that the size of $\tilde{C}_0$ is $O(k)$, the length of $\pi(x)$ can be made at most $k \exp(\log^{\varepsilon/2} k)$.

The final code is defined as $\mathcal{C}_1(x) = (C_0(x)^t, \pi(x))$ where $t = \frac{(\log k - 1)|\pi(x)|}{|C_0(x)|}$. The repetition of the code part $C_0(x)$ is required in order to ensure good distance, since the length of the proof part $\pi(x)$ typically dominates and we have no guarantee on how far apart $\pi(x_1)$ and $\pi(x_2)$ for $x_1 \neq x_2$ are.

For the rest of this section let ℓ denote the proof length. The tester T_1 for $\mathcal{C}_1$ on an input $w = (w_1, \cdots, w_t, \pi) \in \{0,1\}^{tm+\ell}$ picks $i \in [t]$ at random and runs the PCPP verifier V on $w_i \circ \pi$. It also performs a few rounds of the following consistency checks: pick $i_1, i_2 \in [t]$ and $j_1, j_2 \in [m]$ at random and check if $w_{i_1}(j_1) = w_{i_2}(j_2)$. Ben-Sasson et al in [11] show that T_1 is a standard tester. However, T_1 need not be a tolerant tester. To see this, note that the proof part of $\mathcal{C}_1$ forms a $\frac{1}{\log k}$ fraction of the total length. Now consider a received word $w_{rec} = (w_0, \cdots, w_0, \pi')$ where $w_0 \in C_0$ but π' is not a correct proof for w_0 being a valid

codeword in c_0. Note that w_{rec} is close to $\mathcal{C}_1$. However, T_1 is not guaranteed to accept w_{rec} with high probability.

The problem with the construction above was that the proof part was too small: a natural fix is to make the proof part a constant fraction of the codeword. We will show that this is sufficient to make the code tolerant testable. We also remark that a similar idea was used by Ben-Sasson et. al. to give efficient constructions for relaxed locally decodable codes [11].

Construction 8.1. *Let $0 < \beta < 1$ be a parameter, $C_0 : \{0,1\}^k \to \{0,1\}^m$ be a good [2] binary code and V be a PCP of proximity verifier for membership in C_0. Finally let $\pi(x)$ be the proof corresponding to the claim that $C_0(x)$ is a codeword in C_0. The final code is defined as $\mathcal{C}_2(x) = (C_0(x)^{r_1}, \pi(x)^{r_2})$ with $r_1 = \frac{(1-\beta)\log k|\pi(x)|}{m}$ and $r_2 = \beta \log k$.*[3]

For the rest of the section the proof length $|\pi(x)|$ will be denoted by ℓ. Further the proximity parameter and the number of queries made by the PCPP verifier V would be denoted by γ_p and q_p respectively. Finally let ρ_0 denote the relative distance of the code C_0.

The tester T_2 for $\mathcal{C}_2$ is also the natural generalization of T_1. For a parameter q_r (to be instantiated later) and input $w = (w_1, \cdots, w_{r_1}, \pi_1, \cdots, \pi_{r_2}) \in \{0,1\}^{r_1 m + r_2 l}$, T_2 does the following:

1. Repeat the next two steps twice.
2. Pick $i \in [r_1]$ and $j \in [r_2]$ randomly and run V on $w_i \circ \pi_j$.
3. Do q_r repetitions of the following: pick $i_1, i_2 \in [r_1]$ and $j_1, j_2 \in [m]$ randomly and check if $w_{i_1}(j_1) = w_{i_2}(j_2)$.

The following lemma captures the properties of the code $\mathcal{C}_2$ and its tester T_2.

Lemma 8.7. *The code $\mathcal{C}_2$ in Construction 8.1 and the tester T_2 (with parameters β and q_r respectively) above have the following properties:*

1. *The code $\mathcal{C}_2$ has block length $n = \log k \cdot \ell$ with minimum distance d lower bounded by $(1-\beta)\rho_0 n$.*
2. *T_2 makes a total of $q = 2q_p + 4q_r$ queries.*
3. *T_2 is $(\langle 1, q\rangle, \langle 2, 2q_p\rangle, 1-\beta)$-smooth.*

---

[2] This means that $m = O(k)$ and the encoding can be done by circuits of nearly linear size $s_0 = \tilde{O}(k)$.

[3] The factor $\log k$ overhead is overkill, and a suitably large constant will do, but since the proof length $|\pi(x)|$ will anyway be larger than $|x|$ by more than a polylogarithmic factor in the constructions we use, we can afford this additional $\log k$ factor and this eases the presentation somewhat.

4. *T_2 is a (c_1, c_2)-tolerant tester with $c_1 = \frac{n\beta(1-\beta)}{6d\max\{(2q_r+q_p)\beta,\, 2(1-\beta)q_p\}}$ and $c_2 = \frac{n}{d}(\gamma_p + \frac{4}{q_r} + \beta)$.*

Proof. From the definition of $\mathcal{C}_2$, it has block length $n = r_1 m + r_2 \ell = \frac{(1-\beta)\ell \log k}{m} \cdot m + \beta \log k \cdot \ell = \log k \cdot \ell$. Further as C_0 has relative distance ρ_0, $\mathcal{C}_2$ has relative distance at least $\frac{r_1 \rho_0 m}{\ell \log k} = (1-\beta)\rho_0$.

T_2 makes the same number of queries as V which is q_p in Step 2. In Step 3, T_2 makes $2q_r$ queries. As T_2 repeats Steps 2 and 3 twice, we get the desired query complexity.

To show the smoothness of T_2 we need to define the appropriate subset $A \subset [n]$ such that $|A| = (1-\beta)n$. Let A be the set of indices with the code part: i.e. $A = [r_1 m]$. T_2 makes $2q_r$ queries in A in Step 3 each of which is uniformly distributed. Further by Proposition 8.6, T_2 in step 2 makes at most q_p queries in A which are uniformly distributed and at most q_p queries in $[n] \setminus A$ each of which are within a factor 2 of being queried uniformly at random. To complete the proof of property 3 note that T_2 repeats step 2 and 3 twice.

The tolerance of T_2 follows from property 3 and Lemma 8.1. For the soundness part note that if $w = (w_1, \cdots, w_{r_1}, \pi_1, \cdots, \pi_{r_2}) \in \{0,1\}^{r_1 m + r_2 l}$ is γ-far from $\mathcal{C}_2$ then $w' = (w_1, \cdots, w_{r_1})$ is at least $\frac{\gamma n - r_2 \ell}{n} = \frac{\gamma n - \beta n}{n} = \gamma - \beta$ far from the repetition code $C' = \{C_0(x)^{r_1} | x \in \{0,1\}^k\}$. For $\gamma = c_2 d/n$ with the choice of c_2 in the lemma, we have $\gamma - \beta \geqslant \gamma_p + 4/q_r$. The rest of the proof just follows the proof in [11] (also see [68, Chap. 12]) of the soundness of the tester T_1 for the code $\mathcal{C}_1$– for the sake of completeness we complete the poof here. We will show that one invocation of Steps 2 and 3 results in T_2 accepting w with probability strictly less than $\frac{1}{2}$. The two repetitions of Steps 2 and 3 reduces this error to at most $\frac{1}{4}$.

Let $u \in \{0,1\}^m$ be the string such that u^t is the "repetition sequence" that is closest to w', that is one that minimizes $\Delta(w', u^t) = \sum_{i=1}^{r_1} \Delta(w_i, u)$. We now consider two cases:

- Case 1: $\Delta(w', u^t) \geqslant r_1 m/q_r$. In this case, a single execution of the test in Step 3 rejects with probability

$$\begin{aligned}
\mathbb{E}_{i_1, i_2 \in [r_1]}\left[\Delta(w_{i_1}, w_{i_2})/m\right] &= \frac{1}{m(r_1)^2} \sum_{i_2} \sum_{i_1} \Delta(w_{i_1}, w_{i_2}) \\
&\geqslant \frac{1}{m(r_1)^2} \sum_{i_2} \sum_{i_1} \Delta(w_{i_1}, u) \\
&= \frac{1}{m r_1} \sum_{i_1=1}^{r_1} \Delta(w_{i_1}, u) \\
&= \Delta(w', u^t)/(m r_1) \\
&\geqslant 1/q_r,
\end{aligned}$$

where the first inequality follows from the choice of u and the second inequality

follows from the case hypothesis. Thus, after q_r repetitions the test will accept with probability $(1 - 1/q_r)^{q_r} < 1/e < 1/2$.

- Case 2: $\Delta(w', u^t) < r_1 m/q_r$. In this case we have the following (where for any subset S of vectors and a vector u, we will use $\Delta(u, S) = \min_{v \in S} \Delta(u, v)$):

$$\frac{\Delta(u, C_0)}{r_1} = \frac{\Delta(u^t, C')}{r_1 m} \geqslant \frac{\Delta(w', C') - \Delta(w', u^t)}{r_1 m} \geqslant \gamma_p + 4/q_r - 1/q_r = \gamma_p + 3/q_r, \tag{8.1}$$

where the first inequality follows from the triangle inequality and the last inequality follows from the case hypothesis (and the fact that w' is $\gamma_p + 4/q_r$-far from C'). Now by the case hypothesis, for an average i, $\Delta(w_i, u) \leqslant m/q_r$. Thus, by a Markov argument, at most one thirds of the w_i's are $3/q_r$-far from u. Since u is $\gamma_p + 3/q_r$-far from C_0 (by (8.1)), this implies (along with triangle inequality) that for at least two thirds of the w_i's are γ_p-far from C_0. Thus, by the property of the PCPP, for each such w_i the test in Step 2 should accept with probability at most $1/4$. Thus the total acceptance probability in this case is at most $\frac{1}{3} \cdot 1 + \frac{2}{4} \cdot \frac{1}{4} = \frac{1}{2}$, as desired.

Thus, in both cases the tester accepts w with probability at most $1/2$, as required. □

Fix any $0 < \gamma < 1$ and let $\beta = \frac{\gamma}{2}$, $\gamma_p = \frac{\gamma}{6}$, $q_r = \frac{12}{\gamma}$. With these settings we get $\gamma_p + \frac{4}{q_r} + \beta = \gamma$ and $q_p = O(\frac{1}{\gamma})$ from Proposition 8.6 with the choice $\varepsilon = 2\gamma$. Finally, $q = 2q_p + 4q_r = O(\frac{1}{\gamma})$. Substituting the parameters in c_2 and c_1, we get $c_2 = \frac{\gamma n}{d}$ and

$$\frac{c_1 d}{n} = \frac{\gamma}{24 \max\{\gamma(q_r + q_p/2),\ (2 - \gamma)q_p\}} = \Omega(\gamma^2)\ .$$

Also note that the minimum distance $d \geqslant (1 - \beta)\rho_0 n = (1 - \frac{\gamma}{2})\rho_0 n \geqslant \frac{\rho_0}{2} n$. Thus, we have the following result for tolerant testable binary codes.

Theorem 8.1. *There exists an absolute constant $\alpha_0 > 0$ such that for every γ, $0 < \gamma < 1$, there exists an explicit binary linear code $\mathcal{C} : \{0,1\}^k \to \{0,1\}^n$ where $n = k \cdot \exp(\log^\gamma k)$ with minimum distance $d \geqslant \alpha_0 n$ which admits a (c_1, c_2)-tolerant tester with $c_2 = O(\gamma)$, $c_1 = \Omega(\gamma^2)$ and query complexity $O(\frac{1}{\gamma})$.*

The claim about explicitness follows from the fact that the PCPP of Lemma 8.4 and hence Proposition 8.6 has an explicit construction. The claim about linearity follows from the fact that the PCPP for CIRCUITVAL is a linear function of the input when the circuit computes linear functions — this aspect of the construction is discussed in detail in Chapter 9 in [68].

8.5 Product of Codes

Tensor product of codes (or just *product of codes*) is simple way to construct new codes from existing codes such that the constructed codes have testers with sub-linear query complexity even though the original code need not admit a sub-linear complexity tester [14]. We start with the definition of product of codes.

Definition 8.6 (Tensor Product of Codes). *Given $\mathcal{C}_1$ and $\mathcal{C}_2$ that are $[k_1, n_1, d_1]$ and $[k_2, n_2, d_2]$ codes, their tensor product, denoted by $\mathcal{C}_1 \otimes \mathcal{C}_2$, consists of $n_2 \times n_1$ matrices such that every row of the matrix is a codeword in $\mathcal{C}_1$ and every column is a codeword in $\mathcal{C}_2$.*

It is well known that $\mathcal{C}_1 \otimes \mathcal{C}_2$ is an $[n_1 n_2, k_1 k_2, d_1 d_2]$ code.

A special case in which we will be interested is when $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{C}$. In such a case, given an $[n, k, d]_q$ code $\mathcal{C}$, the product of $\mathcal{C}$ with itself, denoted by $\mathcal{C}^2$, is a $[n^2, k^2, d^2]_q$ code such that a codeword (viewed as a $n \times n$ matrix) restricted to any row or column is a codeword in $\mathcal{C}$. It can be shown that this is equivalent to the following [100]. Given the $k \times n$ generator matrix M of $\mathcal{C}$, $\mathcal{C}^2$ is precisely the set of matrices in the set $\{M^T \cdot X \cdot M \mid X \in F_q^{k \times k}\}$.

8.5.1 Tolerant Testers for Tensor Products of Codes

A very natural test for $\mathcal{C}^2$ is to randomly choose a row or a column and then check if the restriction of the received word on that row or column is a codeword in $\mathcal{C}$ (which can be done for example by querying all the n points in the row or column). Unfortunately, as we will see in Section 8.5.2, this test is not robust in general.

Ben-Sasson and Sudan in [14] considered the more general product of codes $\mathcal{C}^t$ for $t \geqslant 3$ (where $\mathcal{C}^t$ denotes $\mathcal{C}$ tensored with itself $t-1$ times) along with the following general tester: Choose at random $b \in \{1, \cdots, t\}$ and $i \in \{1, \cdots, n\}$ and check if b^{th} coordinate of the received word (which is an element of $\mathbb{F}_q^{n^t}$) when restricted[4] to i is a codeword in $\mathcal{C}^{t-1}$. It is shown in [14] that this test is robust, in that if a received word is far from $\mathcal{C}^t$, then many of the tested substrings will be far from $\mathcal{C}^{t-1}$. This tester lends itself to recursion: the test for $\mathcal{C}^{t-1}$ can be reduced to a test for $\mathcal{C}^{t-2}$ and so on till we need to check whether a word in $\mathbb{F}_q^{n^2}$ is a codeword of $\mathcal{C}^2$. This last check can done by querying all the n^2 points, out of the n^t points in the original received word, thus leading to a sub-linear query complexity. As shown in [14], the reduction can be done in $\log t$ stages by the standard halving technique.

Thus, even though $\mathcal{C}$ might not have a tester with a small query complexity, we can test $\mathcal{C}^t$ with a polylogarithmic number of queries.

We now give a tolerant version of the test for product of codes given by Ben-Sasson and Sudan [14]. In what follows $t \geqslant 4$ will be a power of two. As mentioned above the tester T for the tensor product $\mathcal{C}^t$ reduces the test to checking if some restriction of the given string belong to $\mathcal{C}^2$. For the rest of this section, with a slight abuse of notation let $v_f \in \mathbb{F}_q^{n^2}$ denote

[4]For the $t = 2$ case b signifies either row or column and i denotes the row/column index.

the final restriction being tested. In what follows we assume that by looking at all points in any $v \in \mathbb{F}_q^{n^2}$ one can determine if $\delta(v, \mathcal{C}^2) \leqslant \tau$ in time polynomial in n^2.

The tolerant version of the test of [14] is a simple modification as mentioned in Section 8.3: reduce the test on $\mathcal{C}^t$ to $\mathcal{C}^2$ as in [14] and then accept if v_f is τ-close to $\mathcal{C}^2$.

First we make the following observation about the test in [14]. The test recurses $\log t$ times to reduce the test to $\mathcal{C}^2$. At step l , the test chooses an random coordinate b_l (this will just be a random bit) and fixes the value of the b_l^{th} coordinate of the current $\mathcal{C}^{\frac{t}{2^l}}$ to an index i_l (where i_l takes values in the range $1 \leqslant i_l \leqslant n^{t/2^l}$). The key observation here is that for each fixed choice of $b_1, \cdots, b_{\log t}$, distinct choices of $i_1, \cdots, i_{\log t}$ correspond to querying disjoint sets n^2 points in the original $v \in \mathbb{F}_q^{n^t}$ string, which together form a partition of all coordinates of v. In other words, T has a *partitioned* query pattern, which will be useful to argue tolerance. For soundness, we use the results in [14], which show that their tester is $C^{\log t}$-robust for $C = 2^{32}$.

Applying Lemmas 8.2 and 8.3, therefore, we have the following result:

Theorem 8.2. *Let $t \geqslant 4$ be a power of two and $0 < \tau \leqslant 1$. There exist $0 < c_1 < c_2 \leqslant 1$ with $\frac{c_2}{c_1} = C^{\log t}(1 + 2/\tau)$ such that the proposed tolerant tester for $\mathcal{C}^t$ is a (c_1, c_2)-tolerant tester with query complexity $N^{2/t}$ where N is the block length of $\mathcal{C}^t$. Further, c_1 and c_2 are constants (independent of N) if t is a constant and $\mathcal{C}$ has constant relative distance.*

Corollary 8.3. *For every $\gamma > 0$, there is an explicit family of asymptotically good binary linear codes which are tolerant testable using n^γ queries, where n is the block length of the concerned code. (The rate, relative distance and thresholds c_1, c_2 for the tolerant testing depend on γ.)*

8.5.2 Robust Testability of Product of Codes

Recall that a standard tester for a code is robust if for every received word which is far from being a codeword, the tester not only rejects the codeword with high probability but also with high probability the tester's local view of the received word is far from any accepting view (see Section 8.3 for a more formal definition).

As was mentioned before for the product code $\mathcal{C}_1 \otimes \mathcal{C}_2$, there is a natural tester (which we call $T_{\mathcal{C}_1 \otimes \mathcal{C}_2}$)– flip a coin; if it is heads check if a random row is a codeword in $\mathcal{C}_1$; if it is tails, check if a random column is a codeword in $\mathcal{C}_2$. This test is indeed robust in a couple of special cases– for example, when both $\mathcal{C}_1$ and $\mathcal{C}_2$ are Reed-Solomon codes (see Section 8.6.1 for more details) and when both $\mathcal{C}_1$ and $\mathcal{C}_2$ are themselves tensor product of a code [14].

P. Valiant showed that there are linear codes $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{C}_1 \otimes \mathcal{C}_2$ is not robustly testable [103]. Valiant constructs linear codes $\mathcal{C}_1$, $\mathcal{C}_2$ and a matrix v such that every row (and column) of v is "close" to some codeword in $\mathcal{C}_1$ (and $\mathcal{C}_2$) while v is "far" from every codeword in $\mathcal{C}_1 \otimes \mathcal{C}_2$ (where close and far are in the sense of hamming distance).

However, Valiant's construction *does not* work when $\mathcal{C}_1$ and $\mathcal{C}_2$ are the same code. In this section, we show a reduction from Valiant's construction to exhibit a code $\mathcal{C}$ such that $\mathcal{C}^2$ is not robustly testable.

Preliminaries and Known Results

$\mathcal{C}$ is said to be robustly testable if it has a $\Omega(1)$-robust tester. For a given code $\mathcal{C}$ of block length n over $\mathbb{F}_q$ and a vector $v \in \mathbb{F}_q$, the (relative) Hamming distance of v to the closest codeword in $\mathcal{C}$ is denoted by $\delta_{\mathcal{C}}(v)$.

Asking whether $T_{\mathcal{C}_1 \otimes \mathcal{C}_2}$ is a robust tester has the following nice interpretation. The q queries $i_1, \cdots, i_q$ are either rows or columns of the received word v. Let the row or column corresponding to the random seed s be denoted by v^s. Then the robustness of $T_{\mathcal{C}_1 \otimes \mathcal{C}_2}$ on inputs (v, s), $\rho^{T_{\mathcal{C}_1 \otimes \mathcal{C}_2}}(v, s)$ is just $\delta_{\mathcal{C}_1}(v^s)$ when i_s corresponds to a row and $\delta_{\mathcal{C}_2}(v^s)$ when i_s corresponds to a column. Therefore the expected robustness of $T_{\mathcal{C}_1 \otimes \mathcal{C}_2}$ on v is the average of the following two quantities: the average relative distance of the rows of v from $\mathcal{C}_1$ and the average relative distance of the columns of v from $\mathcal{C}_2$.

In particular, if $T_{\mathcal{C}_1 \otimes \mathcal{C}_2}$ is $\Omega(1)$-robust then it implies that for every received word v such that all rows (and columns) of v are $o(1)$-close to $\mathcal{C}_1$ (and $\mathcal{C}_2$), v is $o(1)$-close to $\mathcal{C}_1 \otimes \mathcal{C}_2$. P. Valiant proved the following result.

Theorem 8.4 ([103]). *There exist linear codes $[n_1, k_1, d_1 = n_1/10]$ and $[n_2 = n_1^2, k_2, d_2 = n_2/10]$ (call them $\mathcal{C}_1$ and $\mathcal{C}_2$) and a $n_2 \times n_1$ received word v such that every row of v is $1/n_1$-close to $\mathcal{C}_1$ and every column of v is a codeword $\mathcal{C}_2$ but v is $1/20$-far from $\mathcal{C}_1 \times \mathcal{C}_2$.*

Note that in the above construction, $n_2 \neq n_1$ and in particular $\mathcal{C}_1$ and $\mathcal{C}_2$ are not the same code.

Reduction from the Construction of Valiant

In this section, we prove the following result.

Theorem 8.5. *Let $\mathcal{C}_1 \neq \mathcal{C}_2$ be $[n_1, k_1, d_1 = \Omega(n_1)]$ and $[n_2, k_2, d_2 = \Omega(n_2)]$ codes respectively (with $n_2 > n_1$) and let v be a $n_2 \times n_1$ matrix such that every row (and column) of v is $g(n_1)$-close to $\mathcal{C}_1$ ($g(n_2)$-close to $\mathcal{C}_2$) but v is ρ-far from $\mathcal{C}_1 \otimes \mathcal{C}_2$. Then there exists a linear code $\mathcal{C}$ with parameters $n, k, d = \Omega(n)$ and a received word v' such that such that every row (and column) of v' is $g(n_1)/2$-close (and $g(n_2)/2$-close) to $\mathcal{C}$ but v' is $\rho/4$-far from $\mathcal{C}^2$.*

Proof. We will first assume that n_1 divides n_2 and let $m = \frac{n_2}{n_1}$. For any $x \in \Sigma^{k_1}$ and $y \in \Sigma^{k_2}$, let

$$\mathcal{C}(\langle x, y \rangle) = \langle (\mathcal{C}_1(x))^m, \mathcal{C}_2(y) \rangle$$

Thus, $k = k_1 + k_2$ and $n = mn_1 + n_2$. Also as $d_1 = \Omega(n_1)$ and $d_2 = \Omega(n_2)$, $d = \Omega(n)$.

We now construct the $n \times n$ matrix v' from v. The lower left $n_2 \times mn_1$ sub-matrix of v' contains the matrix v^m where v^m is the horizontal concatenation of m copies of v (which is a $n_2 \times n_1$ matrix). Every other entry in v' is 0. See figure 8.1 for an example with $m = 2$.

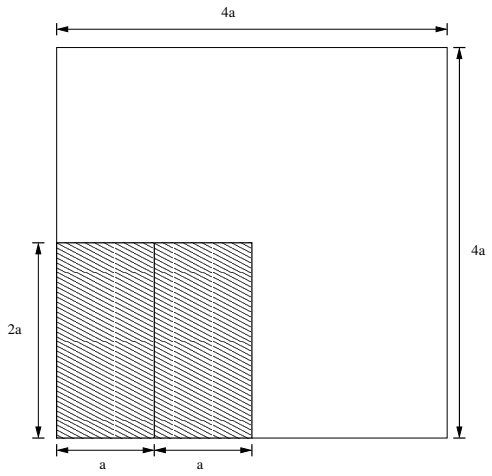


Figure 8.1: The construction of the new received word v' from v for the case when $n_1 = a$, $n_2 = 2a$ and $m = 2$. The shaded boxes represent v and the unshaded regions has all 0s.

Let w be the codeword in $\mathcal{C}_1 \otimes \mathcal{C}_2$ closest to v and construct w' in the same manner as v' was constructed from v. We first claim that w' is the codeword in[5] $\mathcal{C}^2$ closest to v'. For the sake of contradiction, assume that there is some other codeword w'' in $\mathcal{C}^2$ such that $\Delta(v', w'') < \Delta(v', w')$. For any $2n' \times 2n'$ matrix u let u_{lb} denote the lower left $n' \times n'$ sub-matrix of u. Note that by definition of $\mathcal{C}$, $w''_{lb} = x^m$ where $x \in \mathcal{C}_1 \otimes \mathcal{C}_2$. Further, as v' (necessarily) has 0 everywhere other than v'_{lb} and $\Delta(v', w'') < \Delta(v', w')$, it holds that $\Delta(v, w) > \Delta(v, x)$ which contradicts the definition of w.

Finally, it is easy to see that

$$\delta_{\mathcal{C}^2}(v') = \Delta(v', w')/n^2 = \Delta(v, w)m/(mn_1 + n_2)^2 = \Delta(v, w)/(4n_1n_2) = \frac{\rho}{4}$$

and if for any row (or column), the relative distance of v restricted to that row (or column) from $\mathcal{C}_1$ ($\mathcal{C}_2$) is at most α then for every row (or column), the relative distance of v' restricted to that row (or column) from $\mathcal{C}$ is at most $\alpha/2$.

This completes the proof for the case when n_1 divides n_2. For the case when n_1 does not divide n_2 a similar construction works if one defines $\mathcal{C}$ in the following manner (for any $x \in \Sigma^{k_1}$ and $x_2 \in \Sigma^{k_2}$)

$$\mathcal{C}(\langle x, y\rangle) = \langle (\mathcal{C}_1(x))^{\ell/n_1}, (\mathcal{C}_2(y))^{\ell/n_2}\rangle$$

where $\ell = \mathrm{lcm}(n_1, n_2)$. The received word v' in this case would have its lower left $\ell \times \ell$ matrix as $v^{(\ell/n_1, \ell/n_2)}$ (where $v^{(m_1,m_2)}$ is the matrix obtained by vertically concatenating m_2 copies of v^{m_1}) and it has 0s everywhere else. □

[5]Note that $w' \in \mathcal{C}^2$ as the all zeros vector is a codeword in both $\mathcal{C}_1$ and $\mathcal{C}_2$ and $w \in \mathcal{C}_1 \otimes \mathcal{C}_2$.

Theorem 8.4 and 8.5 imply the following result.

Corollary 8.6. *There exist a linear code $\mathcal{C}$ with linear distance such that the tester $T_{\mathcal{C}^2}$ is not $\Omega(1)$-robust for $\mathcal{C}^2$.*

8.6 Tolerant Testing of Reed-Muller Codes

In this section, we discuss testers for codes based on multivariate polynomials.

8.6.1 Bivariate Polynomial Codes

As we saw in Section 8.5.2, one cannot have a robust standard testers for $\mathcal{C}^2$ in general. In this subsection, we consider a special case when $\mathcal{C} = \mathrm{RS}[n, k+1, d = n-k]_q$, that is, the Reed–Solomon code based on evaluation of degree k polynomials over $\mathbb{F}_q$ at n distinct points in the field. We show that the tester for $\mathcal{C}^2$ considered in Section 8.5.2 is tolerant for this special case. It is well-known (see, for example, Proposition 2 in [88]) that in this case $\mathcal{C}^2$ is the code with codewords being the evaluations of bivariate polynomials over $\mathbb{F}_q$ of degree k in each variable. The problem of low-degree testing for bivariate polynomials is a well-studied one: in particular we use the work of Polishchuk and Spielman [88] who analyze a tester using axis parallel lines. Call a bivariate polynomial to be one of degree (k_1, k_2) if the maximum degrees of the two variables are k_1 and k_2 respectively. In what follows, we denote by $Q' \in \mathbb{F}_q^{n \times n}$ the received word to be tested (thought of as an $n \times n$ matrix), and let $Q(x, y)$ be the degree (k, k) polynomial whose encoding is closest to Q'.

We now specify the tolerant tester T'. The upper bound of $1 - \sqrt{1 - d/n}$ on τ comes from the fact that this is largest radius for which decoding an $\mathrm{RS}[n, k+1, d]$ code is known to be solvable in polynomial time [63].

1. Fix τ where $0 \leqslant \tau \leqslant 1 - \sqrt{1 - d/n}$.

2. With probability $\frac{1}{2}$ choose $b = 0$ or $b = 1$.

   - If $b = 0$, choose a row r randomly and reject if $\delta(Q'(r, \cdot), P(\cdot)) > \tau$ for every univariate polynomial P of degree k and accept otherwise.
   - If $b = 1$, choose a column c randomly and reject if $\delta(Q'(\cdot, c), P(\cdot)) > \tau$ for every univariate polynomial P of degree k and accept otherwise.

The following theorem shows that T' is a tolerant tester.

Theorem 8.7. *There exists an absolute constant $c_0 > 0$ such that for $\tau \leqslant 1 - \sqrt{1 - d/n}$, the tester T' with threshold τ is a $(c_1, c_2, \sqrt{N})$-tolerant tester for $\mathcal{C}^2$ (where $\mathcal{C} = \mathrm{RS}[n, k+1, d]$) where $c_1 = \frac{n\tau}{3d}$, $c_2 = \frac{2nc_0(\tau+2)}{3d}$ and N is the block length of $\mathcal{C}^2$.*

Proof. To analyze T' let $R^*(r,\cdot)$ be the closest degree k univariate polynomial (breaking ties arbitrarily) for each row r. Similarly construct $C^*(\cdot,c)$. We will use the following refinement of the Bivariate testing lemma of [88]:

Lemma 8.8 ([88, 13]). *There exists an universal constant $c_0 \leqslant 128$ such that the following holds. If $8k \leqslant n$ then $\delta(Q', \mathcal{C}^2) = \delta(Q', Q) \leqslant c_0 \cdot (\delta(R^*, Q') + \delta(C^*, Q'))$.*

The following proposition shows that the standard tester version of T' (that is T' with $\tau = 0$) is a robust tester–

Proposition 8.9. *T' with $\tau = 0$ is a $2c_0$ robust tester, where c_0 is the constant from Lemma 8.8.*

Proof. By the definition of the row polynomial R, for any row index r, the robustness of the tester with $b = 0$ and r, $\rho(Q', \langle b, r\rangle) = \delta(Q'(r,\cdot), R^*(r,\cdot))$. Similarly for $b = 1$, we have $\rho(Q', \langle b, c\rangle) = \delta(Q'(\cdot,c), C^*(\cdot,c))$. Now the expected robustness of the test is given by

$$\begin{aligned}\rho(Q') &= \Pr[b=0]\sum_{i=1}^{n}\Pr[r=i]\cdot\delta(Q'(r,\cdot),R^*(r,\cdot)) + \\ &\qquad \Pr[b=1]\sum_{j=1}^{n}\Pr[c=j]\cdot\delta(Q'(\cdot,c),C^*(\cdot,c)) \\ &= \frac{1}{2}(\delta(Q',R^*)+\delta(Q',C^*))\,.\end{aligned}$$

Using Lemma 8.8, we get $\delta(Q', Q) \leqslant 2c_0\rho(Q')$, as required. □

From the description of T', it is clear that it has a *partitioned* query pattern. There are two partitions: one for the rows (corresponding to the choice $b = 0$) and one for the columns (corresponding to the choice $b = 1$).

Lemmas 8.2 and 8.3 prove Theorem 8.7 where c_0 is the constant from Lemma 8.8. □

8.6.2 General Reed-Muller Codes

We now turn our attention to testing of general Reed-Muller codes. Recall that $\mathrm{RM}_q(k,m)$ the linear code consisting of evaluations of m-variate polynomials over $\mathbb{F}_q$ of *total degree* at most k at all points in $\mathbb{F}_q^m$. [6] To test codewords of $\mathrm{RM}_q(k,m)$, we need to, given a function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ as a table of values, test if f is close to an m-variate polynomial of total degree k. We will do this using the following natural and by now well-studied *low-degree test* which we call the *lines test*: pick a random line in $\mathbb{F}_q^m$ and check if the restriction of f on the line is a univariate polynomial of degree at most k. In order to achieve tolerance,

[6] The results of the previous section were for polynomials which had degree in each *individual* variable bounded by some value; here we study the total degree case.

we will modify the above test to accept if the restriction of f on the picked line is within distance τ from some degree k univariate polynomial, for a threshold τ. Using the analysis of the low-degree test from [42], we can show the following.

Theorem 8.8. *For $0 \leqslant \tau \leqslant 1 - \sqrt{k/q}$ and $q = \Omega(k)$, $\mathrm{RM}_q(k, m)$ is (c_1, c_2, p) testable with $c_1 = \frac{n\tau}{3d}$, $c_2 = \frac{3(\tau+2)n}{d}$ and $p = n^{1-1/m}$ where $n = q^m$ and d are the block length and the distance of the code.*

Proof. Recall that our goal is to test if a given function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ is close to an m-variate polynomial of total degree k. For any $x, h \in \mathbb{F}_q^m$, a line passing through x in direction h is given by the set $L_{x,h} = \{x + th | t \in \mathbb{F}_q\}$. Further define $P^f_{x,h}(\cdot)$ to be the univariate polynomial of degree at most k which is closest (in Hamming distance) from the restriction of f on $L_{x,h}$. We will use the following result.

Theorem 8.9 ([42]). *There exists a constant c such that for all k, if q is a prime power that is at least ck, then given a function $f : \mathbb{F}_q^m \to \mathbb{F}_q$ with*

$$\rho \stackrel{\text{def}}{=} E_{x,h \in \mathbb{F}_q^m} \Pr_{t \in \mathbb{F}_q}[P^f_{x,h}(t) \neq f(x + th)] \leqslant \frac{1}{9},$$

there exists an m-variate polynomial g of total degree at most k such that $dist(f, g) \leqslant 2\rho$.

The above result clearly implies that the line test is robust which we record in the following corollary.

Corollary 8.10. *There exists a constant c such that the line test for $\mathrm{RM}_q(k, m)$ with $q \geqslant ck$ is 9-robust.*

The line test picks a random line by choosing x and h randomly. Consider the case when h is fixed. It is not hard to check that for there is a partition of $\mathbb{F}_q^m = X_1 \cup \cdots \cup X_q$ where each X_i has size q^{m-1} such that $\cup_{x \in X_i} L_{x,h} = \mathbb{F}_q^m$. In other words:

Proposition 8.10. *The line test has a partitioned query pattern.*

The proposed tolerant tester for $\mathrm{RM}_q(k, m)$ is as follows: pick $x, h \in \mathbb{F}_q^m$ uniformly at random and check if the input restricted to $L_{x,h}$ is τ-close to some univariate polynomial of degree k. If so accept, otherwise reject. When the threshold τ satisfies $\tau \leqslant 1 - \sqrt{k/q}$, the test can be implemented in polynomial time [63]. From Corollary 8.10, Proposition 8.10, Lemmas 8.2 and 8.3, the above is indeed a tolerant tester for $\mathrm{RM}_q(k, m)$, and Theorem 8.8 follows. □

8.7 Bibliographic Notes and Open Questions

The results in this chapter (other than those in Section 8.5.2) were presented in [57]. Results in Section 8.5.2 appear in [26].

In the general context of property testing, the notion of tolerant testing was introduced by Parnas *et al.* [83] along with the related notion of distance approximation. Parnas *et al.* also give tolerant testers for clustering. We feel that codeword-testing is a particularly natural instance to study tolerant testing. (In fact, if LTCs were defined solely from a coding-theoretic viewpoint, without their relevance and applications to PCPs in mind, we feel that it is likely that the original definition itself would have required tolerant testers.)

The question of whether the natural tester for $C_1 \otimes C_2$ is a robust one was first explicitly asked by Ben-Sasson and Sudan [14]. P. Valiant showed that in general, the answer to the question is no. Dinur, Sudan and Wigderson [30] further show that the answer is positive if at least one of C_1 or C_2 is a *smooth* code, for a certain notion of smoothness. They also show that any non-adaptive and *bi-regular* binary linear LTC is a smooth code. A bi-regular LTC has a tester that in every query probes the same number of positions and every bit in the received word is queried by the same number of queries. The latter requirement (in the terminology of this chapter) is that the tester is $(\langle 1, q\rangle, \langle 0, 0\rangle, 1)$-smooth, where the tester makes q queries. The result of [30] however only works with constant query complexity. Note that for such an LTC, Lemma 8.1 implies that the code is also tolerant testable.

Obtaining non-trivial lower bounds on the the block length of codes that are locally testable with very few (even 3) queries is an extremely interesting question. This problem has remained open and resisted even moderate progress despite all the advancements in constructions of LTCs. The requirement of having a tolerant local tester is a stronger requirement. While we have seen that we can get tolerance with similar parameters to the best known LTCs, it remains an interesting question whether the added requirement of tolerance makes the task of proving lower bounds more tractable. In particular,

Open Question 8.1. *Does there exists a code with constant rate and linear distance that has a tolerant tester that makes constant number of queries ?*

This seems like a good first step in making progress towards understanding whether locally testable codes with constant rate and linear distance exist, a question which is arguably one of the grand challenges in this area. For interesting work in this direction which proves that such codes, if they exist, cannot also be *cyclic*, see [10].

The standard testers for Reed-Muller codes considered in Section 8.6 (and hence, the tolerant testers derived from them) work only for the case when the size of the field is larger than the degree of the polynomial being tested. Results in Chapter 7 and those in [74] give a standard tester for Reed-Muller codes which works for all fields. These testers do have a partitioned query pattern– however, it is not clear if the testers are robust. Thus, our techniques to convert it into a tolerant tester fail. It will be interesting to show the following result.

Open Question 8.2. *Design tolerant testers for RM codes over any finite field.*

Chapter 9

CONCLUDING REMARKS

9.1 Summary of Contributions

In this thesis, we looked at two different relaxation of the decoding problems for error correcting codes: list decoding and property testing.

In list decoding we focused on the achieving the best possible tradeoff between the rate of a code and the fraction of errors that could be handled by an efficient list decoding algorithm. Our first result was an explicit construction of a family of code along with efficient list decoding algorithm that achieves the list decoding capacity. That is, for any rate $0 < R < 1$, we presented folded Reed-Solomon codes of rate R along with polynomial time list decoding algorithms that can correct up to $1 - R - \varepsilon$ fraction of errors (for any $\varepsilon > 0$). This was the first result to achieve the list decoding capacity for any rate (and over any alphabet) and answered one of the central open questions in coding theory. We also constructed explicit codes that achieve the tradeoff above with alphabets of size $2^{O\left(\varepsilon^{-4}\log(1/\varepsilon)\right)}$, which are not that much bigger than the optimal size of $2^{\Omega(1/\varepsilon)}$.

For alphabets of fixed size, we presented explicit codes along with efficient list decoding algorithms that can correct a fraction of errors up to the so called Blokh-Zyablov bound. In particular, these give binary codes of rate $\Omega(\varepsilon^3)$ that can be list decoded up to a $1/2 - \varepsilon$ fraction of errors. These codes have rates that come close to the optimal rate of $\Theta(\varepsilon^2)$ that can be achieved by random codes with exponential time list decoding algorithms.

A key ingredient in designing codes over smaller alphabets was to come up with optimal list recovery algorithms. We also showed that the list recovery algorithm for Reed-Solomon codes due to Guruswami and Sudan is the best possible. We also presented some explicit bad list decoding configurations for list decoding Reed Solomon codes.

Our contributions in property testing of error correcting codes are two-fold. First, we presented local testers for Reed-Muller codes that use near optimal number of queries to test membership in Reed-Muller codes over fixed alphabets. Second, we defined a natural variation of local testers called tolerant testers and showed that they had comparable parameters with those of the best known LTCs.

9.2 Directions for Future Work

Even though we made some algorithmic progress in list decoding and property testing of error correcting codes, there are many questions that are still left unanswered. We have

highlighted the open questions throughout the thesis. In this section, we focus on some of the prominent ones (and related questions that we did not talk about earlier).

9.2.1 List Decoding

The focus of this thesis in list decoding was on the optimal tradeoff between the rate and list decodability of codes. We first highlight the algorithmic challenges in this vein (most of which have been highlighted in the earlier chapters).

- The biggest unresolved question from this thesis is to come up with explicit codes over fixed alphabets that achieve the list decoding capacity. In particular is there a polynomial time construction of a binary codes of rate $\Omega(\varepsilon^2)$ that be list decoded in polynomial time up to $1/2 - \varepsilon$ fraction of errors? (Open Question 4.1)

- A less ambitious goal than the one above would be to give a polynomial time construction of a binary code with rate $\Omega(\varepsilon)$ that can be list decoded up to $1 - \varepsilon$ fraction of *erasures*? Erasures are a weaker noise model that we have not considered in this thesis. In the erasure noise model, the only kind of errors that are allowed is the "dropping" of a symbol during transmission. Further, it is assumed that the receiver knows which symbols have been erased. For this weaker noise model, one can show that for rate R the optimal fraction of errors that can be list decoded is $1 - R$.

- Another less ambitious goal would be to resolve the following question. Is there a polynomial time construction of a code that can be list decoded up to $1/2 - \varepsilon$ fraction of errors with rate that is asymptotically better than ε^3?

- Even though we achieved list decoding capacity for large alphabets (that is, for rate R code, list decode $1 - R - \varepsilon$ fraction of errors), the worst case list size was $n^{\Omega(1/\varepsilon)}$, which is very far from the $O(1/\varepsilon)$ worst case list size achievable by random codes. A big open question is to come up with explicit codes that achieve the list decoding capacity with constant worst case list size. As a less ambitious goal would be to reduce the worst case list size to n^c for some constant c that is independent of ε. (See Section 3.7)

We now look at some questions that relate to the combinatorial aspects of list decoding.

- For a rate R Reed-Solomon code, can one list decode more than $1 - \sqrt{R}$ fraction of errors in polynomial time? (Open Question 6.2)

- To get to within ε of list decoding capacity can one prove a lower bound on the worst case list size? For random codes it is known that list of size $O(1/\varepsilon)$ suffice but no general lower bound is known.[1]

[1] For high fraction of errors, tight bounds are known [65].

- For codes of rate R over fixed alphabet of size $q > 2$, can one show existence of linear codes that have $q^{o(1/\varepsilon)}$ many codewords in any Hamming ball of radius $1-H_q(R)-\varepsilon$? (See discussion in Section 2.2.1)

9.2.2 Property Testing

Here are some open questions concerning property testing of codes.

- The biggest open question in this area is to answer the following question. Are there codes of constant rate and linear distance that can be locally tested with constant many queries?

- A less ambitious (but perhaps still very challenging) goal is to show that the answer to the question above is no for 3 queries.

- Can one show that the answer to the first question (or even the second) is no, if one also puts in the extra requirement of tolerant testability? (Open Question 8.1)

BIBLIOGRAPHY

[1] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing Reed-Muller codes. *IEEE Transactions on Information Theory*, 51(11):4032–4039, 2005.

[2] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley and Sons, Inc., 1992.

[3] Sigal Ar, Richard Lipton, Ronitt Rubinfeld, and Madhu Sudan. Reconstructing algebraic functions from mixed data. *SIAM Journal on Computing*, 28(2):488–511, 1999.

[4] Sanjeev Arora, László Babai, Jacques Stern, and Z Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54:317–331, 1997.

[5] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the intractibility of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[6] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[7] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[8] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing(STOC)*, pages 21–31, 1991.

[9] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[10] László Babai, Amir Shpilka, and Daniel Stefankovic. Locally testable cyclic codes. *IEEE Transactions on Information Theory*, 51(8):2849–2858, 2005.

[11] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust PCPs of proximity, shorter PCPs and application to coding. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 2004.

[12] Eli Ben-Sasson, Swastik Kopparty, and Jaikumar Radhakrishnan. Subspace polynomials and list decoding of Reed-Solomon codes. In *Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 207–216, 2006.

[13] Eli Ben-Sasson and Madhu Sudan. Simple PCPs with poly-log rate and query complexity. In *Proceedings of 37th ACM Symposium on Theory of Computing (STOC)*, pages 266–275, 2005.

[14] Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Structures and Algorithms*, 28(4):387–402, 2006.

[15] Elwyn Berlekamp. *Algebraic Coding Theory*. McGraw Hill, New York, 1968.

[16] Elwyn Berlekamp. Factoring polynomials over large finite fields. *Mathematics of Computation*, 24:713–735, 1970.

[17] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24:384–386, 1978.

[18] Daniel Bleichenbacher, Aggelos Kiayias, and Moti Yung. Decoding of interleaved Reed Solomon codes over noisy data. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 97–108, 2003.

[19] E. L. Blokh and Victor V. Zyablov. Existence of linear concatenated binary codes with optimal correcting properties. *Prob. Peredachi Inform.*, 9:3–10, 1973.

[20] E. L. Blokh and Victor V. Zyablov. *Linear Concatenated Codes*. Moscow: Nauka, 1982. (in Russian).

[21] Manuel Blum, Micahel Luby, and Ronit Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[22] Donald G. Chandler, Eric P. Batterman, and Govind Shah. Hexagonal, information encoding article, process and system. *US Patent Number 4,874,936*, October 1989.

[23] C. L. Chen and M. Y. Hsiao. Error-correcting codes for semiconductor memory applications: A state-of-the-art review. *IBM Journal of Research and Development*, 28(2):124–134, 1984.

[24] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, and David A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[25] Qi Cheng and Daqing Wan. On the list and bounded distance decodability of Reed-Solomon codes. *SIAM Journal on Computing*, 37(1):195–209, 2007.

[26] Don Coppersmith and Atri Rudra. On the robust testability of product of codes. In *Electronic Colloquium on Computational Complexity (ECCC) Tech Report TR05-104*, 2005.

[27] Don Coppersmith and Madhu Sudan. Reconstructing curves in three (and higher) dimensional spaces from noisy data. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 136–142, June 2003.

[28] Philippe Delsarte, Jean-Marie Goethals, and Florence Jessie MacWilliams. On generalized Reed-Muller codes and their relatives. *Information and Control*, 16:403–442, 1970.

[29] Peng Ding and Jennifer D. Key. Minimum-weight codewords as generators of generalized Reed-Muller codes. *IEEE Trans. on Information Theory.*, 46:2152–2158, 2000.

[30] Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of LDPC codes. In *Proceedings of the 10th International Workshop on Randomization and Computation (RANDOM)*, pages 304–315, 2006.

[31] Rodney G. Downey, Michael R. Fellows, Alexander Vardy, and Geoff Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM Journal on Computing*, 29(2):545–570, 1999.

[32] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003.

[33] Ilya I. Dumer. Concatenated codes and their multilevel generalizations. In V. S. Pless and W. C. Huffman, editors, *Handbook of Coding Theory*, volume 2, pages 1911–1988. North Holland, 1998.

[34] Peter Elias. List decoding for noisy channels. *Technical Report 335, Research Laboratory of Electronics, MIT*, 1957.

[35] Peter Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37:5–12, 1991.

[36] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[37] Uriel Fiege and Daniele Micciancio. The inapproximability of lattice and coding problems with preprocessing. *Journal of Computer and System Sciences*, 69(1):45–67, 2004.

[38] Eldar Fischer. The art of uninformed decisions: A primer to property testing. *Bulletin of the European Association for Theoretical Computer Science*, (75):97–126, 2001.

[39] Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for boolean properties. *Theory of Computing*, 2(9):173–183, 2006.

[40] G. David Forney. *Concatenated Codes*. MIT Press, Cambridge, MA, 1966.

[41] G. David Forney. Generalized Minimum Distance decoding. *IEEE Transactions on Information Theory*, 12:125–131, 1966.

[42] Katalin Friedl and Madhu Sudan. Some improvements to total degree tests. In *Proceedings of the 3rd Israel Symp. on Theory and Computing Systems (ISTCS)*, pages 190–198, 1995.

[43] Peter Gemmell, Richard Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approxiamte functions. In *Proceeding of the 23rd Symposium on the Theory of Computing (STOC)*, pages 32–42, 1991.

[44] Oded Goldreich. Short locally testable codes and proofs (Survey). *ECCC Technical Report TR05-014*, 2005.

[45] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[46] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost linear length. In *Proceedings of 43rd Symposium on Foundations of Computer Science (FOCS)*, pages 13–22, 2002.

[47] Andrew Granville. The arithmetic properties of binomial coefficients. In `http://www.cecm.sfu.ca/organics/papers/granville/`, 1996.

[48] Venkatesan Guruswami. Limits to list decodability of linear codes. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, pages 802–811, 2002.

[49] Venkatesan Guruswami. *List decoding of error-correcting codes*. Number 3282 in Lecture Notes in Computer Science. Springer, 2004. (Winning Thesis of the 2002 ACM Doctoral Dissertation Competition).

[50] Venkatesan Guruswami. Algorithmic results in list decoding. In *Foundations and Trends in Theoretical Computer Science (FnT-TCS)*, volume 2. NOW publishers, 2006.

[51] Venkatesan Guruswami, Johan Håstad, Madhu Sudan, and David Zuckerman. Combinatorial bounds for list decoding. *IEEE Transactions on Information Theory*, 48(5):1021–1035, 2002.

[52] Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 658–667, 2001.

[53] Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting Gilbert-Varshamov bound for low rates. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 756–757, 2004.

[54] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Transactions on Information Theory*, 51(10):3393–3400, October 2005.

[55] Venkatesan Guruswami and Anindya C. Patthak. Correlated Algebraic-Geometric codes: Improved list decoding over bounded alphabets. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, October 2006.

[56] Venkatesan Guruswami and Atri Rudra. Limits to list decoding Reed-Solomon codes. In *Proceedings of the 37th ACM Symposium on Theory of Computing (STOC)*, pages 602–609, May 2005.

[57] Venkatesan Guruswami and Atri Rudra. Tolerant locally testable codes. In *Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 306–317, 2005.

[58] Venkatesan Guruswami and Atri Rudra. Explicit capacity-achieving list-decodable codes. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10, May 2006.

[59] Venkatesan Guruswami and Atri Rudra. Limits to list decoding Reed-Solomon codes. *IEEE Transactions on Information Theory*, 52(8), August 2006.

[60] Venkatesan Guruswami and Atri Rudra. Better binary list-decodable codes via multilevel concatenation. In *Proceedings of the 11th International Workshop on Randomization and Computation (RANDOM)*, 2007. To Appear.

[61] Venkatesan Guruswami and Atri Rudra. Concatenated codes can achieve list decoding capacity. *Manuscript*, June 2007.

[62] Venkatesan Guruswami and Atri Rudra. Explicit bad list decoding configurations for Reed Solomon codes of constant rate. *Manuscript*, May 2006.

[63] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.

[64] Venkatesan Guruswami and Madhu Sudan. Extensions to the Johnson bound. *Manuscript*, February 2001.

[65] Venkatesan Guruswami and Salil Vadhan. A lower bound on list size for list decoding. In *Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 318–329, 2005.

[66] Venkatesan Guruswami and Alexander Vardy. Maximum-likelihood decoding of Reed-Solomon codes is NP-hard. *IEEE Transactions on Information Theory*, 51(7):2249–2256, 2005.

[67] Richard W. Hamming. Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 29:147–160, April 1950.

[68] Prahladh Harsha. *Robust PCPs of Proximity and Shorter PCPs*. PhD thesis, Massachusetts Institute of Technology, 2004.

[69] Edward F. Assmus Jr. and Jennifer D. Key. Polynomial codes and finite geometries. In V. S. Pless and W. C. Huffman, editors, *Handbook of Coding Theory*, volume 2, pages 1269–1343. North Holland, 1998.

[70] Jørn Justesen and Tom Høholdt. Bounds on list decoding of MDS codes. *IEEE Transactions on Information Theory*, 47(4):1604–1609, May 2001.

[71] Charanjit S. Jutla, Anindya C. Patthak, and Atri Rudra. Testing polynomials over general fields. manuscript, 2004.

[72] Charanjit S. Jutla, Anindya C. Patthak, Atri Rudra, and David Zuckerman. Testing low-degree polynomials over prime fields. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 423–432, 2004.

[73] Tali Kaufman and Simon Litsyn. Almost orthogonal linear codes are locally testable. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 317–326, 2005.

[74] Tali Kaufman and Dana Ron. Testing polynomials over general fields. *SIAM Journal on Computing*, 36(3):779–802, 2006.

[75] Victor Y. Krachkovsky. Reed-Solomon codes for correcting phased error bursts. *IEEE Transactions on Information Theory*, 49(11):2975–2984, November 2003.

[76] Michael Langberg. Private codes or Succinct random codes that are (almost) perfect. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 325–334, October 2004.

[77] Rudolf Lidl and Harald Niederreiter. *Introduction to Finite Fields and their applications*. Cambridge University Press, Cambridge, MA, 1986.

[78] Yu. V. Linnik. On the least prime in an arithmetic progression. I. The basic theorem. *Mat. Sbornik N. S.*, 15(57):139–178, 1944.

[79] Antoine C. Lobstein. The hardness of solving subset sum with preprocessing. *IEEE Transactions on Information Theory*, 36:943–946, 1990.

[80] Florence Jessie MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. Elsevier/North-Holland, Amsterdam, 1981.

[81] Robert J. McEliece. On the average list size for the Guruswami-Sudan decoder. In *7th International Symposium on Communications Theory and Applications (ISCTA)*, July 2003.

[82] D. E. Muller. Application of boolean algebra to switching circuit design and to error detection. *IEEE Transactions on Computers*, 3:6–12, 1954.

[83] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.

[84] Farzad Parvaresh and Alexander Vardy. Multivariate interpolation decoding beyond the Guruswami-Sudan radius. In *Proceedings of the 42nd Allerton Conference on Communication, Control and Computing*, 2004.

[85] 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.

[86] Anindya C. Patthak. *Error Correcting Codes : Local-testing, List-decoding, and Applications to Cryptography*. PhD thesis, University of Texas at Austin, 2007.

[87] Larry L. Peterson and Bruce S. Davis. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, San Francisco, 1996.

[88] A. Polishchuk and D. A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 194–203, 1994.

[89] Irving S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *IEEE Transactions on Information Theory*, 4:38–49, 1954.

[90] Irving S. Reed and Gustav Solomon. Polynomial codes over certain finite fields. *SIAM Journal on Applied Mathematics*, 8:300–304, 1960.

[91] Oded Regev. Improved inapproximability of lattice and coding problems with preprocessing. *IEEE Transactions on Information Theory*, 50:2031–2037, 2004.

[92] Dana Ron. Property Testing. In S. Rajasekaran, P. M. Pardalos, J. H. Reif, and J. D. P. Rolim, editors, *Handbook of Randomization*, pages 597–649, 2001.

[93] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[94] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[95] Michael Sipser and Daniel Spielman. Expander codes. *IEEE Transactions on Information Theory*, 42(6):1710–1722, 1996.

[96] Madhu Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*. ACM Distinguished Theses Series. Lecture Notes in Computer Science, no. 1001, Springer, 1996.

[97] Madhu Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[98] Madhu Sudan. List decoding: Algorithms and applications. *SIGACT News*, 31:16–27, 2000.

[99] Madhu Sudan. Lecture notes on algorithmic introduction to coding theory, Fall 2001. Lecture 15.

[100] Madhu Sudan. Lecture notes on algorithmic introduction to coding theory, Fall 2001. Lecture 6.

[101] Amnon Ta-Shma and David Zuckerman. Extractor Codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, 2001.

[102] Christian Thommesen. The existence of binary linear concatenated codes with Reed-Solomon outer codes which asymptotically meet the Gilbert-Varshamov bound. *IEEE Transactions on Information Theory*, 29(6):850–853, November 1983.

[103] Paul Valiant. The tensor product of two codes is not necessarily robustly testable. In *Proceedings of the 9th International Workshop on Randomization and Computation (RANDOM)*, pages 472–481, 2005.

[104] Jacobus H. van Lint. *Introduction to Coding Theory*. Graduate Texts in Mathematics **86**, (Third Edition) Springer-Verlag, Berlin, 1999.

[105] Stephen B. Wicker and Vijay K. Bhargava, editors. *Reed-Solomon Codes and Their Applications*. John Wiley and Sons, Inc., September 1999.

[106] John M. Wozencraft. List Decoding. *Quarterly Progress Report, Research Laboratory of Electronics, MIT*, 48:90–95, 1958.

[107] Chaoping Xing. Nonlinear codes from algebraic curves improving the Tsfasman-Vladut-Zink bound. *IEEE Transactions on Information Theory*, 49(7):1653–1657, 2003.

[108] Victor A. Zinoviev. Generalized concatenated codes. *Prob. Peredachi Inform.*, 12(1):5–15, 1976.

[109] Victor A. Zinoviev and Victor V. Zyablov. Codes with unequal protection. *Prob. Peredachi Inform.*, 15(4):50–60, 1979.

[110] Victor V. Zyablov and Mark S. Pinsker. List cascade decoding. *Problems of Information Transmission*, 17(4):29–34, 1981 (in Russian); pp. 236-240 (in English), 1982.

VITA

Atri Rudra was born in Dhanbad, India which is also where he grew up. He got a Bachelors in Technology in Computer Science and Engineering from Indian Institute of Technology, Kharagpur in 2000. After spending two years at IBM India Research Lab in New Delhi, he joined the graduate program at University of Texas at Austin in 2002. He moved to the Computer Science and Engineering department at the University of Washington in 2004, where he earned his Master of Science and Doctor of Philosophy degrees in 2005 and 2007 respectively under the supervision of Venkatesan Guruswami. Beginning September 2007, he will be an Assistant Professor at the University at Buffalo, New York.