UNIVERSITY OF CALIFORNIA, SAN DIEGO

Algebraic List-Decoding of Error-Correcting Codes

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering
(Communications Theory and Systems)

by

Farzad Parvaresh

2007

The dissertation of Farzad Parvaresh is approved, and it is acceptable in quality and form for publication on microfilm:

Chair

University of California, San Diego

2007

To my parents

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost, I like to thank my advisor, Alexander Vardy. Now that I look back to the time I got admission from UCSD, and started working with Alex, it all looks like a miracle to me. I truly believe, during the past five years, I have been evolved as a scientist and a person, and more than any other person, Alex was influential in this change. This work would not be possible with the very friendly and supportive research environment that he provided for me and his constant attention and believe on what I was doing for my research. I feel very lucky to have had Alex Vardy as my advisor and look forward to learn and work with him in future.

I like to thank my dissertation committee members. I am very grateful to Paul Siegel. I learned a lot about coding theory from his classes and enjoyed attending STAR group meetings. I am grateful to Alon Orlitsky. Alon's class on information theory had a huge impact on attracting me toward coding and information theory. Also, I could not imagine doing my Ph.D. at UCSD without his help with my visa. Jack Wolf has been a role model for me. I am hoping that some day I would be able to explain complex subjects in simple terms like him. I am grateful to Ilya Dumer. Ilya's insight that the bounds of Chapter 2 are optimal in a certain sense makes the results of Chapter 2 much stronger. And, although he had to drive from Riverside to San Diego each time for my qualification exam and defense, he kindly agreed to be in my defense committee.

I had a chance to do an internship at Bell laboratories under the supervision of Alexei Ashikhmin. I learned a lot about quantum codes from Alexei during the short period that I spent at Bell Labs. I am very grateful to him for his magic introduction to tough concept of quantum codes in a language which is plausible for a coding guy. I also like to thank Gerhard Kramer, Tom Marzetta, Emina Soljanin, Adriaan van Wijngaarden and others at Bell Labs mathematical research center for making my stay at Bell Labs an enjoyable one.

I am grateful to Bob McEliece for giving me a chance to visit his group at Caltech and Shuki Bruck for letting me to attend his group meetings during my stay at Caltech. I truly enjoyed the discussions in the meetings which helped me a lot to diversify my research.

I also like to thank Ralf Koetter. Ralf's suggestions on my research always helped me to understand the problem in a much deeper level . I hope someday I will be as comfortable as he is with algebra. I like to acknowledge Madhu Sudan and Venkat Guruswami for their invaluable comments on my research.

I would also like to thank my officemates, Nandu Santi for all the enjoyable discussions we had, Jun Ma, Navin Kashyap and Moshe schwartz. I like to thank my friends, Houman Ghajari for being a great roommate and great friend, Amin Shameli, Alireza Vakil Amirkhizi, Mohsen Bayati, Navid Ehsan, Mohammad Farazian , Zeinab Taghavi, Hossein Taghavi, Kambiz Samadi, Amir Farajidana, Shafigh Shirinfard, Aydin Babakhani, Olgica Milenkovic, Narayana P. Santhanam, Mostafa Al-Khamy, Mihailo Stojnic, Ashay Dhamdhere, Mishal Algharabally, Sharon Aviran, Liwen Yu, Seyhan Karkulak, Mike Cheng, Joseph Soriaga, Henry Pfister, Junsheng Han, Patrick Amihood, Chaitanya Rao, Sormeh Shadbakht, Ali Vakili, Weiyu Xu, Frédérique Oggier, Haris Vikalo, Vincent Bohossian, Yuval Cassuto, Michael Langberg, Sarah Fargol, Edwin Soedarmadji, and many others that made my graduate life a fruitful one.

I am specially grateful to my family, to my father, Ahmad, for always being there for me, paying attention to all the details of my work and giving his help and support in all aspect of my life, to my mom, Parvin, for always encouraging and complementing my work and giving me confidence on what I am doing and my sister, Maryam, which her sense of humor made my telephone calls more fun. Although, I did not have a chance to visit them during my study at UCSD, I had their love and support even from such a long distance. And, I am very grateful to my uncles, Hushang, Farideh, Parynaz, Shahram, Peyman, Ebi, Ann and my second cousins for their support and for making me feel that I am back home here in US.

The results of the Chapter 2 have been presented, in part, at the 43^{ed} *Annual Allerton Conference on Communication, Control, and Computing*, Parvaresh, Farzad; Vardy, Alexander, and at the *2006 International Symposium on Information Theory (ISIT)*, Parvaresh, Farzad; Taghavi, Mohammad; Vardy, Alexander. The dissertation author was the primary investigator and author of both papers.

The material of Chapter 3 has been presented, in full, at the 46th *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Parvaresh, Farzad; Vardy, Alexander. The dissertation author was the primary investigator and author of this paper.

We have presented the results of Chapter 4, in part, at the *2003 International Symposium on Information Theory (ISIT)*, Parvaresh, Farzad; Vardy, Alexander. The dissertation author was the primary investigator and author of this paper.

The results of Chapter 5 are published, in part, in proceedings of *2004 International Symposium on Information Theory (ISIT)*, Parvaresh, Farzad; Vardy, Alexander. The dissertation author was the primary investigator and author of this paper.

VITA

2001	B. S., Electrical Engineering, Sharif University of Technology, Iran.
2003	M. S., Electrical Engineering (Communications Theory and Systems), University of California San Diego.
2007	Ph. D., Electrical Engineering (Communications Theory and Systems), University of California San Diego.

PUBLICATIONS

F. Parvaresh and A. Vardy, *Correcting Errors beyond the Guruswami-Sudan Radius,* in preparation for submission to the Journal of the ACM.

F. Parvaresh and A. Vardy, *Multivariate Interpolation Decoding of Reed-Solomon codes Beyond the Guruswami–Sudan Radius*, in preparation for submission to the IEEE Transaction on Information Theory.

F. Parvaresh, M. El-Khamy, R. J. McEliece and A. Vardy, *Algebraic List-Decoding of Reed-Solomon Product Codes*, in preparation.

F. Parvaresh, M. H. Taghavi and A. Vardy, *On the performance of Multivariate Decoding of Reed-Solomon Codes,* Proceeding of the IEEE Symposium on Information Theory (ISIT), Seattle, WA, July 2006.

F. Parvaresh and A. Vardy, *Correcting Errors beyond the Guruswami-Sudan Radius in Polynomial Time,* 46th Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, PA, 2005.

F. Parvaresh and A. Vardy, *Multivariate interpolation decoding beyond the Guruswami-Sudan radius*, Proceedings of the 42nd Annual Allerton Conference on Communications, Control and Computing, Monticello, IL, October 2004.

F. Parvaresh and A. Vardy, *Polynomial Matrix-Chain Interpolation in Sudan-type Reed-Solomon Decoders,* Proceeding of the IEEE Symposium on Information Theory (ISIT), Chicago, IL, July 2004.

F. Parvaresh and A. Vardy, *Multiplicity Assignment for Algebraic Decoding of Reed–Solomon Codes,* Proceeding of the IEEE Symposium on Information Theory (ISIT), Yokohama, Japan, July 2003.

ABSTRACT OF THE DISSERTATION

Algebraic List-Decoding of Error-Correcting Codes

by

Farzad Parvaresh

Doctor of Philosophy in Electrical Engineering

(Communications Theory and Systems)

University of California San Diego, 2007

Professor Alexander Vardy, Chair

This dissertation is concerned with algebraic list-decoding of error-correcting codes. During the past decade, significant advances in this are were achieved. The breakthrough papers of Sudan, Guruswami & Sudan, and Koetter & Vardy showed that the well-known Reed-Solomon (and other algebraic) codes can correct many more errors – in the list-decoding sense – than previously thought possible. Herein, we extend the theory developed in these seminal papers, and improve upon the results reported therein.

We first extend the bivariate polynomial interpolation method of Guruswami-Sudan to *multivariate* interpolation decoding. To this end, we develop a new decoding algorithm for Reed-Solomon codes, which decodes some M codewords together. We show that if the channel errors are synchronized then, with high probability, our multivariate interpolation decoding algorithm corrects up to $n\big(1 - R^{M)/(M+1)}\big)$ errors in a Reed-Solomon code of length n and rate R. This is much higher than the Guruswami-Sudan decoding radius of $n\big(1 - R^{1/2}\big)$.

Next, we consider the case of *adversarial errors*. We introduce a new family of codes that have a polynomial-time list-decoder correcting a fraction of $1 - \varepsilon$ adversarial errors for a code of rate $\Omega\big(\varepsilon/\log(1/\varepsilon)\big)$. The best previously known

results required a rate of $O(\varepsilon^2)$ for the same error-correction radius. In addition to the transition from bivariate to multivariate interpolation, we also modify the Reed-Solomon codes in an essential way. Reed-Solomon encoders view a message as a polynomial $f(X)$, and produce the corresponding codeword by evaluating $f(X)$ at n distinct elements of $\mathbb{F}_q$. In Chapter 3, given $f(X)$, we compute one or more related polynomials $g_i(X)$ and produce the corresponding codeword by evaluating *all* these polynomials. The a priori correlation between $f(X)$ and $g_i(X)$ then enables us to recover the encoded message from the output of the interpolation process.

Finally, we consider *soft-decision* list-decoding. Koetter & Vardy have shown that algebraic soft-decision decoding can be achieved by converting symbol probabilities observed at the channel output into interpolation multiplicities. Such conversion is known as the *multiplicity assignment problem*. In Chapter 4, we first recast the multiplicity assignment problem into a geometric framework, and use this framework to establish certain properties of the optimal solution. We then devise a sub-optimal solution based upon optimization of second-order statistics. Specifically, we minimize the Chebyshev bound on the probability of failure of the soft-decision decoder. This leads to coding gains of 0.25 dB to 0.75 dB, as compared to the Koetter-Vardy multiplicity assignment algorithm.

It is widely recognized that bivariate (or multivariate) interpolation is the most computationally intensive step in algebraic list-decoding algorithms. In Chapter 5, we show that bivariate polynomial interpolation is equivalent to computing a certain chain product of polynomial matrices. We then derive a dynamic-programming algorithm to parse this matrix-chain product in an optimal way. This leads to a reduction in the computational complexity of the interpolation process by a factor of at least two, as compared to the interpolation algorithm of Koetter.

CHAPTER 1

Introduction

" ... The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point ... If the channel is noisy it is not in general possible to reconstruct the original message or the transmitted signal with *certainty* by any operation on the received signal E. There are however, ways of transmitting the information which are optimal in combating noise. "

A Mathematical Theory of Communication – C.E. SHANNON

American Heritage dictionary defines communication as the exchange of thoughts, messages, or information, as by speech, signals, writing, or behavior. During the act of communicating, unwanted noise usually intervenes between the source and the destination. For example, during a conversation in a subway, clatter makes the communication harder than during a conversation in a quiet room. We speak louder and sometimes repeat a word or a sentence if our audience doesn't receive the message. However, sometimes when we miss some of the words in a sentence during the conversation we are nevertheless able to recover the whole sentence. The spoken language has *redundancy*. This helps us to recover parts of a sentence which we might miss during the conversation. Here, we look at communication in a more abstract setting. Consider the following scenario.

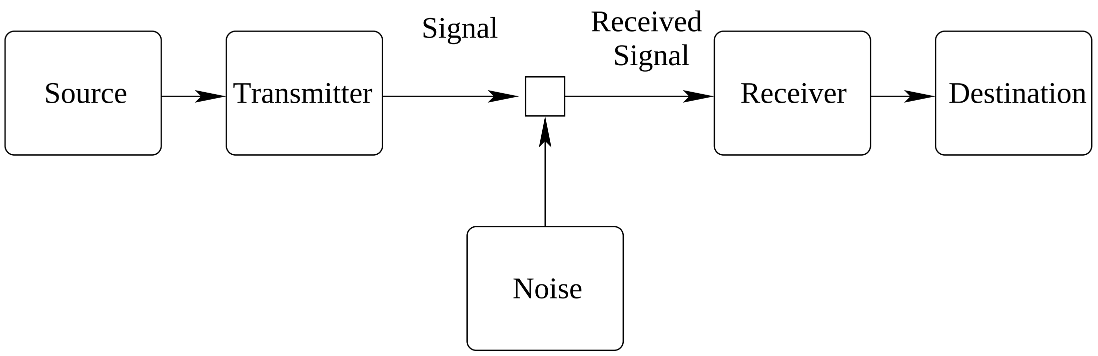


Figure 1.1: Schematic of a generic communication system

Problem 1.1. Sue makes a sentence out of k words and asks Terry to send the sentence to Richard, who is waiting at the other end of a line at a bus-stop. Terry writes an *encoded* version of the sentence, which has n words, on a piece of paper and passes it to the next person in line. Each person then passes the paper to the next person till it gets to Richard. There are t people between Richard and Terry. It is late afternoon and people in the line are clumsy, so each person might randomly change **one** of the words in the sentence to something else.

How should Terry *encode* Sue's message, i.e. add redundancy to it so that Richard can still recover the sentence when t words are changed? How can Richard *decode* the message from the corrupted sentence?

Most communication systems can be modeled as the on in Problem 1.1. A general communication system consists of a source that produces a message, a transmitter/encoder that maps the message to a signal, a noise source that corrupts the signal in a specific way, and a receiver/decoder that reconstructs the message by exploiting the knowledge it has of the encoder and the channel (see Figure 1.1).

Shannon, in his seminal paper [Sha48], proves that, when k/n is smaller than a certain value, called *channel capacity*, there exist encoding and decoding algorithms that transmit the message with probability of error that approaches zero when k and n tend to infinity. The proof is *existential*. It just shows that such an encoder and decoder exist for every channel, but it does not provide a way to *actually* construct the encoder and/or the decoder.

The theory of error-correcting codes dates back to work of Shannon, and Hamming [Sha48, Ham50] in the early 1950's. The main challenge of the theory is to invent efficient encoding and decoding algorithms that transmit information reliably close to channel capacity. Since the 1950's, many advanced coding techniques have been introduced using powerful mathematical tools such as algebra, geometry, probability, and combinatorics. The resulting *error-correcting codes* have now diverse applications in computer science, engineering, bio-informatics, and many other areas.

One of the first classes of codes invented during the early days of coding theory was Reed-Solomon codes [RS60]. Nowadays, Reed-Solomon codes are ubiquitous. They have been used in many different applications, ranging from deep-space communication to computer disk drives and DVD's. Around the time of their invention, Peterson [Pet60] and Berlekamp [Ber68] came up with very efficient algorithms for decoding Reed-Solomon codes up to half of their minimum distance.

In Problem 1.1, if Terry uses Reed-Solomon codes to encode Sue's message, then Richard can recover the message whenever t, the number of words changed, is smaller than $\frac{1}{2}(n-k)$. However, Shannon [Sha48] showed, if the people in line replace the words of Terry's message with other words in the dictionary uniformly random then, for large enough k, n and t, there is an encoding that lets Richard recover the message whenever

$$k/n \;\leqslant\; 1 - H_q\,(t/n)\,, \tag{1.1}$$

where $H_q(\cdot)$ is the q-ary entropy function and q is the total number of words in the dictionary. For large q, we know that $H_q(t/n)$ is roughly equal to t/n, so Shannon predicts that when t is smaller than about $n-k$ reliable communication is possible. In other words, we should be able to send the message through a line that has *twice the length* of the line wherein a Reed-Solomon code with the Peterson-Berlekamp-Massey [Pet60, Ber68, Mas69] decoding algorithm is used for communication.

For almost 30 years, there were no significant improvements on the Berlekamp-Massey decoding algorithm for Reed-Solomon codes. In a breakthrough result, Sudan [Sud97] and Guruswami–Sudan [GS99] showed that we can decode many more errors with Reed-Solomon codes than previously thought possible. Using their algorithm, the number of people in the line can be extended to $\lceil n - \sqrt{(k-1)n}\,\rceil$. Is this the best we can hope for? In Chapter 2, we extend the algorithm of Guruswami-Sudan to higher dimensions. We show that for a certain type of Reed-Solomon codes, using an adaptive recovery algorithm, we can often correct up to

$$n\left\lceil 1 - \left(\frac{k-1}{n}\right)^{\frac{M}{M+1}}\right\rceil$$

errors, where M is a constant. For large M, this bound approaches the Shannon bound when q, the size of the dictionary, tends to infinity.

However, let's make communication a little bit harder for Terry and Richard.

Problem 1.2. Suppose that it is April Fool's day, and people in the line are interested to corrupt the sentence *maliciously* so that that Richard would have a hard time to recover Sue's message. They are aware of the algorithm that Terry uses for encoding. Also, they have enough computational power and time to decide which word of the sentence to change, and they can pick any word they want from the dictionary. In addition, they are free to communicate with each other. The only restriction is that they are allowed to change *at most t words* in the sentence and not more.
The question is how many errors can Richard tolerate in the sentence in this scenario?

Theorem 1.3. *Suppose there are q words in the dictionary. Then Richard can list-recover the Sue's message, assuming Terry encodes the message in a the correct way [Gur05b], if and only if*

$$k/n \;<\; 1 - H_q\left(t/n\right), \tag{1.2}$$

where $H_q(\cdot)$ is the q-ary entropy function. Here, list-recovery means that Richard obtains a small list of sentences and Sue's message is one of them.

Amazingly, the bound of (1.2) is very similar to the Shannon bound of (1.1). The proof is also *existential*, based on random coding arguments. If Terry uses Reed-Solomon codes, Richard can still correct up to $\lceil n - \sqrt{(k-1)n}\,\rceil$ errors in this scenario. However, there is some evidence [GR05, BKR05] that the people in line can come up with a strategy which makes Richard unable to recover Sue's message whenever the number of errors is more than $\lceil n - \sqrt{(k-1)n}\,\rceil$. In Chapter 3, we show that a variant of Reed-Solomon codes, combined with a modified version of the decoding algorithm of Chapter 2, can get close to the bound of (1.2) when k/n is small. Later, Guruswami and Rudra [GR06c] showed that a compressed version of the codes we define in Chapter 3 can get close to the ultimate bound of $n-k$ for any k/n between zero and one.

1.1 Contributions

This dissertation is concerned with list-decoding of error-correcting codes. List decoding was introduced by Elias [Eli57] and Wozencraft [Woz58] in the late 1950's. Nevertheless, most algebraic decoding algorithms that were known up to the work of Sudan [Sud97] in 1997 are *unique decoding* algorithms. These algorithms can decode up to half the minimum distance of a code, where the *minimum distance* is the smallest Hamming distance between two distinct codewords. Indeed, if we receive a word that is half way between two codewords that are at the minimum distance from each other, then in order to decode beyond the half-the-distance bound we have to output *both* codewords. Hence, unique decoding fails in these situations.

On the other hand, list-decoding algorithms make it possible to decode beyond the half-the-distance bound. In contrast to unique decoding, list decoders are allowed to produce a *small list* of codewords as the output. For Reed-Solomon codes, we know (cf. [McE03b]) that the list size is **one** on the average, when decoding up to the Johnson bound. Thus, most of the time, a list decoder would output only a single codeword. However, the difference is that the list decoder can decode many received words for which unique decoding fails.

During the past decade, significant advances on in the area of algebraic list-decoding were achieved. In a series of breakthrough papers, Sudan [Sud97], Guruswami and Sudan [GS99], and Koetter and Vardy [KV03a] showed that the well-known Reed-Solomon codes (and other algebraic codes) can correct many more errors, in the list-decoding sense, than previously thought possible. In this dissertation, we extend the theory developed in these seminal papers, and improve upon the results reported therein.

In Chapter 2, we extend the bivariate polynomial interpolation method of Guruswami-Sudan [GS99] to *multivariate* interpolation decoding. To this end, we develop a new decoding algorithm for Reed-Solomon codes, which decodes some M codewords together, where M is an arbitrary constant. We show that if the channel errors are synchronized – occur at the same positions in all the M codewords – then, with high probability, our multivariate interpolation decoding algorithm corrects up to $n\big(1 - R^{M)/(M+1)}\big)$ errors in a Reed-Solomon code of length n and rate R. This is much higher than the Guruswami-Sudan decoding radius of $n\big(1 - R^{1/2}\big)$. The results of this chapter have been presented in part at the 43^{rd} ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING and in part at the 2006 INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY (ISIT). A journal version is currently in preparation for submission to the IEEE TRANSACTION FOR INFORMATION THEORY.

In Chapter 3, we consider the case of *adversarial errors*. We introduce a new family of error-correcting codes that have a polynomial-time list-decoder correcting a fraction of $1 - \varepsilon$ adversarial errors for a code of rate $\Omega\big(\varepsilon/\log(1/\varepsilon)\big)$. The best previously known results for polynomial-time list-decoding of adversarial errors required a rate of $O(\varepsilon^2)$ to achieve the same error-correction radius. In addition to the transition from bivariate interpolation to multivariate interpolation, our results in Chapter 3 are based on parting ways with Reed-Solomon codes: rather than devising a better list-decoder for these codes, we devise better codes. Reed-Solomon encoders view a message as a polynomial $f(X)$ over a field $\mathbb{F}_q$, and produce the corresponding codeword by evaluating $f(X)$ at n distinct elements of $\mathbb{F}_q$. In Chapter 3, given $f(X)$, we first compute one or more related polynomials $g_1(X), g_2(X), \ldots, g_{M-1}(X)$ and produce the corresponding codeword by evaluating *all* these polynomials. Correlation between $f(X)$ and $g_i(X)$ then provides the information we need to recover the encoded message from the output of the multivariate interpolation process. Our construction furthermore leads to a class of codes, called *folded Reed-Solomon codes*, introduced by Guruswami and Rudra [GR06c] that can be list-decoded up to the radius of $1 - R$, which is the ultimate limit for list-decoding of adversarial errors. The results of Chapter 3 have been presented in part at the 46^{th} ANNUAL IEEE SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE (FOCS). A journal version is in preparation for submission to the JOURNAL OF THE ACM (JACM).

Next, we consider *soft-decision* list-decoding. Koetter and Vardy [KV03a] have shown that algebraic soft-decision decoding of Reed-Solomon codes can be achieved by converting symbol probabilities observed at the channel output into interpolation multiplicities in the Guruswami-Sudan algorithm. Such conversion is known as the *multiplicity assignment problem*. In Chapter 4, we first recast the multiplicity assignment problem into a geometric framework in Euclidean space, and use this framework to establish certain properties of the optimal solution. We then devise a sub-optimal solution to the multiplicity assignment problem based upon the optimization of second-order statistics. Specifically, we minimize the Chebyshev bound on the probability of failure of the soft-decision decoder. Assuming BPSK-modulation over an AWGN channel, this leads to coding gains of 0.20 dB to 0.75 dB for Reed-Solomon codes of length 255 and 15, respectively, as compared to the Koetter-Vardy multiplicity assignment algorithm. We have presented some of these results at the 2003 INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY (ISIT). A section for a tutorial monograph on algebraic list decoding of Reed-Solomon codes, based upon Chapter 4, is in preparation.

It is widely recognized that bivariate (or multivariate) polynomial interpolation is the most computationally intensive step in algebraic list-decoding of Reed-Solomon codes. Consequently, many different algorithms for bivariate interpolation have been developed in the past decade. In Chapter 5, we show that

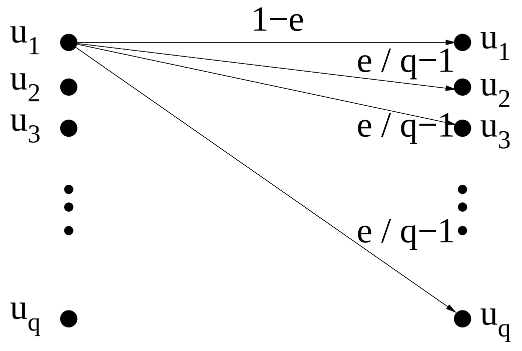


Figure 1.2: q-ary symmetric channel

bivariate polynomial interpolation is equivalent to computing a certain matrix-chain product of polynomial matrices. We then derive a dynamic-programming algorithm to parse this matrix-chain product in an optimal way. This leads to a reduction in the computational complexity of the interpolation process by a factor of at least two (in most cases, even more), as compared to the iterative interpolation algorithm of Koetter. The results of this chapter were presented in part at the 2004 INTERNATIONAL SYMPOSIUM ON INFORMATION THEORY (ISIT).

1.2 Waiting for a bus in front of a pub

One of the most fundamental and ubiquitous models of a communication systems is the *q-ary symmetric channel*. During the transmission over a q-ary symmetric channel, each symbol is either transmitted as is with probability $1 - e$ or changed to any other symbol in the dictionary with probability $e/(q-1)$ (see Figure 1.2). The communication in Problem 1.1 becomes similar to the q-ary symmetric channel when the length of the transmitted message is large. The people in line change the words in the sentence with a certain probability e. Moreover, they change every word to some other word in the dictionary uniformly random. For this channel, our goal is to find how many errors Richard can correct when Terry uses Reed-Solomon codes for transmission. First, let's see how Terry would encode the message using Reed-Solomon codes.

1.2.1 Reed-Solomon codes

Let q be a power of a prime, and let $\mathbb{F}_q$ be the finite field with q elements. Reed-Solomon codes are obtained by evaluating polynomials of degree less than k in a set of points $\mathcal{D} = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{F}_q$, which is known as the *locator set* or the *evaluation set* (cf. [RR00, GV05]) of the code. Specifically, a *Reed-Solomon* code $\mathbb{C}_q(n,k)$ of *length* n and *dimension* k is defined as follows:

$$\mathbb{C}_q(n,k) \overset{\text{def}}{=} \{(f(x_1), \ldots, f(x_n)) : x_1, x_2, \ldots, x_n \in \mathcal{D}, f(X) \in \mathbb{F}_q[X], \deg f(X) < k\}$$

Therefore, in order to encode, Terry first picks a finite field $\mathbb{F}_q$ that has as many elements as the dictionary. Then she maps each word in Sue's sentence to an element of this finite field. Let's say that Sue's sentence is $\alpha_1\alpha_2 \ldots \alpha_k$, where $\alpha_1, \alpha_2, \ldots, \alpha_k$ are words in the dictionary. Then Terry maps this sentence into $u_1u_2 \ldots u_k$, where $u_1, u_2, \ldots, u_k$ are the corresponding elements of $\mathbb{F}_q$. The mapping between the dictionary and $\mathbb{F}_q$ is fixed, and is known also to Richard, who will use it to eventually recover Sue's message. Next, Terry constructs the polynomial $f(X) = u_1 + u_2X + u_3X^2 + \cdots + u_kX^{k-1}$ and evaluates this polynomial at n different elements of the finite field to generate the codeword $(c_1, c_2, \ldots, c_n) \in \mathbb{F}_q^n$, where $c_i = f(x_i)$ for all i. Now, she can use the inverse of the mapping between the dictionary and $\mathbb{F}_q$ to produce the n-word sentence $w_1w_2 \ldots w_n$ from the codeword $(c_1, c_2, \ldots, c_n)$.

Observe that one can reconstruct any polynomial of degree $< k$ from its values at any k points. Thus Terry adds redundancy to Sue's message by transmitting $n - k$ more evaluations of $f(X)$ than necessary. Hence if at least k out of the n transmitted symbols reach Richard without error, he can interpolate through these k symbols and thereby recover Sue's message. But, in our case, Richard doesn't know which symbols have been corrupted during the transmission. He only knows that with certain probability some of the symbols are corrupted and some are not. What would be his best strategy in this case?

One possible strategy for Richard would be to find a polynomial of degree $< k$ that passes through as many points of the received word as possible. The resulting optimization problem was first solved by Peterson [Pet60] in the early 1960's. Later, Berlekamp [Ber68] and Massey [Mas69] developed a much more efficient algorithm to solve the same problem. All these algorithms can decode up to $\frac{1}{2}(n-k)$ errors, and may fail if the number of errors is greater than $\frac{1}{2}(n-k)$. The Peterson and the Berlekamp-Massey algorithms are unique decoding algorithms. One can show that they decode correctly whenever the number of errors is less than half of the minimum distance of the code; we have already seen earlier that *any* unique decoding algorithm may fail if the number of errors exceeds this bound.

An alternative strategy for Richard becomes possible if he is allowed to recover Sue's message after a few tries. He looks at the received message and, based upon the information he has about Terry's encoding algorithm, generates several sentences as potential candidates for Sue's message. The number of candidates Richard generates is bounded polynomially in the length of the message. Whenever Sue's message is included in the list of sentences produced by Richard, we say that communication is successful.

Some 10 years ago, Sudan [Sud97] and Guruswami-Sudan [GS99] come up with a new decoding algorithm for Reed-Solomon codes. This algorithm outputs a list of possible codewords instead of just a single codeword. This algorithm also makes it possible for Richard to recover Sue's message even in the presence of more than $\frac{1}{2}(n-k)$ errors.

1.2.2 Guruswami-Sudan algorithm

As we have seen, a Reed-Solomon code of $\mathbb{C}_q(n,k)$ consists of evaluations of univariate polynomials $f(X)$ of degree $< k$ at n points of the finite filed $\mathbb{F}_q$. Thus the codeword that corresponds to $f(X)$ is $\boldsymbol{c} = (f(x_1), f(x_2), \ldots, f(x_n))$. We can think of this codeword as a set of zeros of the bivariate polynomial $Y - f(X)$ over $\mathbb{F}_q$. During transmission, some of the symbols in $\boldsymbol{c}$ get corrupted and we receive the word $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$. Let us suppose that some t symbols of $\boldsymbol{y}$ are in error. Using the received word $\boldsymbol{y}$, we first construct a bivariate polynomial $Q(X,Y)$ that passes through the points $\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$

with multiplicity m. Then $Q(X,Y)$ and $Y - f(X)$ intersect each other m times at those points where $\boldsymbol{y}$ is *not* in error – for these points, both $Q(x_i, y_i)$ and $y_i - f(x_i)$ are zero. How can we find $Y - f(X)$ using the fact that $Q(X,Y)$ and $Y - f(X)$ intersect in at least $m(n-t)$ many points?

Theorem 1.4. [Bézout] *Let $P(X,Y)$ and $Q(X,Y)$ be bivariate polynomials of total degree δ and d, respectively. If $P(X,Y)$ and $Q(X,Y)$* ***do not have common factors****, then they intersect in exactly δd points, counted with multiplicity.*

Thus, if we can find a polynomial $Q(X,Y)$ such that the number of points of intersection between $Q(X,Y)$ and $Y - f(X)$ is strictly greater than the product of their degrees, then $Y - f(X)$ and $Q(X,Y)$ must have a ***common factor***. Since $Y - f(X)$ is an irreducible polynomial, it has to divide $Q(X,Y)$ in this case. It follows that we can recover $Y - f(X)$ by factoring $Q(X,Y)$. This is the main idea behind the decoding algorithm of Guruswami and Sudan [GS99].

How can we find such a polynomial $Q(X,Y)$? First, let us define the $(1,k-1)$-weighted degree of a monomial X^aY^b as

$$\mathrm{w}\deg X^aY^b \overset{\text{def}}{=} a + (k-1)b$$

Next, we define the $(1,k-1)$-weighted degree of a bivariate polynomial as the maximum of the weighted degrees of its monomials.

Theorem 1.5. *There is a bivariate polynomial $Q(X,Y)$ that passes through all the points $\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$ with multiplicity m, whose $(1,k-1)$-weighted degree is bounded by*

$$\mathrm{w}\deg Q(X,Y) \leqslant \left\lceil \sqrt{n(k-1)m(m+1)} \right\rceil .$$

Moreover, given $\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$ and m, this polynomial $Q(X,Y)$ can be computed in time that is bounded by a polynomial in m and n.

Now that we have a bound on the weighted degree of $Q(X,Y)$, let us count the number of intersections between $Y - f(X)$ and $Q(X,Y)$. Let $p(X)$ be the univariate polynomial $Q(X, f(X))$. Then, by definition, $\deg p(X)$ is at most

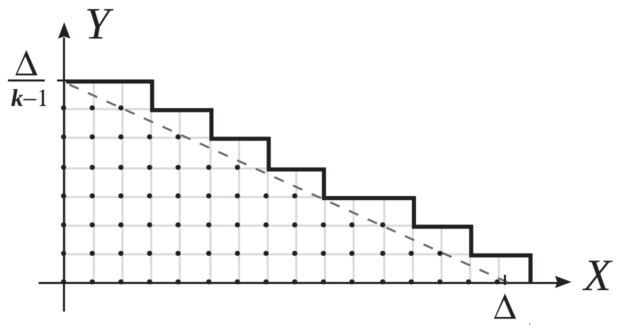


Figure 1.3: Monomials of weighted degree smaller than Δ

${}_{\mathbf{w}}\deg Q(X,Y)$. On the other hand, $p(X)$ has at least $m(n-t)$ zeros: for each received symbol that is not in error, we have $p(x_i) = Q(x_i, f(x_i)) = Q(x_i, y_i) = 0$ with multiplicity m. If the number of zeros of $p(X)$ is larger than its degree, then by the fundamental theorem of algebra $p(X)$ must be the all-zero polynomial. It follows that if $m(n-t) > {}_{\mathbf{w}}\deg Q(X,Y)$ then $Y - f(X)$ divides $Q(X,Y)$. Using the bound on ${}_{\mathbf{w}}\deg Q(X,Y)$ given in Theorem 1.5, we can guarantee that whenever the number of errors is bounded by

$$t \leqslant \left\lceil n\sqrt{\frac{k-1}{n}\left(1+\frac{1}{m}\right)}\,\right\rceil \tag{1.3}$$

$Y - f(X)$ will be a factor of $Q(X,Y)$, which means that the Guruswami-Sudan decoding will be successful.

1.2.3 Multivariate interpolation decoding algorithm

A careful look at the Guruswami-Sudan [GS99] list-decoding algorithm reveals that one of the essential elements in deriving the bound (1.3) on the decoding radius is the total number of bivariate monomials of weighted-degree smaller than some constant Δ. In the construction of $Q(X,Y)$, in order to guarantee a nonzero solution, we need to have $O(n)$ “available” monomials, where n is length of the code. As shown in Figure 1.3, the number of monomials of weighted-degree at most Δ is proportional to Δ^2. If we could “somehow” have more monomials of weighted-degree at most Δ, we would expect to find

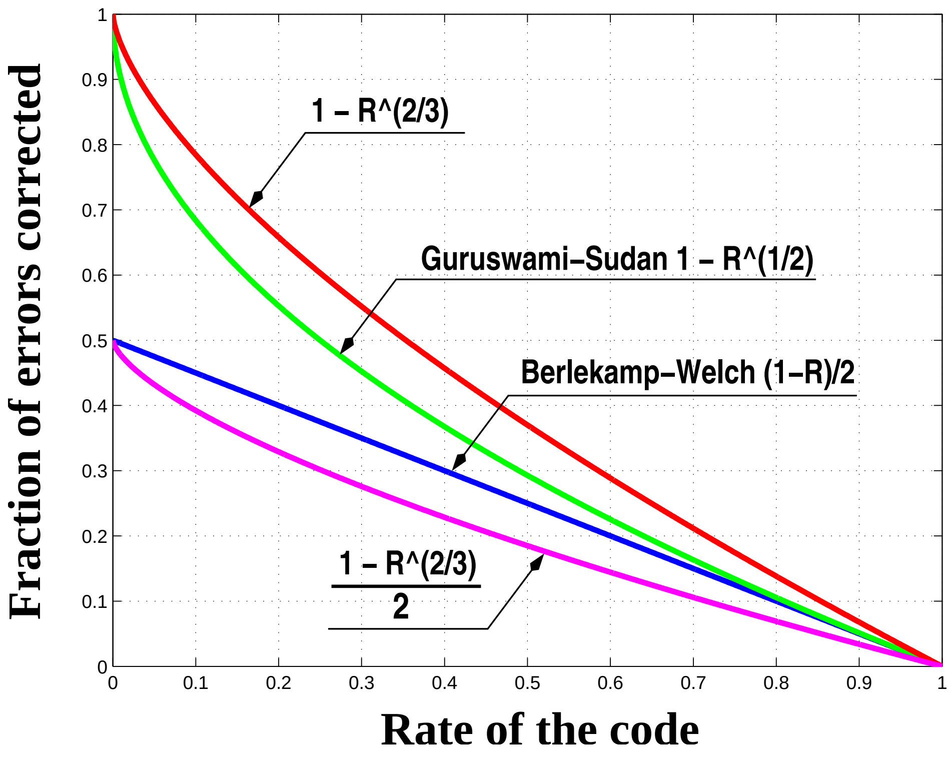
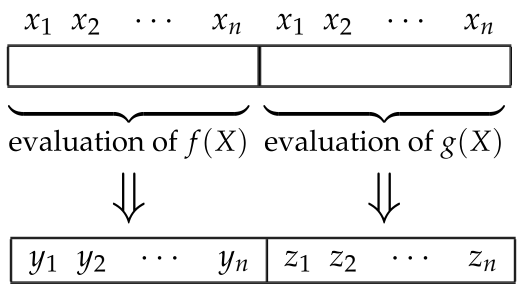


Figure 1.4: Error-correction radius of trivariate interpolation decoding

an interpolation polynomial with smaller weighted degree and, ultimately, a decoding algorithm with a better decoding radius than the Guruswami-Sudan bound (1.3).

One way to increase the number of monomials with a certain weighted-degree is to use *trivariate*, rather than bivariate, monomials. The number of trivariate monomials with weighted degree at most Δ is proportional to Δ^3, rather than Δ^2 in the bivariate case. Thus, for a given Δ, there are many more trivariate monomials of weighted degree at most Δ than bivariate monomials. However, in order to take advantage of this dimensionality gain," we need to construct *trivariate interpolation polynomials*. Hence, we need to interpolate through points in a three-dimensional space. To generate three-dimensional interpolation points, we will decode ***two*** Reed-Solomon codewords – let's say the evaluations of $f(X)$ and $g(X)$ – together, as shown below:


$x_1 \ x_2 \ \cdots \ x_n \ x_1 \ x_2 \ \cdots \ x_n$
evaluation of $f(X)$
evaluation of $g(X)$
$y_1 \ y_2 \ \cdots \ y_n$
$z_1 \ z_2 \ \cdots \ z_n$

where $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$ denote the two received vectors. To decode $\boldsymbol{y}$ and $\boldsymbol{z}$, we construct a *trivariate* polynomial $Q(X, Y, Z)$ of least $(1, k-1, k-1)$-weighted degree that passes through all the received points (x_i, y_i, z_i) for $i = 1, 2, \ldots, n$, with multiplicity m. We show in Chapter 2 that if the number of errors is upper bounded by

$$t \leqslant \left\lfloor n - n\sqrt[3]{R^2\left(1+\frac{1}{m}\right)\left(1+\frac{2}{m}\right)} - \frac{1}{m} \right\rfloor \xrightarrow{m\to\infty} n\left(1 - R^{2/3}\right) \qquad (1.4)$$

then the polynomial $Q(X, Y, Z)$ is such that $Q\big(X, f(X), g(X)\big) \equiv 0$. Thus, assuming that $f(X)$ and $g(X)$ can be recovered from the fact that $Q(X, f(X), g(X)) \equiv 0$, trivariate interpolation decoding makes it possible to correct up to $n(1 - R^{2/3})$ errors in a block of $2n$ symbols. This is, in fact, not very much (see Figure 1.4), as compared to half-the-distance or the Guruswami-Sudan bounds on the decoding radius. However, observe that, if two errors occur in the *same position* j of the two transmitted codewords, then both errors affect the same interpolation point (x_j, y_j, z_j). Following [CS03], we call such errors *synchronized*. It is not difficult to see that the combined effect of two synchronized errors is equivalent to a single error for our decoding algorithm. It follows that, if all the errors are synchronized, we can correct up to $2n(1 - R^{2/3})$ errors in a block of $2n$ symbols. This is much more than the $2n(1 - R^{1/2})$ errors corrected by the Guruswami-Sudan decoder (see again Figure 1.4).

However, the number of pairs of polynomials $f(X), g(X)$ of degree $< k$ that satisfy the equation $Q(X, f(X), g(X)) \equiv 0$ is, in general, super-polynomial (certainly so if $Q(X, Y, Z)$ is irreducible). Hence, in order to reduce the size of the set of possible solutions to the equation $Q(X, f(X), g(X)) \equiv 0$, we will adjoin one more equation $P(X, f(X), g(X)) \equiv 0$, where $P(X, Y, Z)$ is another interpolation polynomial that passes through all the received points with multiplicity m. We show in Chapter 2 that whenever the greatest common divisor of $Q(X, Y, Z)$ and $P(X, Y, Z)$ is in $\mathbb{F}_q[X]$, the number of pairs of polynomials $f(X)$ and $g(X)$ that satisfy $Q(X, f(X), g(X)) \equiv P(X, f(X), g(X)) \equiv 0$ is bounded by the product of the degrees of $Q(X, Y, Z)$ and $P(X, Y, Z)$. Thus, if the degrees of $Q(X, Y, Z)$ and $P(X, Y, Z)$ are small, then the number of possible solutions is also small. This idea is the basis for the general *recovery algorithm*, which we will introduce in Chapter 2.

Note that the argument above easily generalizes from trivariate to M-variate interpolation decoding. If all the errors are synchronized then M-variate interpolation decoding can, in principle, correct up to $(M-1)n\big(1 - \sqrt[M]{R^{M-1}}\big)$ such errors. This is much higher than the Guruswami-Sudan decoding radius of $n\big(1 - R^{1/2}\big)$. However, the success of decoding up to $(M-1)n\big(1 - \sqrt[M]{R^{M-1}}\big)$ is *not* guaranteed. All we can show analytically (cf. Chapter 2) is that such decoding *is* successful with a very high probability, at least in certain important cases.

1.3 Waiting for a bus on an April Fool's day

In Section 1.2.3, we observed that decoding Reed-Solomon codes in three or more dimensions improves substantially upon the $1 - \sqrt{R}$ decoding radius. However, the approach discussed in that section works only with a certain (high) probability — it is not guaranteed to work in the worst case. In the worst case, if the errors are introduced by a malicious adversary rather than by the q-ary symmetric channel, the list-decoder discussed in Section 1.2.3 decodes exactly the same number of errors as the Guruswami-Sudan [GS99] algorithm. In fact, numerous prior attempts [BKY03,BMS05,CS03,GI01] at breaking the bound of $1 - \sqrt{R}$ established by Guruswami and Sudan [GS99] in the worst case — or, equivalently, for adversarial errors — have been unsuccessful.

So, suppose that on April Fool's day, Terry encodes Sue's message into a sentence for transmission to Richard, but people in the line maliciously corrupt

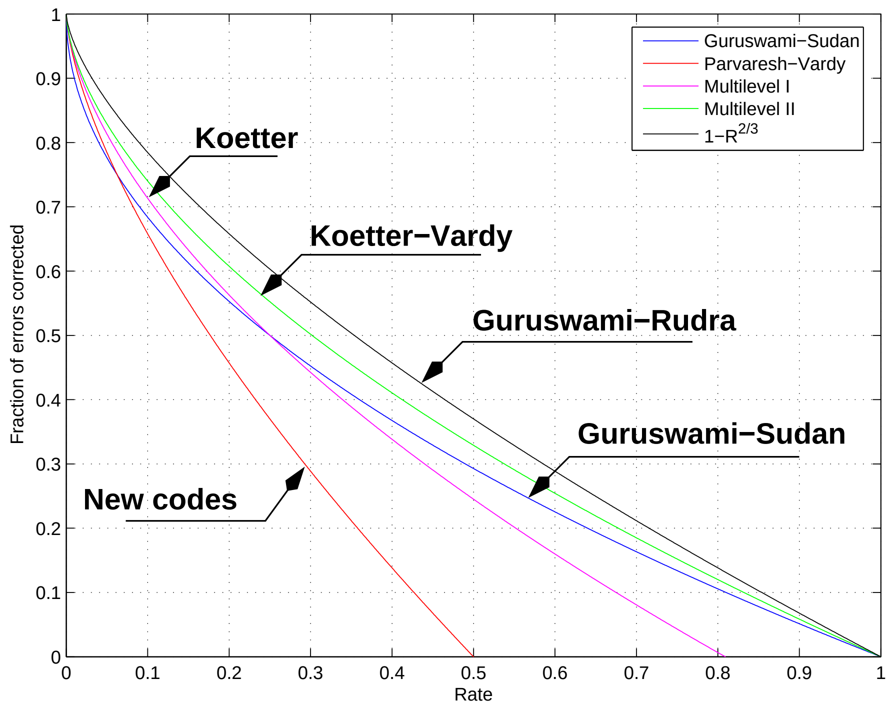


Figure 1.5: Error-correction radius of different codes in the worst-case

Terry's sentence so as to make its recovery as difficult as possible for Richard. The question we study in Chapter 3 is as follows:

> What is the largest fraction of errors Richard can correct, in the worst case, when Terry uses the best possible encoding?

In this setting, Guruswami [Gur05b] showed that there *exist* codes decoding algorithms that enable Richard to decode the message as long as the number of errors in the sentence is bounded by

$$t \ < \ n\big(1 - H_q^{-1}(R)\big) \tag{1.5}$$

where R is the rate of the code, n is its length, q is the number of words in the dictionary, and $H_q(\cdot)$ is the q-ary entropy function. Unfortunately, the proof of this result is existential rather than constructive: we do not know how to *construct* such codes in polynomial time, let alone *decode* them in polynomial time.

In Chapter 3, we explicitly construct a class of codes and develop a polynomial-time list decoding algorithm for these codes which exceeds the Guruswami-Sudan bound of $1-\sqrt{R}$ in the worst-case, for adversarial errors. The codes we construct are a certain highly-structured *nonlinear* subset of Reed-Solomon codes. The decoding radius we achieve with these codes is $1-\sqrt[M+1]{(MR)^M}$, where M is a constant. This radius becomes higher than the Guruswami-Sudan radius of $1-\sqrt{R}$ for rates $R \leqslant 1/16$. Recently, Guruswami and Rudra [GR05] have introduced a further improvement to our codes. With this improvement, one can decode up to the radius of $1-\sqrt[M+1]{R^M}$. When M tends to infinity, this meets the bound of (1.5), for large q.

In the trivariate decoding algorithm presented in Chapter 2, the decoder needs at least **two** "independent" interpolation polynomials in order to decode. It basically constructs a system of polynomial equations with $f(X)$ and $g(X)$ as the unknowns (recall that a codeword is the evaluation of $f(X)$ and $g(X)$ at n distinct points of the finite field). For some error patterns, all the interpolation polynomials that the decoder can compute have a common factor, and so the algorithm fails. To resolve this problem, we "shift" one of the two interpolation polynomials to the encoder. Explicitly, we fix a polynomial $\mathcal{T}(X,Y,Z)$ and only allow the encoder to encode those pairs of polynomials $f(X)$, $g(X)$ for which $\mathcal{T}(X,f(X),g(X)) \equiv 0$. We choose $\mathcal{T}(X,Y,Z)$ to be irreducible of sufficiently high degree, and hence *relatively prime* to the interpolation polynomial $Q(X,Y,Z)$. Thus, the decoder can now use both $\mathcal{T}(X,Y,Z)$ and the interpolation polynomial to decode. Since $\mathcal{T}(X,Y,Z)$ and $Q(X,Y,Z)$ are relatively prime, the number of solutions to the system of equations $Q(X,f(X),g(X)) \equiv \mathcal{T}(X,f(X),g(X)) \equiv 0$ is bounded by the product of the degrees of $Q(X,Y,Z)$ and $\mathcal{T}(X,Y,Z)$.

Of course, there will be a penalty in the rate of the code, since we are not transmitting all pairs of polynomials $f(X)$, $g(X)$ but only those that satisfy the equation $\mathcal{T}(X,f(X),g(X)) \equiv 0$. Luckily, the trade-off between rate and decoding radius is in our favor for rates $\leqslant 1/16$ and the resulting decoder improves upon the Guruswami-Sudan decoding radius of $1-\sqrt{R}$.

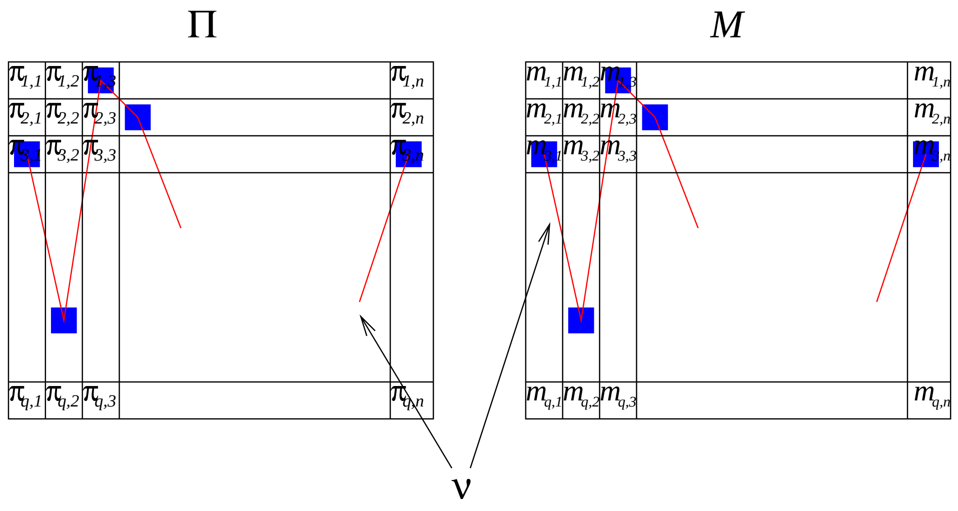
1.4 A new kind of Tic-Tac-Toe with multiplicity

During the bus ride, Terry and Richard decide to play the following game. The game is played on two identical tables (boards) of size $q \times n$ with respect to a threshold parameter Δ. One of the tables is filled with real numbers between zero and one. We denote this table by Π. The other table, which we denote by M, is empty. Each player has to fill out M with real numbers $m_{i,j}$ subject to the constraint $\sum_{i=1}^{q} \sum_{j=1}^{n} m_{i,j}^2 = 1$. The player who achieves a better *score* wins the game, where the score is defined as follows. First, let us define a *path* in the tables Π and M as an n-tuple $\nu = (\nu_1, \nu_2, \ldots, \nu_n)$ with each ν_i is between 1 and q. The elements of $\Pi = [\pi_{i,j}]$ that lie on the path ν are $\pi_{\nu_1,1}, \pi_{\nu_2,2}, \ldots, \pi_{\nu_n,n}$. Similarly, the elements of $M = [m_{i,j}]$ on the path ν are $m_{\nu_1,1}, m_{\nu_2,2}, \ldots, m_{\nu_n,n}$. Now if $m_{\nu_1,1} + m_{\nu_2,2} + \cdots + m_{\nu_n,n}$ is strictly greater than the threshold Δ, we add the product $\pi_{\nu_1,1}\pi_{\nu_2,2} \cdots \pi_{\nu_n,n}$ to the score. The total score is then the sum of the scores over all the q^n possible paths.

As it happens, this game is closely related to soft-decision decoding of Reed-Solomon codes. Here, Π is the reliability matrix ($\pi_{i,j}$ is the probability that symbol $\alpha_i \in \mathbb{F}_q$ was transmitted, given the received symbol y_j), while M is the interpolation multiplicity matrix. For more details on this, see [KV03a].

In the Guruswami-Sudan algorithm, we compute the interpolation polynomial $Q(X, Y)$ that passes through each of the received points with multiplicity m. This corresponds to a single path in the above game. But what if we require $Q(X, Y)$ to pass not only through the received points, but also through other points, and moreover allow $Q(X, Y)$ to have different multiplicities at different points. Can this improve the performance of the decoding algorithm? If so, what is the best assignment of such interpolation multiplicities?

We shall see in Chapter 4 that for large m, the best normalized multiplicities are precisely the elements of the matrix M which gives the highest possible score in the game played by Terry and Richard. This multiplicity matrix M with the highest score would lead to the smallest *probability of failure* in Sudan-type soft-decision decoders, at least in the probabilistic model proposed in [KV03a].



Π
M
$\pi_{1,1}$ $\pi_{1,2}$ $\pi_{1,3}$ $\pi_{1,n}$
$\pi_{2,1}$ $\pi_{2,2}$ $\pi_{2,3}$ $\pi_{2,n}$
$\pi_{3,1}$ $\pi_{3,2}$ $\pi_{3,3}$ $\pi_{3,n}$
$\pi_{q,1}$ $\pi_{q,2}$ $\pi_{q,3}$ $\pi_{q,n}$
$m_{1,1}$ $m_{1,2}$ $m_{1,3}$ $m_{1,n}$
$m_{2,1}$ $m_{2,2}$ $m_{2,3}$ $m_{2,n}$
$m_{3,1}$ $m_{3,2}$ $m_{3,3}$ $m_{3,n}$
$m_{q,1}$ $m_{q,2}$ $m_{q,3}$ $m_{q,n}$
ν

Figure 1.6: Tic-Tac-Toe with multiplicity

Specifically, for a given multiplicity assignment M and reliability matrix Π, the probability of failure is given by $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$, where $\mathcal{S}_M$ is the score random variable and $\Delta(M)$ is a deterministic function of M. Koetter and Vardy give an algorithm which finds the multiplicity assignment that maximizes the expected value of $\mathcal{S}_M$. They also show that when the length of the code tends to infinity, the variance of $\mathcal{S}_M$ tends to zero. Thus, asymptotically, their algorithm is optimal. However, for finite-length codes, a better multiplicity assignment can be derived by directly optimizing $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$. This is a difficult optimization problem, and an exact solution to this problem remains unknown (nobody knows the optimal strategy in the game played by Richard and Terry). In Chapter 4, we consider the Chebyshev bound on the probability of failure $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$, and find the multiplicity assignment that minimizes this bound. This leads to coding gains of 0.20 dB to 0.75 dB for Reed-Solomon codes of length 255 and 15, respectively, as compared to the Koetter-Vardy multiplicity assignment. We also develop in Section 4.3 in Chapter 4 geometric framework for the multiplicity assignment problem, which "embeds" the required optimization in a high-dimensional Euclidean space. Using this geometric framework, we are able to convert any solution derived for the asymptotic case of large (infinite) multiplicities to the case of finite multiplicities.

We observe that the multiplicity assignments we derive using the Chebyshev bound have been recently improved in [EMH04, EM05, RK05] using better upper bounds (such as the Chernoff bound) on the probability of failure. However, while the multiplicity assignment developed using the Chebyshev bound in Chapter 4 remains simple to implement, the algorithms of [EMH04,EM05,RK05] become increasingly complex, and difficult to implement in practice.

CHAPTER 2

Multivariate decoding of Reed-Solomon codes

We present a new decoding algorithm for Reed-Solomon codes. The algorithm attempts to decode $M-1$ transmitted codewords together, using M-variate polynomial interpolation. It is shown that if the channel errors are *synchronized* – occur in the same positions in all the $M-1$ codewords – this algorithm can, in principle, correct up to $n\big(1-R^{(M-1)/M}\big)$ errors in a Reed-Solomon code of length n and rate R, which is significantly higher than the Guruswami-Sudan decoding radius of $n\big(1-R^{1/2}\big)$. The first nontrivial case $M=3$ is discussed and analyzed in detail. For this case, we show how to achieve trivariate interpolation using (a generalized version of) Koetter's iterative algorithm. We then recover the transmitted polynomials $f(X)$ and $g(X)$ from the output of this algorithm. In contrast to the bivariate case, the recovery of $f(X)$ and $g(X)$ requires at least *two* polynomials that satisfy the interpolation constraints. In fact, our recovery method is based upon computing certain resultants of the elements of a Gröbner basis for the ideal of *all* such polynomials.

The synchronized errors scenario was previously studied by Bleichenbacher, Kiayias, and Yung (BKY) and by Coppersmith and Sudan. Our algorithm corrects many more errors than the "simultaneous polynomial reconstruction" algorithm of BKY and/or the "curve reconstruction" algorithm of Coppersmith-

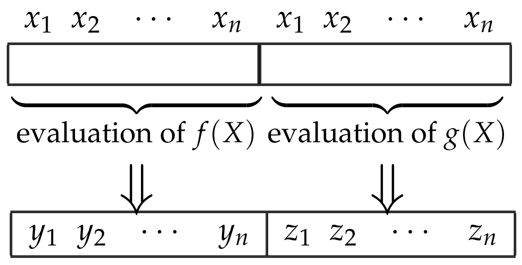
Sudan. Moreover, the algorithms of BKY and Coppersmith-Sudan are *probabilistic*: both assume a memoryless q-ary symmetric channel, where the BKY algorithm fails with probability $O(1/q)$ while the Coppersmith-Sudan algorithm fails with probability at least $O(n^M/q)$. These probabilities are much higher than the error rates typically required in applications of Reed-Solomon codes. In contrast, our algorithm is *deterministic*. In order to achieve deterministic decoding in polynomial time, the algorithm gradually reduces its decoding radius until the number of codewords within a certain decoding region becomes polynomially bounded.

2.1 Introduction

It was recognized early on that decoding Reed-Solomon codes is equivalent to the problem of reconstructing univariate polynomials from their noisy evaluations. Conventional Berlekamp-Massey decoding [Mas69] attempts to solve this problem using *univariate polynomial interpolation*. Specifically, suppose a codeword $\big(f(x_1), f(x_2), \ldots, f(x_n)\big)$ of a Reed-Solomon code $\mathbb{C}_q(n,k)$ was transmitted and a vector $(y_1, y_2, \ldots, y_n) \in \mathbb{F}_q^n$ was received. Then the Berlekamp-Massey algorithm essentially tries to construct a univariate polynomial of degree less than k that passes through as many as possible of the received points $y_1, y_2, \ldots, y_n$. The breakthrough achieved by Sudan [Sud97] and Guruswami-Sudan [GS99] is due in large part to the transition from univariate to *bivariate polynomial interpolation*. Specifically, the Guruswami-Sudan decoding algorithm [GS99] first constructs a nonzero bivariate polynomial $\mathcal{Q}(X,Y)$ of least $(1, k-1)$-weighted degree that passes through $\underline{\text{all}}$ the points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ with prescribed multiplicities, then finds all polynomials $f(X)$ of degree $< k$ such that $\mathcal{Q}\big(X, f(X)\big) \equiv 0$.

This work was motivated by the following question. What if we try to interpolate not in one dimension (conventional decoding) and not in two dimensions (Guruswami-Sudan decoding), but in *three or more dimensions*? As we shall see, provided the channel errors are synchronized (cf. [BKY03, CS03]), multivariate interpolation makes it possible to deterministically decode substantially beyond the Guruswami-Sudan error-correction radius of $n(1-\sqrt{R})$ in polynomial time.

Multivariate interpolation decoding. To set-up an interpolation problem in three dimensions, we decode together *two* codewords – say, the evaluations of $f(X)$ and $g(X)$ – as shown below:



where $\boldsymbol{y}=(y_1,y_2,y_3,\ldots,y_n)\in\mathbb{F}_q^n$ and $\boldsymbol{z}=(z_1,z_2,z_3,\ldots,z_n)\in\mathbb{F}_q^n$ denote the two received vectors. To decode $\boldsymbol{y}$ and $\boldsymbol{z}$, we first construct a *trivariate* polynomial $\mathcal{Q}(X,Y,Z)$ of least $(1,k-1,k-1)$-weighted degree that passes through the n points $(x_1,y_1,z_1),(x_2,y_2,z_2),\ldots,(x_n,y_n,z_n)$, each with multiplicity m. We will show in Section 2.3 that if the number of errors is upper bounded by

$$\left\lfloor n - n\sqrt[3]{R^2\left(1+\frac{1}{m}\right)\left(1+\frac{2}{m}\right)} - \frac{1}{m}\right\rfloor \xrightarrow{m\to\infty} n\left(1-R^{2/3}\right) \qquad (2.1)$$

then the polynomial $\mathcal{Q}(X,Y,Z)$ satisfies $\mathcal{Q}\big(X,f(X),g(X)\big)\equiv 0$. Thus, assuming that $f(X)$ and $g(X)$ can be now recovered, trivariate interpolation decoding makes it possible to correct up to $n(1-R^{2/3})$ errors in a block of $2n$ symbols. This is, in fact, not very much (see Figure 2.1). However, observe that if two errors occur in the *same position* j of the two transmitted codewords, then both errors affect the same interpolation point (x_j,y_j,z_j). Following [CS03], we call such errors *synchronized*. As we shall see, the combined effect of two synchronized errors is equivalent to a single error for our decoding algorithm. It follows that if all the errors are synchronized, we can correct up to $2n(1-R^{2/3})$ errors in a block of $2n$ symbols. This is much more than the $2n(1-R^{1/2})$ errors corrected by the Guruswami-Sudan decoder (cf. Figure 2.1).

The argument above easily generalizes from trivariate to M-variate interpolation decoding. In the general case, we will decode together $M-1$ transmitted

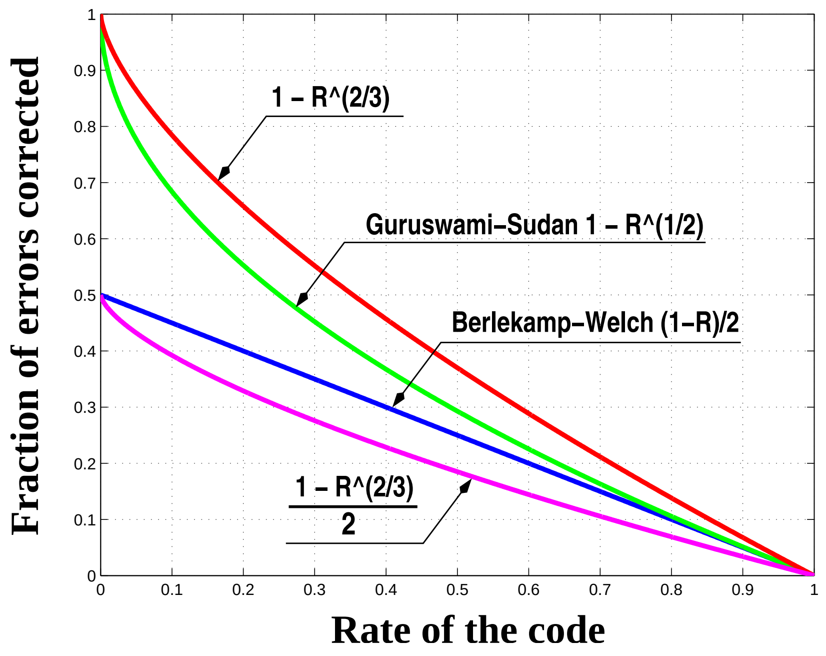


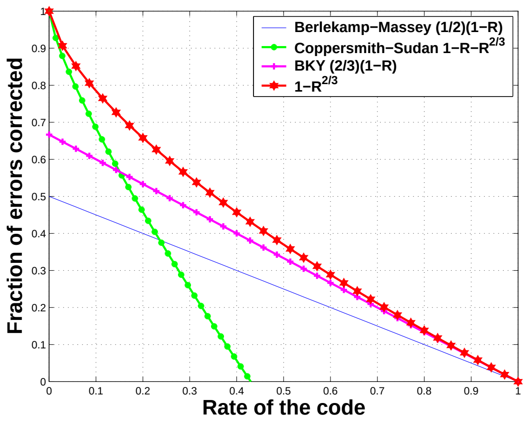
Figure 2.1: Error-correction radius of trivariate interpolation decoding

codewords and assign interpolation points in an M-dimensional affine space. It is shown in Section 2.3 that if all the errors are synchronized (occur in the same positions in all the $M-1$ codewords), then M-variate interpolation decoding can, in principle, correct up to $(M-1)n\big(1 - \sqrt[M]{R^{M-1}}\big)$ such errors.

Synchronized errors: prior work. As discussed in the next section, synchronized errors naturally occur in several distinct situations in coding theory [JTH04]. Decoding Reed-Solomon codes under the synchronized errors assumption is also equivalent to the *simultaneous polynomial reconstruction problem* in computer science [KY02]. This problem has been recently studied by Bleichenbacher, Kiayias, and Yung [BKY03] and by Coppersmith and Sudan [CS03]. BKY in [BKY03] develop an algorithm that, with a certain probability, corrects a fraction of errors given by

$$\tau_{\text{BKY}} \;=\; \frac{M-1}{M}\,\big(1 - R\big) \tag{2.2}$$

where $M-1$ is the number of Reed-Solomon codewords decoded together. The key idea of the BKY algorithm is to construct a single system of linear equations



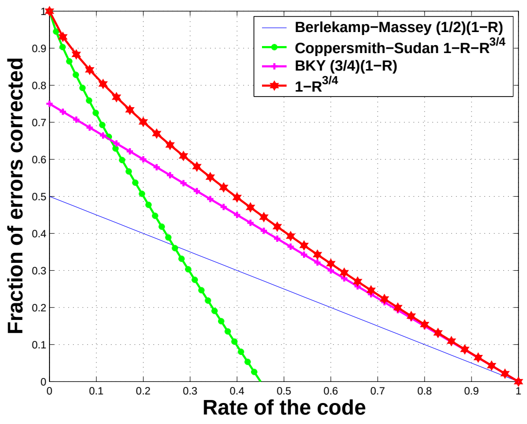


Figure 2.2: Comparison of decoding algorithms that correct synchronized errors, for $M = 3, 4$

for all the $M - 1$ codewords, which can be viewed as a natural generalization of the Berlekamp-Welch decoder [WB86]. Coppersmith and Sudan [CS03] develop a completely different probabilistic algorithm for the same problem, which corrects a fraction of errors given by

$$\tau_{\text{CS}} \; = \; 1 - R - R^{\frac{M-1}{M}} \tag{2.3}$$

This improves upon τ_{BKY} for rates that lie in the interval $[0, \gamma^M)$, where γ is the unique positive root of $\gamma^M + M\gamma^{M-1} = 1$. As discussed in the foregoing paragraph, multivariate interpolation decoding corrects a fraction of synchronized errors that is given by

$$\tau_{\text{MID}} \; = \; 1 - R^{\frac{M-1}{M}} \tag{2.4}$$

This improves substantially upon both τ_{BKY} and τ_{CS} for *all* rates, as illustrated in Figure 2.2. Moreover, as already mentioned, the algorithms of BKY and Coppersmith-Sudan are *probabilistic*. That is, both algorithms correct the fraction of errors in (2.2) and (2.3), respectively, only with a certain probability $1 - \text{P}_{\text{FAIL}}$. The alternative outcome, which occurs with probability P_{FAIL}, is decoding failure. Both BKY and Coppersmith-Sudan assume a memoryless q-ary symmetric channel model, where all the errors are independent and distributed uniformly at random over $\mathbb{F}_q$. For this model, BKY estimate the probability of failure as $\text{P}_{\text{FAIL}} \; = \; t/q$, where t is the number of synchronized errors. In view of (2.2), this behaves as $P_{\text{FAIL}} = O(n/q)$. This estimate was recently improved to $O(1/q)$ by Brown, Minder, and Shokrollahi [BMS05]. The probability of failure for the Coppersmith-Sudan decoder [CS03] is much higher. The best estimate given in [CS03] is $P_{\text{FAIL}} = O(n^{O(M)}/q)$. This is fine if the alphabet size q is very large compared to M and n. However, it is not the case in practice. Indeed, a failure probability of $1/q$ would be significantly *higher* than the error rates typically required in most applications of Reed-Solomon codes.

In contrast, our decoder is *deterministic*. It is widely believed [Gur01, JH01, RR03] that, at least in some cases, deterministic decoding in polynomial time is impossible, since there is an exponential number of codewords within a sphere or radius $n\tau_{\text{MID}}$. However, rather than failing, the proposed algorithm gradually reduces its decoding radius until the number of codewords within a certain decoding region becomes polynomially bounded. We show that the decoding radius never drops below the Guruswami-Sudan threshold of $n\big(1 - R^{1/2}\big)$. All this is explained in Section 2.5.

Organization. The rest of this Chapter is organized as follows. We start in the next section with some definitions and notation. The synchronized errors scenario is also discussed in more detail in the next section. Section 2.3 is concerned with the error-correction radius of multivariate interpolation decoding. We first derive the bound (2.1), and then generalize this bound to the case of M-variate interpolation. Sections 2.4 and 2.5 deal with the algebraic aspects of our decoder, namely the interpolation itself and the recovery of the transmitted codewords from the results of such interpolation. Although our algebraic methods can be easily extended to the general case of M-variate interpolation, we limit our discussion to the first nontrivial case $M = 3$. In Section 2.4, we show how to perform trivariate interpolation using a generalized version of Koetter's iterative interpolation algorithm [Köt96]. Notably, this algorithm produces not just one polynomial that satisfies the interpolation constraints, but a minimal Gröbner basis for the ideal of all such polynomials. In Section 2.5, we show how this Gröbner basis should be processed to recover the message polynomials $f(X)$ and $g(X)$. This recovery process is very different from the bivariate factorization procedure used in the Guruswami-Sudan decoder [GS99, RR00]. As we shall see, straightforward factorization of the interpolation polynomial $\mathcal{Q}(X,Y,Z)$ does not work and, moreover, the number of possible solutions to $\mathcal{Q}\big(X, f(X), g(X)\big) \equiv 0$ usually behaves as $\Omega(q^k)$. Thus the recovery of $f(X)$ and $g(X)$ requires at least *two* polynomials that satisfy the interpolation constraints. In fact, the recovery method developed in Section 2.5 is based upon factoring certain resultants of the (first two) elements of the Gröbner basis computed in Section 2.4. In Sections 2.6 we show that in simulation the algorithm basically correct up to $n(1-R^{2/3})$ fraction of errors when we are decoding to interleaved Reed-Solomon codes on a q-ary symmetric channel. Finally, in Section 2.7 we give analytical evidence that the decoder decodes with high probability up to $n\big(1-(6R)^{2/3}\big)$ when $M = 2$ and $m = 1$.

2.2 Preliminaries

In this section, we set up some of the notation that will be used throughout this work. We first define Reed-Solomon codes. Let q be a power of a prime, and let $\mathbb{F}_q$ be the finite field with q elements. We use $\mathbb{F}_q[X]$, $\mathbb{F}_q[X,Y]$, and $\mathbb{F}_q[X,Y,Z]$ to denote the rings of polynomials over $\mathbb{F}_q$ in one, two, and three variables, respectively. Reed-Solomon codes are obtained by evaluating certain subspaces of $\mathbb{F}_q[X]$ in a set of points $\mathcal{D} = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{F}_q$, which is known as the *locator set* (cf. [RR03]) or the *evaluation set* (cf. [GV05]) of the code. Specifically, a *Reed-Solomon code* $\mathbb{C}_q(n,k)$ of *length* n and *dimension* k is defined as follows:

$$\mathbb{C}_q(n,k) \overset{\text{def}}{=}$$
$$\big\{ \big(f(x_1), \ldots, f(x_n)\big) \;:\; x_1, x_2, \ldots, x_n \in \mathcal{D},\; f(X) \in \mathbb{F}_q[X],\; \deg f(X) < k \big\}$$

The *rate* R of $\mathbb{C}_q(n,k)$ is usually defined as k/n. However, we set $R \overset{\text{def}}{=} (k-1)/n$ as a matter of convenience. This doesn't change much, but simplifies many of the expressions in this paper.

Synchronized errors scenario. Let us begin with a precise definition. Consider M codewords $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_M$ of a Reed-Solomon code $\mathbb{C}_q(n,k)$ which correspond to evaluations of the polynomials $f_1, f_2, \ldots, f_M \in \mathbb{F}_q[X]$. Suppose that these M codewords are transmitted over a (hard-decision) channel and received as $\boldsymbol{c}_1 + \boldsymbol{e}_1, \boldsymbol{c}_2 + \boldsymbol{e}_2, \ldots, \boldsymbol{c}_M + \boldsymbol{e}_M$, where

$$\begin{aligned} \boldsymbol{e}_1 &= (e_{1,1}, e_{1,2}, \ldots, e_{1,n}) \\ \boldsymbol{e}_2 &= (e_{2,1}, e_{2,2}, \ldots, e_{2,n}) \\ &\vdots \\ \boldsymbol{e}_M &= (e_{M,1}, e_{M,2}, \ldots, e_{M,n}) \end{aligned}$$

are the error vectors. We say that the error pattern $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_M \in \mathbb{F}_q^n$ has at most t ***synchronized errors*** if there exists a set $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ of size $|\mathcal{J}| = t$ such that

$$e_{i,j} = 0 \qquad \text{for all } i \in \{1, 2, \ldots, M\} \text{ and } j \notin \mathcal{J}$$

Synchronized errors naturally occur when Reed-Solomon codes are *block interleaved* [BKY03, JH01]. That is, the codewords $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_M$ are viewed as rows of an $M \times n$ array $\mathcal{A}(f_1, f_2, \ldots, f_M)$ over $\mathbb{F}_q$. The elements of this array are then transmitted across a channel column-by-column. If the channel is bursty, the errors are likely to be synchronized. Block interleaving is a very common technique in digital communications [LM88], and whenever Reed-Solomon codes are used as outer codes in a concatenated coding scheme, the channel is likely to be bursty [JH01].

Here is another interesting interpretation of the synchronized errors scenario. One can regard the entire $M \times n$ array $\mathcal{A}(f_1, f_2, \ldots, f_M)$ as a *single codeword* of a code over $\mathbb{F}_Q$, where $Q = q^M$. Indeed, given any fixed basis for $\mathbb{F}_Q$ over $\mathbb{F}_q$, the n columns of the array $\mathcal{A}(f_1, f_2, \ldots, f_M)$ can be viewed as n elements of $\mathbb{F}_Q$. It is easy to see that the set of all such arrays forms a linear code of length n and dimension k over $\mathbb{F}_Q$, which we denote by $\mathscr{C}$. If codewords of $\mathscr{C}$ are now transmitted over a Q-ary channel, then all the errors are obviously synchronized. Curiously, the code $\mathscr{C}$ itself is again a *Reed-Solomon code* (and, hence, an MDS code [MS81]). It is a somewhat special Reed-Solomon code over $\mathbb{F}_Q$ in that its locator set $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ belongs to a subfield $\mathbb{F}_q$ of $\mathbb{F}_Q$. Conversely, *any* Reed-Solomon code $\mathbb{C}_q(n, k)$ whose locator set belongs to a subfield $\mathcal{F}$ of

$\mathbb{F}_q$, of index M, has the structure of a block interleaving of several codewords of a Reed-Solomon code of the same length and dimension, and the same locator set, but over $\mathcal{F}$. We leave the proof of all this as an exercise for the reader. As a practical example, we point out the $(16,k)$ Reed-Solomon codes over $\mathbb{F}_{256}$ used in the cdma2000 communication standard.

More definitions. We need some definitions pertaining to polynomials in one, two, and three variables over $\mathbb{F}_q$. Thus let $f(X) \in \mathbb{F}_q[X]$, $P(X,Y) \in \mathbb{F}_q[X,Y]$, and $Q(X,Y,Z) \in \mathbb{F}_q[X,Y,Z]$. Given nonnegative integers a, b, c, the corresponding *Hasse derivatives* are defined by

$$\mathscr{D}_a\big[f(X)\big] \overset{\text{def}}{=} \sum_{i=a}^{\infty} \binom{i}{a} f_i\, X^{i-a} \tag{2.5}$$

$$\mathscr{D}_{a,b}\big[P(X,Y)\big] \overset{\text{def}}{=} \sum_{i=a}^{\infty}\sum_{j=b}^{\infty} \binom{i}{a}\binom{j}{b} p_{i,j}\, X^{i-a}Y^{j-b} \tag{2.6}$$

$$\mathscr{D}_{a,b,c}\big[Q(X,Y,Z)\big] \overset{\text{def}}{=} \sum_{i=a}^{\infty}\sum_{j=b}^{\infty}\sum_{k=c}^{\infty} \binom{i}{a}\binom{j}{b}\binom{k}{c} q_{i,j,k}\, X^{i-a}Y^{j-b}Z^{k-c} \tag{2.7}$$

Note that all the sums in (2.5) – (2.7) are finite, since by the definition of a polynomial there is only a finite number of nonzero coefficients f_i, $p_{i,j}$, and $q_{i,j,k}$. Also note that for all $x \in \mathbb{F}_q$, the Hasse derivative of $f(X)$ evaluated at x, namely $\mathscr{D}_a[f(X)]\Big|_x$, is just the coefficient of X^a in the polynomial $f(X+x)$. Similarly, $\mathscr{D}_{a,b}[P(X,Y)]\Big|_{(x,y)}$ and $\mathscr{D}_{a,b,c}[Q(X,Y,Z)]\Big|_{(x,y,z)}$ give the coefficients of X^aY^b in $P(X+x, Y+y)$ and of $X^aY^bZ^c$ in $Q(X+x, Y+y, Z+z)$.

Definition 2.1. *Let $x_0 \in \mathbb{F}_q$, $(x_0,y_0) \in \mathbb{F}_q \times \mathbb{F}_q$, and $(x_0,y_0,z_0) \in \mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$. The polynomials $f(X)$, $P(X,Y)$, $Q(X,Y,Z)$ are said to pass through the points x_0, (x_0,y_0), and (x_0,y_0,z_0), respectively, with multiplicity m, or to have a zero of multiplicity m at these points, if*

$$\mathscr{D}_a\big[f(X)\big]\Big|_{x_0} = 0 \ \text{ for all } a \in \mathbb{N} \text{ with } a < m \tag{2.8}$$

$$\mathscr{D}_{a,b}\big[P(X,Y)\big]\Big|_{(x_0,y_0)} = 0 \ \text{ for all } a,b \in \mathbb{N} \text{ with } a+b < m \tag{2.9}$$

$$\mathscr{D}_{a,b,c}\big[Q(X,Y,Z)\big]\Big|_{(x_0,y_0,z_0)} = 0 \ \text{ for all } a,b,c \in \mathbb{N} \text{ with } a+b+c < m \tag{2.10}$$

We next deal with the *weighted degree*. Weighted degree of multivariate polynomials can be defined quite generally. However, the only weighted degrees we will need in this paper are the $(1, k{-}1)$-weighted degree for bivariate polynomials and the $(1, k{-}1, k{-}1)$-weighted degree for trivariate polynomials, where k is the dimension of the Reed-Solomon code at hand. Thus, henceforth, whenever we say "weighted degree" this is what we mean. The ***weighted degree*** of the monomials X^aY^b and $X^aY^bZ^c$ is defined by

$$\mathrm{w}\deg X^aY^b \overset{\text{def}}{=} a + (k{-}1)b, \quad \mathrm{w}\deg X^aY^bZ^c \overset{\text{def}}{=} a + (k{-}1)b + (k{-}1)c \qquad (2.11)$$

We can extend weighted degree to a monomial ordering $\prec_{\mathrm{w}}$ by augmenting it with the lex order. Explicitly, if the weighted degrees of two monomials are equal, we shall write $X^aY^b \prec_{\mathrm{w}} X^uY^v$ iff $a < u$, and $X^aY^bZ^c \prec_{\mathrm{w}} X^uY^vZ^w$ iff either $a < u$ or $a = u$ and $b < v$. Every polynomial in $\mathbb{F}_q[X, Y]$ and $\mathbb{F}_q[X, Y, Z]$ now has a well-defined leading monomial under $\prec_{\mathrm{w}}$, and we define the weighted degree of the polynomial as the weighted degree of its leading monomial.

These are all the definitions we will need (until Section 2.5). Observe that all these definitions generalize in the obvious way to polynomials in more than three variables. We do not spell out such generalizations, although they will be used in the next section.

2.3 Error-correction radius of the MID algorithm

Let $f(X)$ and $g(X)$ be arbitrary polynomials of degree $< k$ over $\mathbb{F}_q$. Suppose that the codewords $\big(f(x_1), f(x_2), \ldots, f(x_n)\big)$ and $\big(g(x_1), g(x_2), \ldots, g(x_n)\big)$ of a Reed-Solomon code $\mathbb{C}_q(n, k)$ were transmitted, and received as $(y_1, y_2, \ldots, y_n)$, $(z_1, z_2, \ldots, z_n) \in \mathbb{F}_q^n$, respectively. We set

$$\mathcal{P} \overset{\text{def}}{=} \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\} \qquad (2.12)$$

and define the ***interpolation polynomial*** $\mathcal{Q}(X, Y, Z)$ as the least weighted degree nonzero polynomial in $\mathbb{F}_q[X, Y, Z]$ that passes through each of the n points in $\mathcal{P}$ with multiplicity m. If there is more than one polynomial of the smallest weighted degree, ties can be broken using the monomial order $\prec_{\mathrm{w}}$ defined in the previous section (or in any other way). In the next section, we show how the interpolation polynomial $\mathcal{Q}(X,Y,Z)$ can be efficiently computed in polynomial time.

Lemma 2.1.

$$\mathrm{w}\deg \mathcal{Q}(X,Y,Z) \;\leqslant\; \left\lceil \sqrt[3]{n(k-1)^2 m(m+1)(m+2)} \right\rceil \tag{2.13}$$

Proof. The proof is a straightforward generalization of the argument used in [GS99] and [KV03a]. Let $N(\Delta)$ denote the number of trivariate monomials whose weighted degree is at most Δ. We claim that there exists a nonzero polynomial of weighted degree at most Δ that passes through each point in $\mathcal{P}$ with multiplicity m, provided

$$N(\Delta) \;>\; n\,\frac{m(m+1)(m+2)}{6} \tag{2.14}$$

Indeed, referring to (2.7), it is easy to see that (2.10) is just a system of linear constraints on the coefficients of such a polynomial. The total number of linear constraints imposed by passing with multiplicity m through each of n points is $n\,|\{(a,b,c)\in\mathbb{N}^3 \,:\, a+b+c \,<\, m\}|$, which evaluates to the right-hand side of (2.14). The total number of unknowns in the system (coefficients of the polynomial) is $N(\Delta)$. If the number of unknowns is strictly greater than the number of constraints, the linear system is guaranteed to have a nonzero solution.

It follows that any Δ such that $N(\Delta)$ satisfies (2.14) is an upper bound on the weighted degree of the interpolation polynomial $\mathcal{Q}(X,Y,Z)$. It remains to estimate $N(\Delta)$. To this end, consider the correspondence between monomials in $\mathbb{F}_q[X,Y,Z]$ and unit cubes in $\mathbb{R}^3$, given by

$$X^aY^bZ^c \;\mapsto\; K(a,b,c) \overset{\mathrm{def}}{=} \Big\{(x,y,z)\in\mathbb{R}^3 \,:\, a\leqslant x<a{+}1,\; b\leqslant y<b{+}1,\; c\leqslant z<c{+}1\Big\}$$

Such cubes are fundamental domains of the lattice $\mathbb{Z}^3$; hence, they do not intersect. Therefore, $N(\Delta)$ is equal to the *volume* of a union of such cubes $K(a,b,c)$,

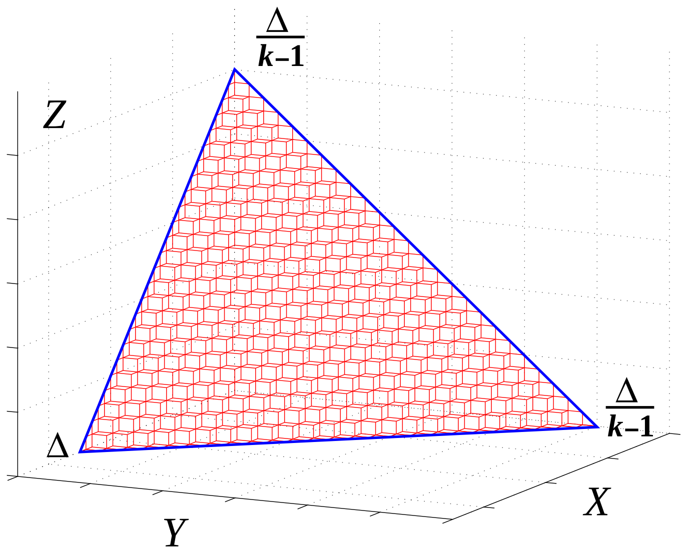


Figure 2.3: The union of cubes $\mathscr{S}$ and the pyramid $\mathscr{P}$

the union being taken over all $a, b, c \in \mathbb{N}$ that satisfy $a + (k-1)b + (k-1)c \leqslant \Delta$. Let $\mathscr{S} \subset \mathbb{R}^3$ denote this union of unit cubes (as illustrated in Figure 2.3). Now consider the pyramid $\mathscr{P} \subset \mathbb{R}^3$ defined by the four half-planes: $x \geqslant 0$, $y \geqslant 0$, $z \geqslant 0$, and $x + (k-1)y + (k-1)z \leqslant \Delta$. It is easy to see that $\mathscr{P} \subset \mathscr{S}$. Indeed, any given point (x, y, z) of $\mathscr{P}$ belongs to the unit cube $K(\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor)$, and the fact that $x + (k-1)y + (k-1)z \leqslant \Delta$ implies that $K(\lfloor x \rfloor, \lfloor y \rfloor, \lfloor z \rfloor)$ is one of the cubes in $\mathscr{S}$. Hence

$$N(\Delta) \;=\; \mathrm{Vol}(\mathscr{S}) \;>\; \mathrm{Vol}(\mathscr{P}) \;=\; \frac{\Delta^3}{6(k-1)^2} \tag{2.15}$$

Substituting the expression on the right-hand side of (2.13) for Δ in (2.15), we find that $N(\Delta)$ satisfies (2.14), and the lemma follows. ∎

Theorem 2.2. *Suppose that codewords $\boldsymbol{c}_1, \boldsymbol{c}_2$ of a Reed-Solomon code $\mathbb{C}_q(n, k)$, corresponding to evaluations of the polynomials $f(X)$, $g(X)$, are transmitted and received as $\boldsymbol{c}_1 + \boldsymbol{e}_1, \boldsymbol{c}_2 + \boldsymbol{e}_2$. If the error pattern $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{F}_q^n$ has at most t synchronized errors, where*

$$t \;\leqslant\; \left\lfloor n \;-\; n\sqrt[3]{R^2\left(1+\frac{1}{m}\right)\left(1+\frac{2}{m}\right)} \;-\; \frac{1}{m} \right\rfloor \tag{2.16}$$

then the interpolation polynomial $\mathcal{Q}(X,Y,Z)$ satisfies $\mathcal{Q}\big(X,f(X),g(X)\big) \equiv 0$. That is, the univariate polynomial $p(X) = \mathcal{Q}\big(X,f(X),g(X)\big)$ is the all-zero polynomial.

Proof. Since $f(X)$ and $g(X)$ are polynomials of degree at most $k-1$, it follows from the definition of weighted degree and Lemma 2.1 that

$$\deg p(X) \;\leqslant\; {}_{\mathbf{w}}\!\deg \mathcal{Q}(X,Y,Z) \;\leqslant\; \left\lceil \sqrt[3]{n(k{-}1)^2 m(m{+}1)(m{+}2)} \right\rceil \tag{2.17}$$

If $j \in \{1,2,\ldots,n\}$ is an error-free position then, by definition, $\mathcal{Q}(X,Y,Z)$ passes through the point $\big(x_j, f(x_j), g(x_j)\big)$ with multiplicity m. It now follows from (2.8) and (2.10) that the total number of zeros of $p(X)$ in $\mathbb{F}_q$ (counted with multiplicities) is at least

$$\#\,\text{zeros of } p(X) \;\geqslant\; m(n-t) \tag{2.18}$$

By the fundamental theorem of algebra, if the right-hand side of (2.18) is strictly greater than the right-hand side of (2.17), then $p(X)$ must be the all-zero polynomial. The former condition is satisfied provided the number of synchronized errors t is upper bounded as in (2.16). ∎

Clearly, the right-hand side of (2.16) tends to $\lceil n(1-R^{2/3})\rceil$ as $m \to \infty$. Thus $\mathcal{Q}(X,Y,Z)$ satisfies $\mathcal{Q}\big(X,f(X),g(X)\big) \equiv 0$ provided the fraction $\tau_{\text{3D}} = t/n$ of synchronized errors is at most

$$\tau_{\text{3D}} \;\leqslant\; 1 - R^{2/3} + o(1) \tag{2.19}$$

where $o(1)$ denotes a function that tends to zero when $m,n \to \infty$. This establishes (2.4) for the special case $M = 3$. We consider the general case of M-variate interpolation next.

The set-up for general M-variate interpolation decoding is as follows. Consider $M-1$ codewords $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{M-1}$ of a Reed-Solomon code $\mathbb{C}_q(n,k)$ which correspond to evaluations of the polynomials $f_1(X), f_2(X), \ldots, f_{M-1}(X)$ of degree $< k$ over $\mathbb{F}_q$. Suppose that these $M-1$ codewords are transmitted over a (hard-decision) channel and received as $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{M-1}$, where $\boldsymbol{y}_i = \boldsymbol{c}_i + \boldsymbol{e}_i = (y_{i,1}, y_{i,2}, \ldots, y_{i,n}) \in \mathbb{F}_q^n$ for $i = 1,2,\ldots,M-1$. As in (2.12), we will assign a set of n interpolation points

$$\mathcal{P} \overset{\text{def}}{=} \{(x_j, y_{1,j}, y_{2,j}, \ldots, y_{M-1,j}) \;:\; j = 1, 2, \ldots, n\} \tag{2.20}$$

and define the interpolation polynomial $\mathcal{Q}(X, Y_1, Y_2, Y_3, \ldots, Y_{M-1})$ as the least weighted degree nonzero polynomial in $\mathbb{F}_q[X, Y_1, Y_2, Y_3, \ldots, Y_{M-1}]$ that passes through each of the n points in $\mathcal{P}$ with multiplicity m. The argument used in the derivation of (2.19) now easily generalizes.

Lemma 2.3.

$$\mathrm{wdeg}\, \mathcal{Q}(X, Y_1, Y_2, \ldots, Y_{M-1}) \leqslant \left\lceil \sqrt[M]{n(k-1)^{M-1} m(m+1)\cdots(m+M-1)} \right\rceil \tag{2.21}$$

Proof. As in Lemma 2.1, let $N(\Delta)$ denote the number of M-variate monomials whose weighted degree is at most Δ. The number of linear constraints imposed by passing with multiplicity m through each of the n points in $\mathcal{P}$ is $n\,|\{(a_1, a_2, \ldots, a_M) \in \mathbb{N}^M : a_1 + a_2 + \cdots + a_M < m\}|$. It is well known that the number of partitions of a positive integer i into exactly M nonnegative integer parts is $\binom{M+i-1}{M-1}$. Thus there exists a nonzero polynomial of weighted degree at most Δ that passes through each point in $\mathcal{P}$ with multiplicity m, provided

$$N(\Delta) > n \sum_{i=0}^{m-1} \binom{M+i-1}{M-1} = n\, \frac{(M+m-1)!}{M!(m-1)!} \tag{2.22}$$

It remains to estimate $N(\Delta)$. As in Lemma 2.1, $N(\Delta)$ is strictly greater than the volume of the pyramid $\mathscr{P} \subset \mathbb{R}^M$ defined by the $M+1$ half-planes: $x \geqslant 0$, $y_1 \geqslant 0$, $y_2 \geqslant 0$, $\ldots$, $y_{M-1} \geqslant 0$, and $x + (k-1)(y_1 + y_2 + \cdots + y_{M-1}) \leqslant \Delta$. Thus

$$N(\Delta) > \mathrm{Vol}(\mathscr{P}) = \frac{\Delta^M}{M!(k-1)^{M-1}} \tag{2.23}$$

As in Lemma 2.1, substituting the expression on the right-hand side of (2.21) for Δ in (2.23), we find that $N(\Delta)$ satisfies (2.22), and the lemma follows. ▮

Theorem 2.4. *Suppose that codewords $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{M-1}$ of a Reed-Solomon code $\mathbb{C}_q(n,k)$, corresponding to evaluations of the polynomials $f_1(X), \ldots, f_{M-1}(X)$,* *are transmitted and received as* $\boldsymbol{c}_1 + \boldsymbol{e}_1, \boldsymbol{c}_2 + \boldsymbol{e}_2, \ldots, \boldsymbol{c}_{M-1} + \boldsymbol{e}_{M-1}$. *If the error pattern* $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_{M-1} \in \mathbb{F}_q^n$ *has at most* t *synchronized errors, where*

$$t \leqslant \left\lfloor n - n \sqrt[M]{R^{M-1}\left(1+\frac{1}{m}\right)\left(1+\frac{2}{m}\right)\cdots\left(1+\frac{M-1}{m}\right)} - \frac{1}{m} \right\rfloor \tag{2.24}$$

then the interpolation polynomial satisfies $\mathcal{Q}(X, f_1(X), f_2(X), \ldots, f_{M-1}(X)) \equiv 0$. *That is, the polynomial* $p(X) = \mathcal{Q}(X, f_1(X), f_2(X), \ldots, f_{M-1}(X))$ *is the all-zero polynomial.*

Proof. Since the degree of $p(X)$ does not exceed the weighted degree of $\mathcal{Q}$, it is bounded by the right-hand side of (2.21). The total number of zeros of $p(X)$ is at least $m(n-t)$, as before. The theorem now follows from the fundamental theorem of algebra. ▮

Remark. While the proofs of Theorems 2.2 and 2.4 are technically quite simple, it is perhaps not clear, on an intuitive level, why M-variate interpolation decoding has the potential to correct more errors. Qualitatively, the underlying reason is a dimensionality gain: the number of monomials of weighted degree at most Δ grows much faster in three dimensions than in two dimensions, faster yet in four dimensions, and so on.

2.4 Iterative multivariate interpolation

It should be obvious that the interpolation polynomial can be always computed in polynomial time by solving a system of linear equations over $\mathbb{F}_q$. However, straightforward Gaussian elimination and/or matrix inversion [ABKR00, BM82, MMM93] is not the most efficient way to accomplish this task.

In this section, we describe an efficient iterative algorithm for multivariate polynomial interpolation. The algorithm is an easy generalization of the procedure devised by Koetter [Köt96] for bivariate interpolation. Since the correctness of this procedure is by now well established [FG01, FO02, Köt96, NH98], we do not provide a proof of the correctness of our algorithm. Moreover, the algorithm is described in detail only for the special case of trivariate interpolation.

Further generalization to interpolation in M dimensions, for an arbitrary M, is briefly sketched out in the last paragraph.

Notably, the iterative interpolation algorithm presented in this section computes much more than just the interpolation polynomial $\mathcal{Q}(X,Y,Z)$. It produces a set of polynomials which forms a Gröbner basis for the ideal of all polynomials over $\mathbb{F}_q$ that satisfy the interpolation constraints. This property of the interpolation algorithm will be crucial in the next section.

The input to our algorithm consists of the set $\mathcal{P}$ of n points in $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$, as defined in (2.12), a positive integer m, which serves as the multiplicity parameter, and the weighted-degree monomial order $\prec_{\mathrm{w}}$ defined in Section 2.2. Let $\mathcal{I}_m(\mathcal{P})$ denote the ideal of all the trivariate polynomials over $\mathbb{F}_q$ that pass through each point in $\mathcal{P}$ with multiplicity m. Then the output of the interpolation algorithm is a Gröbner basis (with respect to $\prec_{\mathrm{w}}$) for the ideal $\mathcal{I}_m(\mathcal{P})$. As noted in Lemma 2.1, passing through each of the points in $\mathcal{P}$ with multiplicity m imposes a total of

$$C(m,n) \overset{\text{def}}{=} n\binom{m+2}{3} = n\,\frac{m(m+1)(m+2)}{6} \tag{2.25}$$

linear constraints on the coefficients of every polynomial in $\mathcal{I}_m(\mathcal{P})$. This number $C(m,n)$ was called the *interpolation cost* in [KV03a, PV03, PV04b]. Let

$$\mu(m,R) \overset{\text{def}}{=} \min\left\{\mu \in \mathbb{N} \,:\, (k-1)\binom{\mu+2}{3} > C(m,n)\right\} \tag{2.26}$$

and let $\ell = \mu(m,R)\big(\mu(m,R)+1\big)/2$. Then the Gröbner basis $\mathcal{G}$ we compute will have ℓ polynomials: $\mathcal{G} = \{G_1, G_2, \ldots, G_\ell\}$. The major difference between our iterative algorithm and that of Koetter [Köt96] is in the initialization step. We initialize as follows

$$\mathcal{G}^{(0)} \overset{\text{def}}{=} \left\{G_1^{(0)}, G_2^{(0)}, \ldots, G_\ell^{(0)}\right\} := \left\{Y^a Z^b \,:\, \forall\, a,b \in \mathbb{N} \text{ such that } a+b < \mu(m,R)\right\} \tag{2.27}$$

Upon initialization, the interpolation algorithm goes through $C(m,n)$ iterations, imposing each of the $C(m,n)$ linear constraints one-by-one. Thus, referring to (2.10), suppose that at iteration i of the algorithm, we are dealing with the

Hasse derivative $\mathscr{D}_{a,b,c}\big[\cdot\big]$ at the point $(x_s, y_s, z_s) \in \mathcal{P}$. Then, given the set $\mathcal{G}^{(i-1)} = \{G_1^{(i-1)}, G_2^{(i-1)}, \ldots, G_\ell^{(i-1)}\}$ computed after the first $i-1$ iterations, the i-th iteration consists of the following:

$\boxed{1}$ For all $j = 1, 2, \ldots, \ell$, compute $\delta_j := \mathscr{D}_{a,b,c}\Big[G_j^{(i-1)}(X, Y, Z)\Big]\Bigg|_{(x_s, y_s, z_s)}$, the ***discrepancy*** of $G_j^{(i-1)}(X, Y, Z)$. If $\delta_j = 0$ for all j, stop.

$\boxed{2}$ Among $\{G_1^{(i-1)}, G_2^{(i-1)}, \ldots, G_\ell^{(i-1)}\}$, find the least with respect to $\prec_{\rm w}$ polynomial such that its discrepancy is nonzero. Let $G_t^{(i-1)}(X, Y, Z)$ be this polynomial, so $\delta_t \neq 0$.

$\boxed{3}$ For all $j = 1, 2, \ldots, \ell$, except $j = t$, compute

$$G_j^{(i)}(X, Y, Z) \ := \ G_j^{(i-1)}(X, Y, Z) \ - \ \frac{\delta_j}{\delta_t}\, G_t^{(i-1)}(X, Y, Z)$$

Then update the pivot polynomial, $G_t^{(i)}(X, Y, Z) := (X - x_s)G_t^{(i-1)}(X, Y, Z)$.

We will refer to the procedure above as `UpdateBasis`$\big(\mathcal{G}, (x_s, y_s, z_s); a, b, c\big)$. This procedure is the computational core of the iterative trivariate interpolation algorithm; for completeness, we state the entire algorithm below.

Iterative interpolation algorithm

`Input:` A set $\mathcal{P}$ of n points in $\mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$, a positive integer m, and monomial order $\prec_{\rm w}$.

`Output:` A Gröbner basis, with respect to $\prec_{\rm w}$, for the ideal $\mathcal{I}_m(\mathcal{P})$.

Initialization step: Set $\mathcal{G} := \big\{Y^a Z^b \ : \ \forall\, a, b \in \mathbb{N} \text{ such that } a + b < \mu(m, R)\big\}$.

Iteration step: For all $(x, y, z) \in \mathcal{P}$, do the following: for $a := 0$ to $m-1$, then for $b := 0$ to $m-a-1$, then for $c := 0$ to $m-a-b-1$, do

`UpdateBasis`$\big(\mathcal{G}, (x, y, z); a, b, c\big)$.

Termination step: Return $\mathcal{G} = \{G_1, G_2, \ldots, G_\ell\}$.

We will not prove that the set $\mathcal{G} = \{G_1, G_2, \ldots, G_\ell\}$ returned by the iterative interpolation algorithm upon completion of the $C(m, n)$ iterations is a Gröbner basis for $\mathcal{I}_m(\mathcal{P})$ (a proof of this fact would be an easy generalization of the arguments in [FG01, NH98, McE03b]). We henceforth order the polynomials in $\mathcal{G}$ according to $\prec_{\rm w}$. Namely, we assume that

$$G_1(X,Y,Z) \prec_{\rm w} G_2(X,Y,Z) \prec_{\rm w} \cdots \prec_{\rm w} G_\ell(X,Y,Z)$$

Then $G_1(X,Y,Z)$ can be taken as the interpolation polynomial $\mathcal{Q}(X,Y,Z)$. Since $\mathcal{G}$ is a Gröbner basis, $G_1(X,Y,Z)$ is guaranteed to have the least weighted degree among all trivariate polynomials over $\mathbb{F}_q$ that pass through each point in $\mathcal{P}$ with multiplicity m.

Referring to (2.25) and (2.26), we see that $\mu(m,R) = O\big(m/\sqrt[3]{R}\big)$ so that the number ℓ of polynomials in $\mathcal{G}$ is $O\big(m^2/R^{2/3}\big)$. Further, each of the $C(m,n) = O(nm^3)$ iterations at the iteration step consists of manipulating ℓ trivariate polynomials, each having at most $C(m,n)+1$ coefficients. Thus the iterative interpolation algorithm takes $O(n^2m^8/R^{2/3})$ operations (additions and multiplications) in $\mathbb{F}_q$. We observe that, in principle, this complexity can be reduced significantly using the methods of [KV03b]. However, explaining how the coordinate-transformation technique of [KV03b] generalizes to multivariate interpolation is beyond the scope of this chapter.

Finally, we point out how the iterative algorithm described in this section generalizes to M-variate interpolation. Steps $\boxed{1}$, $\boxed{2}$, and $\boxed{3}$ of `UpdateBasis`$(\cdot)$ extend in the obvious way to M variables and remain virtually unchanged. All that changes is the initialization in (2.27). A general expression for the cost $C(m,n)$ is given by the right-hand side of (2.22). Thus we take $\mu(m,R)$ as the least integer μ such that $R\mu(\mu+1)\cdots(\mu+M-1) > m(m+1)\cdots(m+M-1)$. We then initialize the M-variate Gröbner basis $\mathcal{G}$ as follows:

$$\Big\{ Y_1^{a_1} Y_2^{a_2} \cdots Y_{M-1}^{a_{M-1}} \;: \\ \forall\,(a_1,a_2,\ldots,a_{M-1}) \in \mathbb{N}^{M-1} \text{ with } a_1 + a_2 \cdots + a_{M-1} < \mu(m,R) \Big\}$$

It is easy to see that this set contains $\ell = O\big(m^{M-1}/R^{\frac{M-1}{M}}\big)$ polynomials, so that the complexity of the resulting M-variate interpolation algorithm is $\ell\, O(n^2m^{2M})$ or it is equal to $O\big(n^2m^{3M-1}/R^{\frac{M-1}{M}}\big)$.

2.5 Multivariate polynomial recovery

As seen in the previous section, multivariate polynomial interpolation is very similar to bivariate interpolation, and essentially the same iterative procedure can be used to accomplish this task. In contrast, trivariate "factorization" — namely, the task of recovering the polynomials $f(X)$ and $g(X)$ from the output of the interpolation step — is completely unlike the bivariate factorization task of the Guruswami-Sudan decoder [GS99, RR00] and all known versions thereof [KV03b, McE03b].

In fact, the term "factorization" itself becomes a misnomer in the trivariate case. Given a bivariate polynomial $\mathcal{Q}(X,Y)$, it is easy to see that $\mathcal{Q}\big(X,f(X)\big) \equiv 0$ if and only if $Y - f(X)$ is a factor of $\mathcal{Q}(X,Y)$. Thus complete factorization of $\mathcal{Q}(X,Y)$ immediately produces all the solutions $f(X)$ to $\mathcal{Q}\big(X,f(X)\big) \equiv 0$ and the number of possible solutions (whether of degree $< k$ or not) is obviously bounded by the Y-degree of $\mathcal{Q}(X,Y)$. In contrast, in the trivariate case $\mathcal{Q}\big(X,f(X),g(X)\big) \equiv 0$ if and only if $\mathcal{Q}(X,Y,Z)$ is of the form

$$\mathcal{Q}(X,Y,Z) \;=\; A(X,Y,Z)\big(Y - f(X)\big) \;+\; B(X,Y,Z)\big(Z - g(X)\big) \qquad (2.28)$$

where $A(X,Y,Z)$, $B(X,Y,Z)$ are arbitrary polynomials in $\mathbb{F}_q[X,Y,Z]$. It follows from (2.28) that neither $Y - f(X)$ nor $Z - g(X)$ are necessarily factors of $\mathcal{Q}(X,Y,Z)$, so that straightforward factorization does not work. We therefore use the term *recovery* rather than factorization.

More importantly, in the trivariate case, the number of pairs $\{f(X),g(X)\}$ such that $\mathcal{Q}\big(X,f(X),g(X)\big) \equiv 0$ is **not** bounded by the degree of $\mathcal{Q}(X,Y,Z)$. In fact, in some situations there are *exponentially many* solutions to the equation $\mathcal{Q}(X,f(X),g(X)) \equiv 0$, all of degree $< k$.

Decoding using two interpolation polynomials: As we mentioned, there are situations that recovery from only the interpolation polynomial leads us to exponentially many solutions. To reduce the size of the solution we can look at $f(X)$ and $g(X)$'s that not only satisfy $Q(X,f(X),g(X)) \equiv 0$ but also satisfy $P(X,f(X),g(X)) \equiv 0$ for some other polynomial in $\mathcal{I}_m(\mathcal{P})$. The intersection of solutions for both P and Q, in principle, is a smaller set than the solutions to $Q(X, f(X), g(X)) \equiv 0$ itself and hopefully we get a set of solutions that is not exponentially large. The best choice for P is the second smallest Gröbner basis of $\mathcal{I}_m(\mathcal{P})$, because it has the least weighted degree after $Q(X, Y, Z)$. Having two polynomials in hand, we can eliminate variables Z between them, in certain conditions, and get a bivariate polynomial in X and Y. That reduces the recovery problem back into bivariate case.

Lemma 2.5. *Let $H(X, Y) = \mathrm{Res}(G_1(X, Y, Z), G_2(X, Y, Z); Z)$ denotes the resultant of G_1 and G_2 respect to Z then $H(X, Y)$ belongs to the ideal of $\langle G_1, G_2 \rangle$, and so for any $(f(X), g(X))$ that both $G_1(X, f(X), g(X))$ and $G_2(X, f(X), g(X))$ are equivalent to zero then $H(X, f(X))$ is also equivalent to zero.*

To recover $f(X)$ and $g(X)$ using both G_1 and G_2 we proceed as follow:

$\boxed{\textbf{1}}$ Compute $H(X, Y) = \mathrm{Res}(G_1, G_2; Z)$. [*polynomial-time computation*]

$\boxed{\textbf{2}}$ Factor $H(X, Y)$ – using e.g., Roth-Ruckenstein algorithm – to recover $f(X)$ such that $H(X, f(X)) \equiv 0$. [# of factors is bounded by $\deg_Y H$]

$\boxed{\textbf{3}}$ Substitute Y by $f(X)$ in $G_1(X, Y, Z)$; then factor the resulting bivariate polynomial to recover $g(X)$. [# of factors is bounded by $\deg_Z G_1$]

To find a bound on the performance of the algorithm, we drive an upper bound on the weighted-degree of $G_2(X, Y, Z)$. For that we use properties of the *deltaset* of $\mathcal{I}_m(\mathcal{P})$.

Definition 2.2. *The deltaset $\Delta(\mathcal{I})$ of a polynomial ideal is defined as the set of all monomials that are <u>not</u> the leading monomials of the polynomials in $\mathcal{I}$.*

Lemma 2.6. *Let $\mathcal{I}_m(\mathcal{P})$ denote the ideal consisting of all polynomials belong to $\mathbb{F}_q[X, Y, Z]$ that pass through n given points in a set $\mathcal{P}$, each with multiplicity m, then $|\Delta(\mathcal{I}_m(\mathcal{P}))| = nm(m + 1)(m + 2)/6$. see [MTV04].*

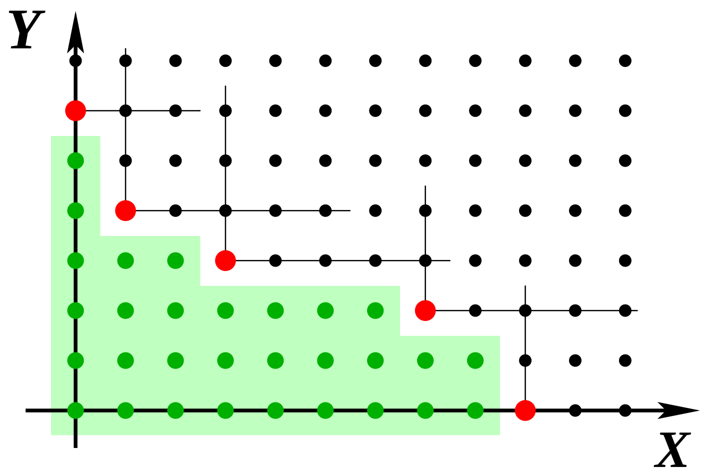


Figure 2.4: Deltaset of the ideal $\mathcal{I}_{\mathcal{P}}$

Lemma 2.7. *For the ideal $\mathcal{I}_m(\mathcal{P})$ assuming Δ_1 and Δ_2 are weighted-degrees of $G_1(X,Y,Z)$ and $G_2(X,Y,Z)$ then*

$$|\Delta(\mathcal{I}_m(\mathcal{P}))| \geqslant \frac{\Delta_2^3 - (\Delta_2 - \Delta_1)^3}{6(k-1)^2} \tag{2.29}$$

Proof. From Lemma 2.1 we know that number of monomials with weighted-degree smaller than Δ is approximately equal to $\Delta^3/6(k-1)^2$. So, number of monomials with weighted-degree smaller than Δ_2 in $\Delta(\mathcal{I}_m(\mathcal{P}))$ is approximately equal to number of monomials with weighted-degree smaller than Δ_2 minus the monomials that G_1 carves out the deltaset, see Figure 2.5. ▮

Theorem 2.8. *Suppose that codewords $\boldsymbol{c}_1, \boldsymbol{c}_2$ of Reed-Solomon code $\mathbb{C}_q(n,k)$, corresponding to evaluations of the polynomials $f(X)$ and $g(X)$, are transmitted and received as $\boldsymbol{c}_1 + \boldsymbol{e}_1$, $\boldsymbol{c}_2 + \boldsymbol{e}_2$. If the error pattern $\boldsymbol{e}_1, \boldsymbol{e}_2$ has at most t synchronized errors, where*

$$t \leqslant \left\lfloor n - n\frac{R}{2}\left(1 + \sqrt{\frac{4}{3R} - \frac{1}{3}}\right)\right\rfloor \tag{2.30}$$

then both $G_1(X, f(X), g(X))$ and $G_2(X, f(X), g(X))$ are equivalent to zero when multiplicity m goes to infinity.

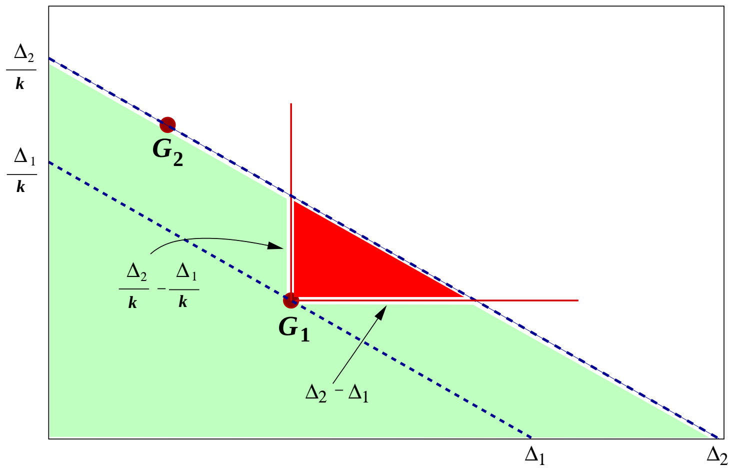


Figure 2.5: Deriving the bound for Δ_2

Proof. By using Lemma 2.6 and Lemma 2.7 we can show the correctness of (2.30). Due to lack of space we omit the proof. ∎

Whenever the resultants of G_1 and G_2 respect to Y and Z are nonzero polynomials, the recovery algorithm works but if G_1 and G_2 has a common factor the resultants become zero.

Lemma 2.9. *The resultant respect to Z, $\mathrm{Res}(G_1(X,Y,Z), G_2(X,Y,Z); Z)$ is the all-zero polynomial if and only if $G_1(X,Y,Z)$ and $G_2(X,Y,Z)$ have a common factor in $\mathbb{F}_q[X,Y,Z]$ which has a positive degree in Z. see [CLO96]*

Let $\Psi(X,Y,Z)$ denotes the $\gcd(G_1(X,Y,Z), G_2(X,Y,Z))$ and $U = G_1/\Psi, V = G_2/\Psi$. For any $f(X)$ and $g(X)$ that $G_1(X, f(X), g(X))$ and $G_2(X, f(X), g(X))$ are equivalent to zero we know either $\Psi(X, f(X), g(X))$ is equivalent to zero or both of $U(X, f(X), g(X))$ and $V(X, f(X), g(X))$ are equivalent to zero. U and V are relatively prime so the algorithm discussed can be used to recover $f(X)$ and $g(X)$ from them but for $f(X)$ and $g(X)$'s that are solution of $\Psi(X, f(X), g(X)) \equiv 0$ we use next Gröbner basis, $G_3(X,Y,Z)$, and $\Psi(X,Y,Z)$ together for recovery. Hence in general we have the following adaptive recovery algorithm:

Adaptive recovery algorithm:

Initialize by setting $Q(X,Y,Z) := G_1(X,Y,Z)$ and $P(X,Y,Z) := G_2(X,Y,Z)$.

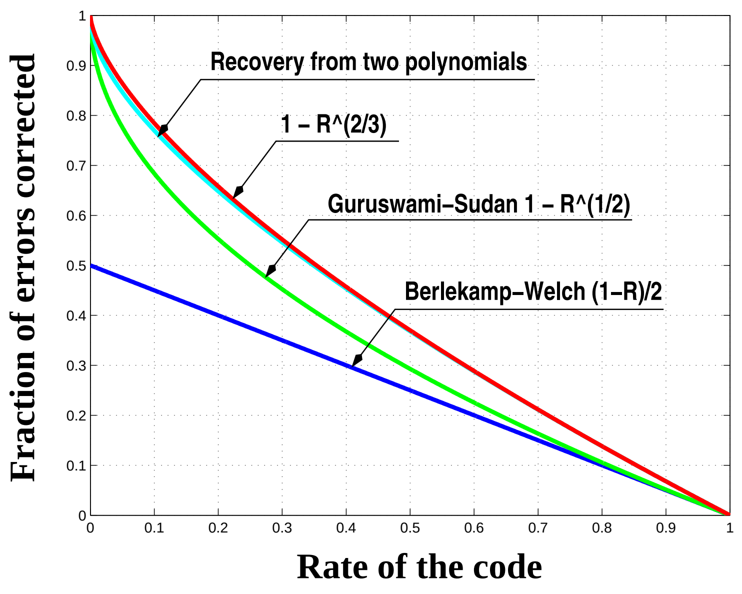


Figure 2.6: Bound on decoding with first two Gröbner basis

Also set $i := 2$, and proceed as follow.

$\boxed{1}$ Compute $\Psi(X,Y,Z) := \gcd(Q,P)$, using the Euclidean algorithm.

$\boxed{2}$ If $\Psi(X,Y,Z)$ is a polynomial in X and not in Y and Z, recover $f(X)$ by factoring $\mathrm{Res}(Q,P;Z)$ and recover $g(X)$ by factoring $\mathrm{Res}(Q,P;Y)$ with the Roth-Ruckenstein algorithm. **Stop.**

$\boxed{3}$ Otherwise, Let $U = Q/\Psi$ and $V = P/\Psi$. Then U and V are relatively prime. Recover $f(X)$ by factoring $\mathrm{Res}(U,V;Z)$ and $g(X)$ by factoring $\mathrm{Res}(U,V;Y)$.

$\boxed{4}$ Set $Q := \Psi(X,Y,Z)$, $P := G_{i+1}(X,Y,Z)$, and $i := i+1$. Go back to $\boxed{1}$

Theorem 2.10. *The adaptive recovery algorithm always terminates in polynomial time. It corrects a fraction τ_{3D} of synchronized errors, where*

$$1-\sqrt{R} \leqslant \tau_{3D} \leqslant 1-\frac{R}{2}\left(1+\sqrt{\frac{4}{3R}-\frac{1}{3}}\right) \simeq 1-R^{2/3} \qquad (2.31)$$

when m and n goes to infinity.

Proof. The maximum number of execution of the algorithm is upper bounded by the number of Gröbner basis which is polynomially bounded and all the steps in the algorithm are polynomial time so the algorithm runs in polynomial time. The upper bound of (2.31) is what we exactly drive previously. The sketch of proof for the lower bound is as follow. Assume $\{(x_1, y_1, z_1), \cdots, (x_n, y_n, z_n)\}$ are the received points. Let $P(X, Y)$ and $Q(X, Z)$ to be the least weighted-degree interpolation polynomials that passes through $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and $\{(x_1, z_1), \cdots, (x_n, z_n)\}$ respectively. From [GS99] we know weighted-degree of both P and Q are smaller than $mn\sqrt{R}$ for large m and n. Also, P and Q are in $\mathcal{I}_m(\mathcal{P})$ so P and Q are linear combination of first v and w Gröbner basis respectively. In addition, we know $G_i < \max\{P, Q\}$ for $i = 1, \cdots, \max\{v, u\}$ that mean the weighted-degree of all G_i, $i = 1, \cdots, \max\{v, u\}$ is less than or equal to maximum of weighted-degrees of P and Q which is smaller than $mn\sqrt{R}$. Finally, $\mu = \gcd(G_1, \cdots, G_{\max\{v,u\}})$ is in $\mathbb{F}_q[X]$ because, μ should divides the $\gcd(P, Q)$ and we know $\gcd(P, Q) \in \mathbb{F}_q$. Therefore, the adaptive algorithm runs at most up to $\max\{v, u\}$ iterations and uses at most $G_{\max\{u,v\}}$ for recovery. As we shown, weighted-degree of $G_{\max\{u,v\}}$ is less than or equal to $mn\sqrt{R}$ for large m and n. So if the fraction of synchronized errors is less than or equal to $1 - \sqrt{R}$ then we have $G_i(X, f(X), g(X)) \equiv 0$ for $i = 1, \cdots, \max\{u, v\}$ where $f(X)$ and $g(X)$ are the corresponding evaluation polynomials for the transmitted codeword. So the adaptive algorithm is able to recover $f(X)$ and $g(X)$ successfully. ∎

2.6 The MID algorithm works

So far we have shown that the multivariate interpolation algorithm can correct some error patterns when number of errors is greater than $n(1 - \sqrt{R})$ and smaller than $n(1 - R^{M/(M+1)})$. The problem is that successful decoding is *not guaranteed:* there are certain patterns of less than $n\tau_{\text{MID}}$ errors which the MID algorithm fails to decode. Nevertheless, simulations show that the actual performance of the MID decoder is very close to what one would expect if *all* patterns of up to $n\tau_{\text{MID}}$ errors were corrected. On the other hand, analysis of the

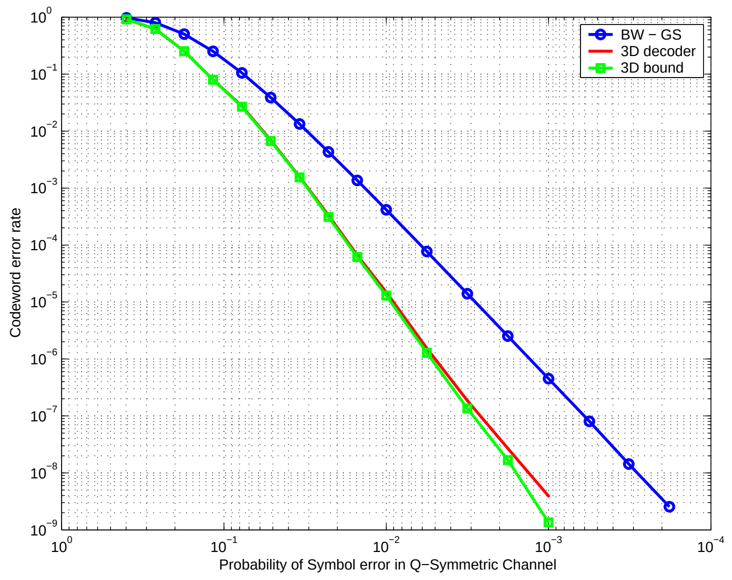


Figure 2.7: Performance of MID algorithm on decoding of $\mathrm{RS}(15, 11)$ over $\mathbb{F}_{256}$

failure probability for the MID algorithm is extremely difficult, and there were no analytic results so far to confirm this empirically observed behavior.

In the following, we first look at the simulation results for the MID algorithm. We consider a q-ary symmetric channel for the communication and in Figures 2.7 and 2.8 result of simulation is plotted. From the simulation results we see that in practice the MID algorithm can correct up to $n\tau_M$ many errors.

In Section 2.7, we provide an analytic results: we present a detailed analysis of the probability of failure in the MID algorithm for the special case where $M = 2$ and the interpolation multiplicity is $m = 1$. In this case, the MID algorithm attempts to correct up to $n\tau_{2,1}$ errors, where $\tau_{2,1} = 1 - \sqrt[3]{6R^2}$. We consider the situation where symbol values received from the channel at the erroneous positions are distributed uniformly at random (a version of the q-ary symmetric channel). We show that, with high probability, the performance of the MID algorithm is very close to the optimum in this case. Specifically, we prove that if the fraction of positions in error is at most $\tau_{2,1} - O(R^{5/3})$, then the probability of failure in the MID algorithm is at most $n^{-\Omega(n)}$. Thus the probability of failure is, indeed, negligible for large n in this case.

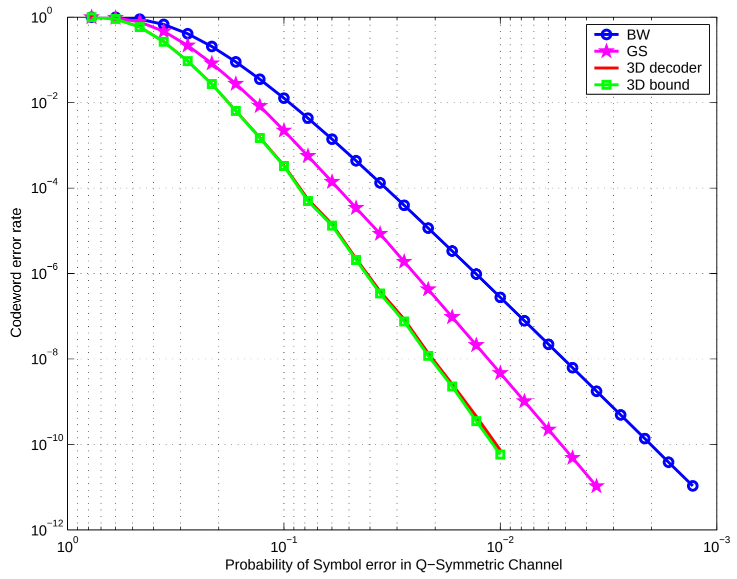


Figure 2.8: Performance of MID algorithm on decoding of RS$(15,7)$ over $\mathbb{F}_{256}$

2.7 A bound on the performance of the multivariate interpolation decoding algorithm

In this section, we study the failure probability of MID algorithm for a relatively simple special case where $M = 2$ and the interpolation multiplicity is $m = 1$ (as opposed to $m \to \infty$, which is needed to achieve the error-correction radii τ_{CS} and τ_{MID}). In this case, the MID algorithm attempts to correct at most $n\tau_{2,1}$ errors, where $\tau_{2,1} = 1 - \sqrt[3]{6R^2}$. We consider the situation where symbol values received from the channel at the erroneous positions are distributed uniformly at random and show that, with high probability, the performance of the MID algorithm is very close to the optimum in this case. Our main result is the following theorem.

Theorem 2.11. *Let $\mathbb{C}$ be an (n,k,d) Reed-Solomon code over the field of order $Q = q^2$ obtained by evaluating polynomials of degree $< k$ over $\mathbb{F}_Q$ in a set of points $\{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{F}_q$. Further assume that*

$$k - 1 = \frac{6(n+1)}{s(s+1)(s+2)} - \frac{3}{s} \tag{2.32}$$

for an integer $s \geqslant 1$. Suppose that a codeword $\boldsymbol{c}$ of $\mathbb{C}$ is transmitted over a Q-ary symmetric channel and the number of channel errors satisfies

$$t \leqslant n\left(1 - \sqrt[3]{6R^2} - O\big(R^{5/3}\big)\right) \tag{2.33}$$

Then the multivariate interpolation decoder recovers $\boldsymbol{c}$ from the channel output with probability at least $1 - n^{-\Omega(n)}$.

Theorem 2.11 shows that the probability of failure in the MID algorithm is, indeed, negligible for large n in this special case.

2.7.1 Background and notation

Let $\mathbb{F}_q$ denote the finite field with q elements, let $Q = q^2$, and let $\{1, \beta\}$ be a fixed basis for $\mathbb{F}_Q$ over $\mathbb{F}_q$. We will consider two Reed-Solomon codes $C_q(n,k)$ and $\mathbb{C}_Q(n,k)$, both obtained by evaluating polynomials of degree $\leqslant k-1$ in the same set of points $\mathcal{D} = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{F}_q$. Throughout this section, we assume that n, k are both $\theta(q)$, namely $k = Rn$ and $n = \eta q$, where R and η are constants. Specifically, the codes $C_q(n,k)$ and $\mathbb{C}_Q(n,k)$ are defined as follows:

$$C_q(n,k) \stackrel{\text{def}}{=} \Big\{ \big(f(x_1), \ldots, f(x_n)\big) \;:\; f \in \mathbb{F}_q[X],\, \deg f < k \Big\}$$
$$\mathbb{C}_Q(n,k) \stackrel{\text{def}}{=} \Big\{ \big(c_1 + \beta c'_1, \ldots, c_n + \beta c'_n\big) \;:\; \boldsymbol{c}, \boldsymbol{c}' \in C_q(n,k) \Big\}$$

We shall, moreover, assume that k satisfies (2.32). We point out that this limits the validity of our results to only certain rates.

Suppose that a codeword $\boldsymbol{c}$ of $\mathbb{C}_Q(n,k)$ is transmitted over a noisy channel and the vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{F}_Q^n$ is received at the channel output. We write $v_j = y_j + \beta z_j$ with y_j and z_j in $\mathbb{F}_q$ for all $j = 1, 2, \ldots, n$, and define

$$\mathcal{P}_{\boldsymbol{v}} \stackrel{\text{def}}{=} \{\, (x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n) \,\} \tag{2.34}$$

We let $\mathcal{I}(\mathcal{P}_{\boldsymbol{v}}) \subset \mathbb{F}_q[X, Y, Z]$ denote the ideal of polynomials over $\mathbb{F}_q$ that pass through all the points in $\mathcal{P}_{\boldsymbol{v}}$, namely

$$\mathcal{I}(\mathcal{P}_{\boldsymbol{v}}) \stackrel{\text{def}}{=} \Big\{ P \;:\; P(x_j, y_j, z_j) = 0 \;\; \forall\, (x_j, y_j, z_j) \in \mathcal{P}_{\boldsymbol{v}} \Big\} \tag{2.35}$$

We denote the minimal Gröbner basis, with respect to the $\prec_{\mathrm{w}}$ order, for the ideal $\mathcal{I}(\mathcal{P}_{\boldsymbol{v}})$ by

$$\mathcal{G}(\mathcal{P}_{\boldsymbol{v}}) \overset{\text{def}}{=} \Big\{G_1(X,Y,Z),\, G_2(X,Y,Z), \ldots, G_\ell(X,Y,Z)\Big\}$$

where $G_1 \prec_{\mathrm{w}} G_2 \prec_{\mathrm{w}} \cdots \prec_{\mathrm{w}} G_\ell$. The following theorem is one of the main results of Section 2.5:

Theorem 2.12. *Let $t = d(\boldsymbol{c},\boldsymbol{v})$ be the number of positions in $\boldsymbol{v}$ that are in error. Then multivariate interpolation decoding successfully recovers the transmitted codeword $\boldsymbol{c}$ provided*

$$\mathrm{GCD}\Big(\big\{G \in \mathcal{G}(\mathcal{P}_{\boldsymbol{v}}) \,:\, \mathrm{w}\!\deg G < n - t\big\}\Big) \in \mathbb{F}_q[X] \tag{2.36}$$

Herein, we will *not* be concerned with the details of the MID algorithm; rather, we'll use the sufficient condition (2.36) as our guiding principle. It is clear from (2.36) that our task involves careful estimation of the weighted degrees of the polynomials in $\mathcal{G}(\mathcal{P}_{\boldsymbol{v}})$. To this end, the following function will be useful

$$N(\Delta) \overset{\text{def}}{=} \Big|\big\{\, X^aY^bZ^c \,:\, \mathrm{w}\!\deg X^aY^bZ^c \leqslant \Delta \,\big\}\Big| \tag{2.37}$$

Observe that

$$\frac{\Delta^3}{6(k-1)^2} \;<\; N(\Delta) \;\leqslant\; \frac{(\Delta+k)^3}{6(k-1)^2} \tag{2.38}$$

as shown in [GS99, PV04a, PV05] and Section 2.2. However, we will need to be more precise. Specifically, we will use the fact that

$$N\big(s(k-1)\big) \;=\; {}^{1}\!/_{6}\,(s+1)(s+2)\big(s(k-1)+3\big) \tag{2.39}$$

for all positive integers s (this fact is the source for the condition (2.32) on k and n in Theorem 2.11). The correctness of (2.39) can be established by straightforward enumeration.

2.7.2 Bound on the failure probability

Before diving into the technical details, we first describe the general strategy of our proof. The key idea is to show that, in most cases, the minimal element of $\mathcal{G}(\mathcal{P}_{\boldsymbol{v}})$ — namely, the polynomial $G_1(X,Y,Z)$ — will be irreducible. More precisely, we will show that $G_1(X,Y,Z) = p(X)G^*(X,Y,Z)$, for an irreducible polynomial $G^*(X,Y,Z)$. Loosely speaking, to prove that $G_1(X,Y,Z)$ is of this form, we will establish an upper bound on ${}_{\mathbf{w}}\deg G_1$ (Lemma 2.13) as well as a lower bound on ${}_{\mathbf{w}}\deg G_1$ which holds with high probability (Lemma 2.14). Next we show in Lemma 2.15 that if G_1 is not of the desired form, then the lower bound exceeds the upper bound — a contradiction. Now, if $G_1(X,Y,Z) = p(X)G^*(X,Y,Z)$, where $G^*(X,Y,Z)$ is irreducible, then Theorem 2.12 implies that the MID algorithm will correct t errors as long as $n-t$ is greater than ${}_{\mathbf{w}}\deg G$, where G is any element of $\mathcal{G}(\mathcal{P}_{\boldsymbol{v}})$ that does not have G^* as its factor. Finally, using the fact that the size of the delta-set of the ideal $\mathcal{I}(\mathcal{P}_{\boldsymbol{v}})$ is n, we derive an upper bound on the weighted degree of G, and Theorem 2.11 follows.

Lemma 2.13.

$$ {}_{\mathbf{w}}\deg G_1 \leqslant s(k-1) $$

Proof. Let $P \in \mathbb{F}_q[X,Y,Z]$. Then $P \in \mathcal{I}(\mathcal{P}_{\boldsymbol{v}})$ if and only if the coefficients of P satisfy n linear equations — one for each point in $\mathcal{P}_{\boldsymbol{v}}$. Thus we can think of the coefficients of P as unknowns in a system of n linear equations. If P has at least $n+1$ nonzero coefficients, then the system is guaranteed to have a nonzero solution. It follows that $\mathcal{I}(\mathcal{P}_{\boldsymbol{v}})$ contains a polynomial of weighted degree Δ provided $N(\Delta) \geqslant n+1$. But $N\big(s(k-1)\big) = n+1$ by (2.32) and (2.39), and so $\mathcal{I}(\mathcal{P}_{\boldsymbol{v}})$ contains a polynomial of weighted degree $s(k-1)$. Since G_1 is the minimal element of $\mathcal{G}(\mathcal{P}_{\boldsymbol{v}})$, it has the smallest weighted degree among *all* polynomials in $\mathcal{I}(\mathcal{P}_{\boldsymbol{v}})$, and the lemma follows. ▮

Let us fix a set $\mathcal{E} \overset{\text{def}}{=} \{x_1, x_2, \ldots, x_t\} \subset \mathcal{D}$, where $\mathcal{D} \subseteq \mathbb{F}_q$ is the evaluation set for both $C_q(n,k)$ and $\mathbb{C}_Q(n,k)$. Now let $y_1, y_2, \ldots, y_t$, $z_1, z_2, \ldots, z_t$ be i.i.d. random variables distributed uniformly over $\mathbb{F}_q$, and define

$$ \mathcal{P} \overset{\text{def}}{=} \{\, (x_1,y_1,z_1), (x_2,y_2,z_2), \ldots, (x_t,y_t,z_t) \,\} \tag{2.40} $$

as in (2.34). As in (2.35), let $\mathcal{I}(\mathcal{P}) \subset \mathbb{F}_q[X,Y,Z]$ be the ideal of polynomials that pass through all the points in $\mathcal{P}$. Pick any polynomial $Q(X,Y,Z)$ in $\mathcal{I}(\mathcal{P})$. We

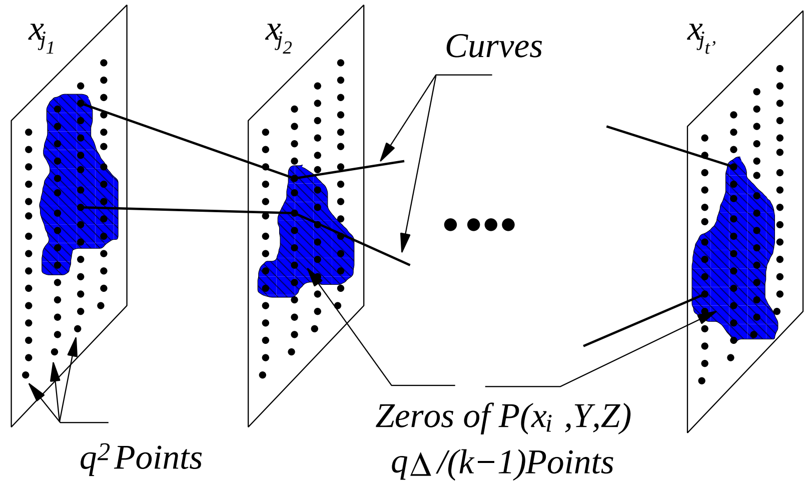


Figure 2.9: Counting the number of curves

will study ${}_{\mathbf{w}}\deg \mathcal{Q}$, which is a random variable. First, observe that

$$ {}_{\mathbf{w}}\deg \mathcal{Q} \geqslant \min\{t, k-1\} \tag{2.41}$$

Indeed, if $\mathcal{Q}$ has positive degree in either Y or Z, then we have ${}_{\mathbf{w}}\deg \mathcal{Q} \geqslant k-1$ by (2.11). Otherwise, $\mathcal{Q} \in \mathbb{F}_q[X]$ and it must be divisible by $\prod_{x_i \in \mathcal{E}}(X - x_i)$, in which case ${}_{\mathbf{w}}\deg \mathcal{Q} \geqslant t$.

Lemma 2.14. *For all $\varepsilon > 0$, with probability at least $1 - n^{-\Omega(n)}$ over the choice of $y_1, y_2, \ldots, y_t$ and $z_1, z_2, \ldots, z_t$, we have*

$$ {}_{\mathbf{w}}\deg \mathcal{Q} \geqslant \min\left\{t-k, \sqrt[3]{6t(k-1)^2} - k\right\} - \varepsilon n \tag{2.42}$$

Proof. Let $A(t,\varepsilon)$ be the event in (2.42), and let $\bar{A}(t,\varepsilon)$ be its complement. We need to show that $\Pr\{\bar{A}(t,\varepsilon)\} \leqslant n^{-\Omega(n)}$.

We say that a point $x \in \mathcal{E}$ is an *X-zero* of $\mathcal{Q}$ if $(X - x)$ is a factor of $\mathcal{Q}$. For each subset $\mathcal{Z} \subseteq \mathcal{E}$, we let $E_{\mathcal{Z}}$ denote the event that the set of all X-zeros of $\mathcal{Q}$ is exactly $\mathcal{Z}$. Then

$$\Pr\{\bar{A}(t,\varepsilon)\} = \sum_{\mathcal{Z} \subseteq \mathcal{E}} \Pr_{\mathcal{Z}}\{\bar{A}(t,\varepsilon)\} \Pr\{E_{\mathcal{Z}}\} \tag{2.43}$$

$$= \sum_{|\mathcal{Z}| < t-k} \Pr_{\mathcal{Z}}\{\bar{A}(t,\varepsilon)\} \Pr\{E_{\mathcal{Z}}\} \tag{2.44}$$

where $\Pr_{\mathcal{Z}}$ is the conditional probability measure $\Pr\{\,\cdot\,|E_{\mathcal{Z}}\}$. Let us explain the second equality above. Given that $E_{\mathcal{Z}}$ occurred, we can factor $\mathcal{Q}$ as follows

$$\mathcal{Q}(X,Y,Z) \;=\; P(X,Y,Z) \prod_{x\,\in\,\mathcal{Z}} (X-x) \tag{2.45}$$

This makes it clear that ${}_{\mathbf{w}}\!\deg \mathcal{Q} = |\mathcal{Z}| + {}_{\mathbf{w}}\!\deg P$. Hence, if $|\mathcal{Z}| \geqslant t-k$, then ${}_{\mathbf{w}}\!\deg \mathcal{Q}$ is greater than the right-hand side of (2.42), and $\Pr_{\mathcal{Z}}\{\bar{A}(t,\varepsilon)\} = 0$. Now let $\mathcal{X} = \mathcal{E}\backslash\mathcal{Z}$, and let B_Δ be the event that ${}_{\mathbf{w}}\!\deg P \leqslant \Delta$ in the factorization of (2.45). We derive a bound on $\Pr\{B_\Delta \cap E_{\mathcal{Z}}\}$ in what follows.

First, observe that $X-x$ is *not* a factor of P for all $x \in \mathcal{X}$ (in view of (2.45) and the definition of $\mathcal{Z}$), which implies that $P(x,Y,Z) \not\equiv 0$ (it is not the all-zero polynomial). Also note

$$P(x,y,z) = 0 \qquad \forall (x,y,z) \in \mathcal{P} \text{ such that } x \in \mathcal{X} \tag{2.46}$$

We associate with such polynomial P the *set of curves* $C_{\mathcal{X},P}$ defined as follows. Let $\mathcal{X} = \{x_{j_1}, x_{j_2}, \ldots, x_{j_{|\mathcal{X}|}}\}$; we say that

$$\Big\{(x_{j_1},\alpha_1,\beta_1),(x_{j_2},\alpha_2,\beta_2),\ldots,(x_{j_{|\mathcal{X}|}},\alpha_{|\mathcal{X}|},\beta_{|\mathcal{X}|})\Big\} \in C_{\mathcal{X},P}$$

if and only if $P(x_{j_i},\alpha_i,\beta_i) = 0$ for all $i = 1,2,\ldots,|\mathcal{X}|$. Note that total degree of $P(x,Y,Z)$ is less than or equal to ${}_{\mathbf{w}}\!\deg /P/(k-1)$ for all $x \in \mathcal{X}$. Therefore, if ${}_{\mathbf{w}}\!\deg P \leqslant \Delta$, then for each $x \in \mathcal{X}$, there are at most $q\Delta/(k-1)$ pairs (α,β) in $\mathbb{F}_q^2$ such that $P(x,\alpha,\beta) = 0$, by the Schwartz lemma [CLO96]. With reference to Figure 2.9, it follows that

$$\big|C_{\mathcal{X},P}\big| \;\leqslant\; \left(\frac{q\Delta}{k-1}\right)^{|\mathcal{X}|} \tag{2.47}$$

Now let $\mathscr{P}(\mathcal{X},\Delta)$ denote the set of all polynomials of weighted degree at most Δ in $\mathbb{F}_q[X,Y,Z]$ which satisfy (2.46) along with $P(x,Y,Z) \not\equiv 0$ for all $x \in \mathcal{X}$. Further, let us define the corresponding set of curves $\mathscr{C}(\mathcal{X},\Delta) = \bigcup_{P\,\in\,\mathscr{P}(\mathcal{X},\Delta)} C_{\mathcal{X},P}$. Then

$$|\mathscr{C}(\mathcal{X},\Delta)| \;\leqslant\; \left(\frac{q\Delta}{k-1}\right)^{t-|\mathcal{Z}|} q^{N(\Delta)} \tag{2.48}$$

which follows by combining (2.47) with the fact that the total number of polynomials in $\mathscr{P}(\mathcal{X},\Delta)$ is at most $q^{N(\Delta)}$. If both B_Δ and $E_\mathcal{Z}$ occur, then $P \in \mathscr{P}(\mathcal{X},\Delta)$ and the restriction of the random set $\mathcal{P}$ in (2.40) to $\mathcal{X}$ — namely, the set of points of $\mathcal{P}$ with X-coordinate in $\mathcal{X}$ — must belong to $\mathscr{C}(\mathcal{X},\Delta)$. Hence

$$\Pr\left\{B_\Delta \cap E_\mathcal{Z}\right\} \;\leqslant\; \frac{q^{2|\mathcal{Z}|}|\mathscr{C}(\mathcal{X},\Delta)|}{q^{2t}} \tag{2.49}$$

We next consider a carefully chosen value of Δ. Specifically, motivated by the right-hand-side of (2.38), we set

$$\Delta \;=\; \sqrt[3]{6(k-1)^2\big(t-|\mathcal{Z}|\big)} \;-\; k \;-\; \delta n \tag{2.50}$$

where $\delta \leqslant \varepsilon$ is a positive constant to be fixed later. For such Δ, the right-hand-side of (2.38) implies that

$$N(\Delta) - t + |\mathcal{Z}| \;\leqslant\; -\frac{\delta n}{6}\left(\frac{\delta^2}{R^2} - \frac{3\delta}{R}\,\gamma_{t,|\mathcal{Z}|} + 3\gamma_{t,|\mathcal{Z}|}^2\right) \tag{2.51}$$

where $\gamma_{t,|\mathcal{Z}|} \overset{\text{def}}{=} \sqrt[3]{6\big(t-|\mathcal{Z}|\big)/(k-1)}$. In view of (2.44), we are concerned only with the case where $|\mathcal{Z}| < t-k$. For such $\mathcal{Z}$, we have $\gamma_{t,|\mathcal{Z}|} > \sqrt[3]{6}$, which in conjunction with (2.51) implies

$$N(\Delta) - t + |\mathcal{Z}| \;\leqslant\; -\frac{\delta n}{6}\left(\frac{\delta^2}{R^2} - \frac{3\delta}{R}\,\sqrt[3]{6} + 3\sqrt[3]{36}\right) \tag{2.52}$$

provided $\delta \leqslant R\sqrt[3]{6}$. Finally, we observe that with the value of Δ given by (2.50), we have

$$\left(\frac{\Delta}{k-1}\right)^{t-|\mathcal{Z}|} \leqslant \left(\sqrt[3]{\frac{6t}{k-1}}\right)^{n} \leqslant \left(\frac{6}{R}\right)^{n/3} \tag{2.53}$$

Combining (2.49) with (2.48), (2.52), and (2.53), we arrive at the desired bound on $\Pr\{B_\Delta \cap E_\mathcal{Z}\}$: for the Δ in (2.50) with $\delta \leqslant R\sqrt[3]{6}$ and for all $\mathcal{Z}$ with $|\mathcal{Z}| < t-k$, we have

$$\Pr\left\{B_\Delta \cap E_\mathcal{Z}\right\} \;\leqslant\; \left(\frac{6}{R}\right)^{n/3} q^{-\frac{\delta n}{6}\left(\frac{\delta^2}{R^2} - \frac{3\delta}{R}\sqrt[3]{6} + 3\sqrt[3]{36}\right)} \tag{2.54}$$

To complete the proof, we reason as follows. Suppose that the event $\bar{A}(t,\varepsilon) \cap E_\mathcal{Z}$ has occurred, where $|\mathcal{Z}| < t-k$. Then

$$\mathrm{w}\!\deg P \;<\; \sqrt[3]{6t(k-1)^2} \;-\; k - |\mathcal{Z}| - \varepsilon n \tag{2.55}$$

in view of (2.42) and (2.45). Observe that for all $|\mathcal{Z}| < t-k$ and $\delta \leqslant \varepsilon$, the right-hand side of (2.55) is smaller that (2.50). Therefore, the event B_Δ must have also occurred, and we conclude $\Pr_{\mathcal{Z}}\{\bar{A}(t,\varepsilon)\}\Pr\{E_{\mathcal{Z}}\}$ is equal to $\Pr\left\{\bar{A}(t,\varepsilon) \cap E_{\mathcal{Z}}\right\}$ and this is less that or equal to $\Pr\left\{B_\Delta \cap E_{\mathcal{Z}}\right\}$. Finally, combining this with (2.44) and (2.54), we find that

$$\Pr\left\{\bar{A}(t,\varepsilon)\right\} \leqslant 2^n \left(\frac{6}{R}\right)^{n/3} q^{-\frac{\delta n}{6}\left(\frac{\delta^2}{R^2} - \frac{3\delta}{R}\sqrt[3]{6} + 3\sqrt[3]{36}\right)} \tag{2.56}$$

where we have used the trivial bound $\sum_{|\mathcal{Z}|<t-k} 1 \leqslant 2^t \leqslant 2^n$. It remains to set $\delta = \min\{R\sqrt[3]{6}, \varepsilon\}$ and observe that the right-hand-side of (2.56) behaves like $n^{-\Omega(n)}$. This is so because 2 and $6/R$ are constants, while $q = \theta(n)$ by assumption. ∎

The next lemma requires the assumption that the number of channel errors $t = d(\boldsymbol{c}, \boldsymbol{v})$ satisfies

$$n(1-\sqrt{R}) \leqslant t \leqslant n\left(1 - \sqrt[3]{6R^2}\right) \tag{2.57}$$

This assumption is harmless for the following reasons. First, for the proof of Theorem 2.11, we are not interested in the case where t exceeds the right-hand side of (2.57). Second, it is shown in [PV06] that when t is strictly less than the left-hand side of (2.57), the multivariate interpolation decoder *always* recovers $\boldsymbol{c}$ from the channel output $\boldsymbol{v}$. Thus we, again, do not need to worry about this case in the proof of Theorem 2.11.

Lemma 2.15. *Suppose that the total number t of channel errors satisfies* (2.57)*, and let G_1 be the minimal element of the Gröbner basis $\mathcal{G}(\mathcal{P}_{\boldsymbol{v}})$. Then, with probability at least $1 - n^{-\Omega(n)}$ over the choice of $y_1, y_2, \ldots, y_t$ and $z_1, z_2, \ldots, z_t$ in* (2.40)*, the polynomial G_1 is of the form $G_1(X,Y,Z) = p(X)G^*(X,Y,Z)$, where $G^*(X,Y,Z)$ is an irreducible polynomial.*

Proof. Assume to the contrary that $G^*(X,Y,Z)$ is not irreducible, and write G_1 as

$$G_1(X,Y,Z) = U(X,Y,Z)V(X,Y,Z) \tag{2.58}$$

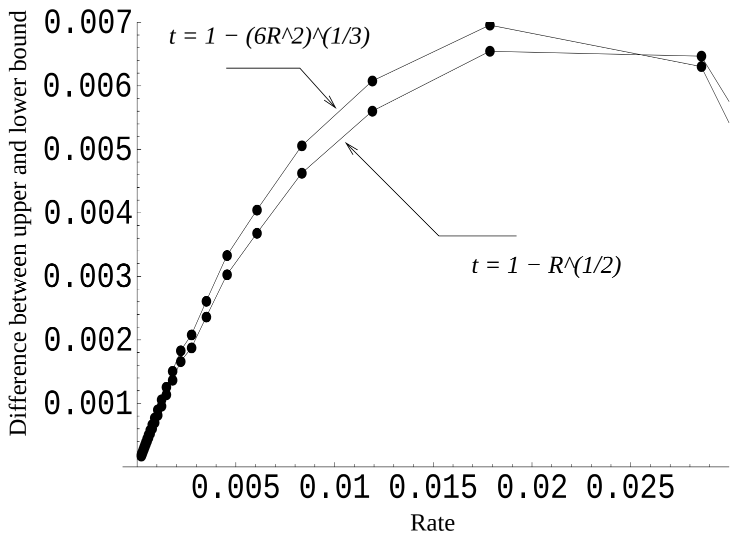


Figure 2.10: Comparing upper and lower bounds on the weighted degree

for some polynomials U and V in $\mathbb{F}_q[X,Y,Z]$. Let Δ_U and Δ_V denote the weighted degrees of U and V, respectively. Let $f(X)$ and $g(X)$ denote the polynomials corresponding to the two codewords of $C_q(n,k)$ that are the components of the transmitted codeword $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ of $\mathbb{C}_Q(n,k)$. That is, let $f(X)$ and $g(X)$ be such that $c_j = f(x_j) + \beta g(x_j)$ for all j.

Claim 1. *Either U or V is in the ideal $\langle Y - f(X), Z - g(X) \rangle$.*

Proof. It is established in Section 2.3 that if t satisfies (2.57), then for $G_1(X,Y,Z)$ we have $G_1\big(X, f(X), g(X)\big) \equiv 0$. In view of (2.58), this implies that either $V\big(X, f(X), g(X)\big) \equiv 0$ and/or $U\big(X, f(X), g(X)\big) \equiv 0$. As shown in Section 2.5, this is exactly what is claimed. ▮

Without loss of generality, assume that U is in the ideal $\langle Y - f(X), Z - g(X) \rangle$. We shall think of the random point set $\mathcal{P}$ in (2.40) as the subset of $\mathcal{P}_{\boldsymbol{v}}$ that corresponds to the t positions in error. We will assume that U passes through some t_1 points of $\mathcal{P}$ while V passes through the $t_2 = t - t_1$ remaining points of $\mathcal{P}$ (as well as, perhaps, other points in $\mathcal{P}$).

Claim 2. *For all $i = 0, 1, \ldots, t_1$, with high probability*

$$N(\Delta_U - i) - (\Delta_U - i) \geqslant t_1 - i \tag{2.59}$$

Proof. Given the set $\mathcal{P}_\nu$ in (2.34), we shift it to construct a modified set $\mathcal{P}'_\nu$ as follows:

$$\mathcal{P}'_\nu \overset{\text{def}}{=} \Big\{ \big(x_j, y_j - f(x_j), z_j - g(x_j)\big) \;:\; (x_j, y_j, z_j) \in \mathcal{P}_\nu \Big\}$$

With respect to the two sets $\mathcal{P}_\nu$ and $\mathcal{P}'_\nu$, we now observe the following facts:

- If $A(X,Y,Z)$ is a polynomial that passes through all the points of $\mathcal{P}_\nu$, the polynomial $A\big(X, Y + f(X), Z + g(X)\big)$ passes through all the points of the set $\mathcal{P}'_\nu$.

- If $A(X,Y,Z)$ is a polynomial that passes through all the points of $\mathcal{P}'_\nu$, the polynomial $A\big(X, Y - f(X), Z - g(X)\big)$ passes through all the points of the set $\mathcal{P}_\nu$.

- The weighted degrees of the polynomial $A(X,Y,Z)$ is the same as $A\big(X, Y \pm f(X), Z \pm g(X)\big)$.

Consequently, for any $U \in \big\langle Y - f(X), Z - g(X)\big\rangle$ that passes through t_1 points of $\mathcal{P}_\nu$, there is an equivalent polynomial U' in the ideal $\langle Y, Z\rangle$ that passes through the t_1 corresponding points of $\mathcal{P}'_\nu$. Note that the weighted degrees of U and U' are the same: $\Delta_U = \Delta_{U'}$. The difference is that U', being in the ideal $\langle Y, Z\rangle$, does not have any monomials of the form X^j for all $j = 0, 1, \ldots, \Delta_{U'}$. Thus, the total number of monomials in the expansion of U' is at most $N(\Delta_{U'}) - \Delta_{U'}$. Using the fact that U' passes through t_1 random points, we now apply an argument similar to Lemma 2.14, and (2.59) follows. ▮

Also note that since U' is nonzero and does not have any monomials of the form X^j, its weighted degree $\Delta_{U'} = \Delta_U$ is at least $k-1$. Finally, observe that for t_2 less than $k-1$, the polynomial $V(X,Y,Z)$ in (2.58) becomes a function of X only, since then the weighted degree of V will be smaller than $k-1$. Therefore, we can assume w.l.o.g. that $t_2 \geqslant k-1$.

Combining all the information on the weighted degrees of U and V, we set up the following optimization problem to obtain a lower bound on the weighted degree of $G_1(X,Y,Z)$.

$$\textbf{\textit{Minimize:}} \quad {}_{\mathbf{W}}\!\deg G_1 \;=\; \Delta_U + \Delta_V \tag{2.60}$$

Subject to:

$$\Delta_V \geqslant \min\{t_2, k-1\}$$

$$\Delta_V \geqslant \min\left\{t_2 - k,\, \sqrt[3]{6t_2(k-1)^2} - k\right\}$$

$$\frac{(\Delta_U - i + k)^3}{6(k-1)^2} - (\Delta_U - i) \;\geqslant\; t_1 - i \quad \text{for } i = 0, 1, \ldots, t_1$$

$$t_1 + t_2 = t, \quad t_2 > k-1, \quad \Delta_U \geqslant k-1$$

$$n(1-\sqrt{R}) \;\leqslant\; t \;\leqslant\; n\left(1 - \sqrt[3]{6R^2}\right)$$

It turns out that the minimum in (2.60) always exceeds the lower bound ${}_{\mathbf{W}}\!\deg G_1 \leqslant s(k-1)$ of Lemma 2.13. The difference between (2.60) and $s(k-1)$ is plotted in Figure 3 for all rates that satisfy (2.32). Since this difference is positive, we have arrived at a contradiction and (2.58) cannot hold. ∎

From this point on, we give only a brief sketch for the rest of the proof. From Theorem 2.12, we know that if some other Gröbner basis polynomial G in $\mathcal{G}(\mathcal{P}_{\boldsymbol{v}})$ does not have G^* as its factor, the MID algorithm will successfully recover the transmitted codeword $\boldsymbol{c}$ whenever the number t of channel errors is strictly smaller than $n - {}_{\mathbf{W}}\!\deg G$. In order to obtain a bound on the decoding radius of the MID algorithm, we derive an upper bound on the weighted degree of such polynomial G.

To this end, we use the concept of a *delta-set*. The delta-set $\Delta(\mathcal{I})$ of a polynomial ideal $\mathcal{I}$ is defined as the set of all monomials that are ***not*** the leading monomials of the polynomials in $\mathcal{I}$. It was proved in [MTV04] that

$$\left|\Delta\big(\mathcal{I}(\mathcal{P}_{\boldsymbol{v}})\big)\right| \;=\; |\mathcal{P}_{\boldsymbol{v}}| \;=\; n \tag{2.61}$$

Now, let $\mathcal{F}(\mathcal{P}_{\boldsymbol{v}})$ denote the set of all elements of $\mathcal{G}(\mathcal{P}_{\boldsymbol{v}})$ that do not have G^* as a factor. Let G denote the minimal element of $\mathcal{F}(\mathcal{P}_{\boldsymbol{v}})$. Then we show that

$${}_{\mathbf{W}}\!\deg G \;\leqslant\; n\left(\sqrt[3]{6R^2} + O(R^{5/3})\right) \tag{2.62}$$

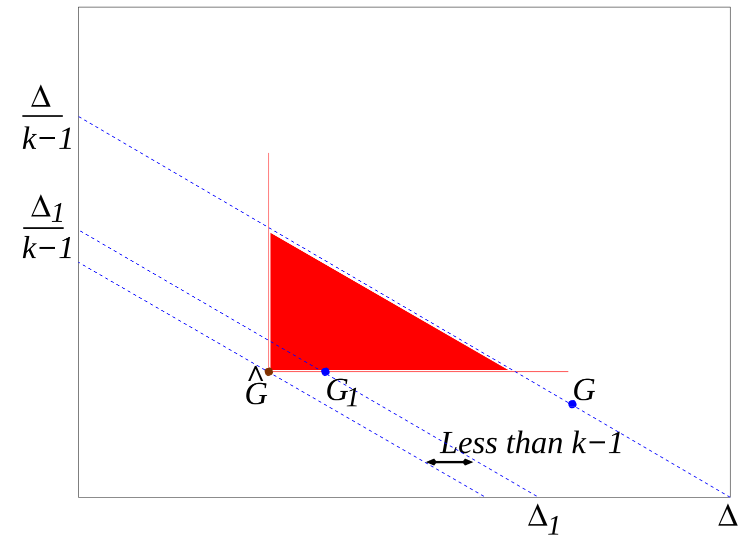


Figure 2.11: Deriving the bound on $_{\mathbf{w}}\deg G^*(X,Y,Z)$

with probability at least $1 - n^{-\Omega(n)}$ provided the number t of channel errors satisfies (2.57). The proof of (2.62) is essentially based on the observation that if $_{\mathbf{w}}\deg G$ is too large, then the size of the delta-set of $\mathcal{I}(\mathcal{P}_v)$ becomes larger than n, in contradiction with (2.61). This is so because G_1 can only "carve-out" a small number of monomials from the delta-set. Specifically, let $\Delta_1 = {}_{\mathbf{w}}\deg G_1$ and $\Delta = {}_{\mathbf{w}}\deg G$. Then Lemma 2.14 implies that with high probability Δ_1 is greater than or equal to $\sqrt[3]{6(k-1)^2 t} - k$. On the other hand, using the argument outlined above, we obtain

$$\frac{\Delta^3}{6(k-1)^2} - \frac{\big(\Delta - (\Delta_1 - k) + k\big)^3}{6(k-1)^2} \leqslant n$$

Combining these two inequalities yields the bound (2.62) on Δ, which finally establishes Theorem 2.11.

The results of the Chapter 2 have been presented, in part, at the 43^{ed} *Annual Allerton Conference on Communication, Control, and Computing*, Parvaresh, Farzad; Vardy, Alexander, and at the *2006 International Symposium on Information Theory (ISIT)*, Parvaresh, Farzad; Taghavi, Mohammad; Vardy, Alexander. The dissertation author was the primary investigator and author of both papers.

CHAPTER 3

Correcting errors beyond the Guruswami-Sudan radius

We introduce a new family of error-correcting codes that have a polynomial-time encoder and a polynomial-time list decoder, correcting a fraction of adversarial errors up to

$$\tau_M = 1 - \sqrt[M+1]{M^M R^M}$$

where R is the rate of the code and $M \geqslant 1$ is an arbitrary integer parameter. This makes it possible to decode beyond the Guruswami-Sudan radius of $1 - \sqrt{R}$ for all rates less than $1/16$. Stated another way, for any $\varepsilon > 0$, we can list decode in polynomial time a fraction of errors up to $1 - \varepsilon$ with a code of length n and rate $\Omega\big(\varepsilon/\log(1/\varepsilon)\big)$, defined over an alphabet of size $n^M = n^{O(\log(1/\varepsilon))}$. Notably, this error-correction is achieved in the worst-case against adversarial errors: a probabilistic model for the error distribution is neither needed nor assumed. The best results at the time of publication of the paper for polynomial-time list decoding of adversarial errors required a rate of $O(\varepsilon^2)$ to achieve the correction radius of $1-\varepsilon$. Later, Guruswami and Rudra [GR06c] show that a very specific class of codes we define here combined by a compression/decompression technique can achieve the bound of $1 - \sqrt[M+1]{R^M}$.

Our codes and list decoders are based on two key ideas. The first is the transition from bivariate polynomial interpolation, pioneered by Sudan and

Guruswami-Sudan [GS99, Sud97], to multivariate interpolation decoding. The second idea is to part ways with Reed-Solomon codes, for which numerous prior attempts [BKY03, CS03, GS99, PV04a] at breaking the $O(\varepsilon^2)$ rate barrier in the worst-case were unsuccessful. Rather than devising a better list decoder for Reed-Solomon codes, we devise better codes. Standard Reed-Solomon encoders view a message as a polynomial $f(X)$ over a field $\mathbb{F}_q$, and produce the corresponding codeword by evaluating $f(X)$ at n distinct elements of $\mathbb{F}_q$. Herein, given $f(X)$, we first compute one or more related polynomials $g_1(X), g_2(X), \ldots, g_{M-1}(X)$ and produce the corresponding codeword by evaluating all these polynomials. Correlation between $f(X)$ and $g_i(X)$, carefully designed into our encoder, then provides the additional information we need to recover the encoded message from the output of the multivariate interpolation process.

3.1 Introduction

Let q be a power of a prime, let $\mathbb{F}_q$ denote the finite field of order q, and let $\mathbb{F}_q^n$ be the vector space of n-tuples over $\mathbb{F}_q$, endowed with the Hamming metric. An *error-correcting code* $\mathbb{C}$ of length n is simply a subset of $\mathbb{F}_q^n$, whose elements are called *codewords*. The *minimum distance* d of $\mathbb{C}$ is the minimum Hamming distance between distinct codewords, while its *rate* is defined as $R = \log_q|\mathbb{C}|/n$. Thus an *encoder* for a code $\mathbb{C} \subseteq \mathbb{F}_q^n$ of rate R is an injective function that maps $\lfloor Rn \rfloor$ arbitrary message symbols over an alphabet of size q into n coded symbols over $\mathbb{F}_q$ (a codeword of $\mathbb{C}$). One of the central questions in coding theory can be stated as follows:

What is the largest possible fraction of errors
that a code $\mathbb{C}$ of rate R can correct? $(\star)$

To make sense of this question, we first need to make it precise. What are "errors" and what does it mean to "correct" them? Throughout this paper, we consider the case of *adversarial errors*. We say that a code $\mathbb{C}$ corrects up to t errors if the "correction" is *guaranteed* whenever we are presented with a vector $\boldsymbol{v} = \boldsymbol{c} + \boldsymbol{e} = (v_1, v_2, \ldots, v_n) \in \mathbb{F}_q^n$ that differs from the original codeword

$\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathbb{C}$ in at most t positions. An alternative approach would be to assume a probability distribution on the error values (the differences $e_j = v_j - c_j$) and then require "correction" only with a certain (high) probability. For more on this, see for example [RU05]. Herein, a probabilistic model for the distribution of error values is *not* assumed. The "correction" itself can be also understood in a number of ways. If correction is defined to be "given $\boldsymbol{v}$, output the original message" and no limit is imposed on the time it might take to produce such output, then any code $\mathbb{C}$ can correct precisely $\lfloor (d-1)/2 \rfloor$ adversarial errors, where d is the minimum distance of $\mathbb{C}$. Thus answering the question in $(\star)$ becomes equivalent to maximizing the relative distance d/n for a given rate R. For much more on this classical problem, see e.g. [MS81]. However, it makes sense to modify the definition of "correction" in two important ways. First, instead of requiring the decoder for $\mathbb{C}$ to produce *only* the original message, one could define correction to be "given $\boldsymbol{v}$, output a small list of messages that is guaranteed to include the original message." This is known as the *list decoding problem*, dating back to the work of [Eli57,Woz58] in the 1950s. Second, it makes sense to require that decoding — namely, producing this list of messages — is accomplished in time that is *polynomial* in the code length n. This, in particular, guarantees that the size of the list produced by the decoder is at most polynomial in n.

For adversarial errors, with error-correction defined as above, this work gives the best known answer to the question in $(\star)$, at least for low rates. We present a new class of algebraic error-correcting codes that correct a fraction of

$$\tau \;=\; 1 - O\big(R \log(1/R)\big) \tag{3.1}$$

adversarial errors in polynomial time. This exceeds the best previously known bound $\tau_{\mathrm{GS}} = 1 - \sqrt{R}$, due to Guruswami and Sudan [GS99], for all rates $R < 1/16$.

3.1.1 Prior work on the list decoding problem

Formally, a code $\mathbb{C} \subseteq \mathbb{F}_q^n$ is said to be (τ, L) *list decodable* if for all $\boldsymbol{v} \in \mathbb{F}_q^n$, the number of codewords at Hamming distance τn or less from $\boldsymbol{v}$ is at most L. Most of the applications of list decodable codes in theoretical computer science correspond to the "high noise" situation where τ is close to 1, and we henceforth emphasize this fact by writing τ as $1 - \varepsilon$. For an excellent overview of the list decoding problem and its applications, read Guruswami [Gur01].

In particular, it is observed in [Gur01] that $\big(1 - \varepsilon, O(1/\varepsilon)\big)$ list decodable codes of rate $\Omega(\varepsilon)$ *exist*, and that $\Omega(\varepsilon)$ is the *best rate possible* for such codes. However, this existence result is based on random coding arguments — we do not know how to explicitly *construct* such codes in polynomial time, let alone *decode* them in polynomial time.

The first real breakthrough on the list decoding problem was achieved by Sudan [Sud97] and by Guruswami-Sudan [GS99] in the late 1990s. The ground-breaking work of [GS99] shows that using Reed-Solomon (and other algebraic) codes, it is possible to list decode in polynomial time a fraction of $1 - \varepsilon$ errors with a code of rate ε^2. Since then, the rate bound of $O(\varepsilon^2)$ has become a seemingly unsurmountable "barrier" for list correcting a fraction of $1 - \varepsilon$ adversarial errors in polynomial time. To quote from [Gur04]:

> *Despite much progress in the general area of list decoding, the $O(\varepsilon^2)$ bound has been a "barrier" for the rate for codes list decodable up to a fraction $(1 - \varepsilon)$ of errors.* [...] *The question of improving the rate beyond the $O(\varepsilon^2)$ barrier is one of the central open questions in the subject of list decoding.*

Here is a brief, and necessarily incomplete, overview of prior attempts at breaking this barrier. In [GI01], the authors present a randomized construction of codes of rate $\Omega(\varepsilon)$ together with an algorithm to list decode such codes up to a fraction of $1 - \varepsilon$ errors. However, the construction of [GI01] is not explicit, since there is no efficient way to certify that the resulting codes have the desired list decoding properties. Moreover, the list decoding algorithm of [GI01] runs in *subexponential* time of the form $2^{O(n^{1/\tau})}$. The first explicit construction of codes with rate better than $\Omega(\varepsilon^2)$ was given by Guruswami in [Gur04]. This construction uses *extractors* in an ingenious way to come up with $(1-\varepsilon, L)$ list decodable codes. The rate that these code achieve is $\varepsilon/\log^{O(1)}(1/\varepsilon)$, which is not far from our results. However, the list decoding algorithm of [Gur04] is *not* polynomial-time: it runs in subexponential time of the form $2^{c(Rn)^{1/\tau}}$ and could return subexponential-sized lists as output. There are other graph-based constructions, most notably by Guruswami and Indyk [GI02], that for rates $\Omega(\varepsilon^2)$ correct a fraction of $O(1-\varepsilon)$ errors. The Guruswami-Indyk codes [GI02] are encodable and list decodable in almost linear time; however, they do not break the $O(\varepsilon^2)$ rate-barrier. A completely different approach was recently developed by Muralidhara and Sen [MS05] to show that Reed-Solomon codes are $(1-\varepsilon, L)$ list decodable at a rate of $\Omega\big((\varepsilon + \frac{c}{n})^2\big)$ for any constant $c > 0$. Unfortunately, the list size and the complexity of decoding in [MS05] grow at least as $\Omega(n^c)$, which again does not allow to break the $O(\varepsilon^2)$ rate-barrier in polynomial time.

Finally, we should mention the work of [BKY03], [CS03], and [PV04a]. Bleichenbacher, Kiayias, and Yung [BKY03] study Reed-Solomon codes of a certain type over an alphabet of size $Q = n^M$. They devise a unique (list of size one) decoding algorithm that, with *probability* at least $1 - O(1/n)$ on the *Q-ary symmetric channel*, corrects a fraction of $1-\varepsilon$ errors using a code of rate $R = \big(\varepsilon(M+1) - 1\big)/M$. Coppersmith and Sudan [CS03] consider the same Reed-Solomon codes[1] over $\mathbb{F}_Q$ and devise a list decoding algorithm that corrects a fraction of $1-\varepsilon$ errors using a code of rate $R = \Omega(\varepsilon^\alpha)$, for any constant $\alpha > 1$. However, this correction is again achieved only with a certain *probability*, which Coppersmith and Sudan [CS03] show to be at least $\Omega\big(R^{M/(M+1)}\big)$, again assuming the same Q-ary symmetric channel model. Thus neither [BKY03] nor [CS03] deal with adversarial errors. In our earlier work [PV04a], we do not assume a probabilistic model for the error distribution. Nevertheless, we cannot provide a *guarantee* on the list decoding radius of our algorithm in [PV04a] that would

[1]Neither Bleichenbacher, Kiayias, and Yung [BKY03] nor Coppersmith and Sudan [CS03] recognize that the codes they consider are, in fact, Reed-Solomon codes, but this is shown in [PV06].

break the $O(\varepsilon^2)$ rate-barrier either.

In summary, there was no construction known (to us) of error-correcting codes that are $(1-\varepsilon, L)$ list decodable in polynomial time against adversarial errors on one hand, and exceed the $O(\varepsilon^2)$ rate-barrier on the other hand.

3.1.2 Our main results

Herein, we present such a construction. We will explicitly describe a new family of algebraic error-correcting codes and devise a novel list decoding algorithm for these codes, which makes it possible to prove the following theorem.

Theorem 3.1. *Let q be a power of a prime. Then for all positive integers m, M, $n \leqslant q$, and $k \leqslant n$, there is a code $\mathbb{C}$ of length n and rate $R = k/nM$ over $\mathbb{F}_Q = \mathrm{GF}(q^M)$, equipped with an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ that have the following properties. Given any vector $\boldsymbol{v} \in \mathbb{F}_Q^n$, the decoder $\mathcal{D}$ outputs the list of all codewords that differ from $\boldsymbol{v}$ in at most*

$$t = \left\lfloor n - n \sqrt[M+1]{M^M R^M \left(1+\frac{1}{m}\right) \cdots \left(1+\frac{M}{m}\right)} - \frac{1}{m} \right\rfloor$$

positions (where m, M are arbitrary integer parameters). The size of this list is at most

$$L = \left\lceil m \sqrt[M+1]{\frac{(1+\frac{1}{m})(1+\frac{2}{m})\cdots(1+\frac{M}{m})}{MR}} \right\rceil^M + o(1)$$

Moreover, both the encoder $\mathcal{E}$ and the decoder $\mathcal{D}$ run in time (number of operations in $\mathbb{F}_q$) that is bounded by a polynomial in n, M, and m.

To arrive at our main result, it remains to set the parameters in Theorem 3.1 appropriately. Given a small $\varepsilon > 0$, we write $t/n = 1 - \varepsilon$. Then, assuming m is sufficiently large, the best choice for M is given by $M = \ln(1/\varepsilon)$. Note that

$$F_M(m) \overset{\text{def}}{=} \prod_{i=1}^{M} \left(1 + \frac{i}{m}\right) = 1 + \frac{M^2 + M}{2m} + O\left(\frac{1}{m^2}\right)$$

Thus we can make this function arbitrarily close to 1, regardless of the value of ε, provided $m = \omega(M^2)$. For example, we could take $m = M^2 \log\log M$. Then the rate R and the list-size L in Theorem 3.1 are given by

$$R \simeq \frac{\varepsilon^{\frac{M+1}{M}}}{M} = \frac{e^{-1}\varepsilon}{\ln\frac{1}{\varepsilon}}; \qquad L \simeq \frac{m^M}{\varepsilon} = \left(\frac{1}{\varepsilon}\right)^{O\left(\log\log\frac{1}{\varepsilon}\right)}$$

Finally, taking $n = q$, the code $\mathbb{C}$ in Theorem 3.1 is defined over an alphabet of size $Q = q^M = n^{\lfloor \log(1/\varepsilon) \rfloor}$.

Corollary 3.2. *For all sufficiently small $\varepsilon > 0$, we can explicitly construct a family of*

$$\left(1-\varepsilon, (1/\varepsilon)^{O\left(\log\log\frac{1}{\varepsilon}\right)}\right)$$

list decodable codes of rate $R = \Omega\big(\varepsilon/\log(1/\varepsilon)\big)$ defined over an alphabet of size $n^{O(\log(1/\varepsilon))}$. Moreover, such codes can be encoded and list decoded in polynomial time.

The codes we construct to establish Theorem 3.1 are certain highly-structured subsets of certain highly-structured Reed-Solomon codes. As we shall see, the underlying ideas are extremely simple. Indeed, proving Theorem 3.1, at least for the important special case $M = 2$, will not require much technical sweat. It is also curious to note that, in general, our codes are *not linear* — neither over $\mathbb{F}_Q$ nor over $\mathbb{F}_q$. This unusual circumstance does not seem to impede in any way polynomial-time encoding and list decoding of these codes.

An important shortcoming of our result in Chapter 3.2 is that the alphabet size grows very fast: $n^{O(\log(1/\varepsilon))}$ is *much* larger than $O(1/\varepsilon)$, which is the best one could hope for [Gur01]. However, in a recent paper Guruswami and Patthack [Gur05a,GP06] extends our methods by replacing the underlying Reed-Solomon codes with more general algebraic-geometric codes. Using this (highly nontrivial!) generalization, they then shows in [GP06] that it is possible to reduce the alphabet size to $(1/\varepsilon)^{O(\log(1/\varepsilon))}$ — which does not depend on the code length n — at the expense of slightly decreasing the rate to $\Omega\big(\varepsilon/\log^2(1/\varepsilon)\big)$.

Another important extension of our codes is the construction introduced by Guruswami and Rudra in [GR06c, GR06b]. They show that for a specific set of parameters our codes can be *compressed* to *folded Reed-Solomon* codes. Currently, for a specific set of data symbols we transmit evaluations of two correlated polynomials $f(X)$ and $g(X)$ at n points of the finite field $\mathbb{F}_q$. With the compression technique of Guruswami and Rudra, we need only to send the evaluation of $f(X)$ and at the decoder evaluation of $g(X)$ can be inferred by decompression of evaluation of $f(X)$. This improve the decoding radius up to a fraction of $1-(1+\varepsilon)R^{M/(M+1)}$ of errors for any $\varepsilon > 0$ and $0 < R < 1$. In the limit for large M, Guruswami and Rudra can decode up to a fraction of $1-R-\varepsilon$ of errors in time $(N/\varepsilon)^{O(1/\varepsilon)}$ where N is the block length of the code with the alphabet size of $(N/\varepsilon)^{O(1/\varepsilon^2)}$.

3.1.3 The underlying ideas

Our results in this chapter are based upon two key ideas. The first is the transition from bivariate polynomial interpolation, pioneered by Sudan [Sud97] and Guruswami-Sudan [GS99], to *multivariate interpolation decoding*. It was recognized early on that decoding Reed-Solomon codes is equivalent to the problem of reconstructing univariate polynomials from their noisy evaluations. Conventional decoding [Ber68, WB86] thus attempts to solve this problem using *univariate* polynomial interpolation. The breakthrough achieved by Sudan [Sud97] and Guruswami-Sudan [GS99] was due in large part to the transition from univariate to *bivariate* polynomial interpolation. Herein, we move on to interpolation decoding in $M+1$ variables, where $M \geqslant 1$ is the integer parameter in Theorem 3.1. Much of this framework was developed in our earlier papers [PV04a, PV06] and presented in Chapter 2. However, to keep the independency of chapters of the dissertation, we will give a brief primer on multivariate interpolation decoding in the next section. The readers that are familiar with material of Section 2.4 of Chapter 2 may skip the next section.

The second key idea is to part ways with Reed-Solomon codes, for which numerous prior attempts [BKY03, CS03, GS99, MS05, PV04a] at breaking the $O(\varepsilon^2)$ rate barrier in the worst-case proved unsuccessful. Rather than trying to devise a better list decoder for Reed-Solomon codes (which, as it turns out [PV06], is what all the previous papers attempted to do), we construct better codes. Our construction is still based upon the Reed-Solomon codes, yet differs from them in an essential way. Standard Reed-Solomon encoders view a message as a polynomial $f(X)$ over a field $\mathbb{F}_q$, and produce the corresponding codeword by evaluating $f(X)$ at n distinct elements of $\mathbb{F}_q$. In this paper, given $f(X)$, we first compute $M-1$ related polynomials $g_1(X), g_2(X), \ldots, g_{M-1}(X)$ and produce the corresponding codeword by evaluating *all* these polynomials. In the degenerate case $M = 1$, we thus rederive the Guruswami-Sudan list decoding algorithm [GS99] for RS codes. However, when $M \geqslant 2$, we can do better: correlation between $f(X)$ and $g_i(X)$, carefully designed into our encoder, provides precisely the kind of information we need to break the $O(\varepsilon^2)$ rate-barrier for adversarial errors.

3.1.4 Overview of the chapter

The next section is a primer on multivariate interpolation. In Sections 3.3 and 3.4, we consider in detail the first nontrivial case $M = 2$. As we shall see, for $M = 2$, there are several different ways to obtain the necessary correlation between $f(X)$ and $g(X)$. In Section 3.3, we describe a particularly elegant and simple way to do so, namely

$$g(X) \;=\; \big(f(X)\big)^a \mod e(X) \tag{3.2}$$

where $e(X)$ is an arbitrary irreducible (over $\mathbb{F}_q$) polynomial of degree k, and a is an arbitrary (but sufficiently large) integer. The encoding rule (3.2) leads to a simple list decoding algorithm. Using this algorithm, we give in Section 3.3 a proof of Theorem 3.1 for the special case $M = 2$. In Section 3.4, we show that the encoding rule (3.2) is just an instance of a much more general situation where the pair $\big(f(X), g(X)\big)$ can be regarded as a zero of an irreducible polynomial $\mathcal{T}(Y, Z)$ over the field $\mathbb{K} = \mathbb{F}_{q^k}$. We furthermore extend the list decoding algorithm of Section 3.3 to this more general scenario. Finally, in Section 3.5, we generalize the construction and the decoding algorithm of Section 3.3 for all $M \geqslant 2$, and thereby complete the proof of Theorem 3.1.

3.2 Multivariate interpolation

We will describe in detail only the (first nontrivial) case of trivariate interpolation. Generalization to multivariate interpolation in an arbitrary number of variables should be fairly straightforward, and will be briefly sketched-out later on.

We need some definitions pertaining to trivariate polynomials. Thus let $Q(X,Y,Z) = \sum_{i,j,k=0}^{\infty} q_{i,j,k} X^i Y^j Z^k$ be a polynomial over $\mathbb{F}_q$, and let $\mathbb{N}$ denote the natural numbers. For $a,b,c \in \mathbb{N}$, the corresponding *Hasse derivative* of Q is

$$\mathscr{D}_{a,b,c}\big[Q(X,Y,Z)\big] \overset{\text{def}}{=} \sum_{i=a}^{\infty}\sum_{j=b}^{\infty}\sum_{k=c}^{\infty} \binom{i}{a}\binom{j}{b}\binom{k}{c} q_{i,j,k}\, X^{i-a}Y^{j-b}Z^{k-c} \tag{3.3}$$

The polynomial $Q(X,Y,Z)$ is said to have a *zero of multiplicity* m at a point $(x_0,y_0,z_0) \in \mathbb{F}_q \times \mathbb{F}_q \times \mathbb{F}_q$ if

$$\mathscr{D}_{a,b,c}\big[Q(X,Y,Z)\big]\Big|_{(x_0,y_0,z_0)} = 0 \qquad \text{for all } a,b,c \in \mathbb{N} \text{ with } a+b+c < m \tag{3.4}$$

We next define the weighted degree of trivariate polynomials. While this is a fairly general concept, we will only need the $(1,k-1,k-1)$-weighted degree, where $k-1$ is the degree of the message polynomial $f(X)$. Thus, we define the *weighted degree* of a monomial $X^aY^bZ^c$ as follows

$$\text{w}\deg X^aY^bZ^c \overset{\text{def}}{=} a + (k-1)b + (k-1)c \tag{3.5}$$

We extend the weighted degree to a monomial ordering $\prec_{\text{w}}$ by augmenting it with the lex order (cf. [CLO96, Chapter 2]). The weighted degree of a polynomial can be now defined as the weighted degree of its leading (under $\prec_{\text{w}}$) monomial.

Now let $x_1, x_2, \ldots, x_n$ be some n distinct elements of $\mathbb{F}_q$, and consider the *interpolation set* $\mathcal{P}$ given by

$$\mathcal{P} \overset{\text{def}}{=} \Big\{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\Big\} \tag{3.6}$$

where $y_j, z_j \in \mathbb{F}_q$ are unrestricted. We define the *interpolation polynomial* with respect to $\mathcal{P}$ as the least weighted degree nonzero polynomial $\mathcal{Q}_m(X, Y, Z)$ in $\mathbb{F}_q[X, Y, Z]$ that has a zero of multiplicity m at each of the n points of $\mathcal{P}$.

It should be obvious that, given $\mathcal{P}$, the interpolation polynomial $\mathcal{Q}_m(X,Y,Z)$ can be computed in time that is bounded by a polynomial in n and m, say with straightforward Gaussian elimination and/or matrix inversion. In fact, using a generalization of the algorithm due to Koetter [Köt96], it is shown in Section 2.4 of Chapter 2 that this computation takes $O\big(n^{8/3}m^8/k^{2/3}\big)$ operations in $\mathbb{F}_q$. Notably, the complexity analysis easily carries over to general multivariate interpolation. Recently Gaborit and Ruatta [GR06a] gave an algorithm with better complexity to solve the Hermite interpolation polynomial.

Lemma 3.3.

$$\mathbf{w}\text{deg}\, \mathcal{Q}_m(X, Y, Z) \leqslant \left\lceil \sqrt[3]{n(k-1)^2 m(m+1)(m+2)} \right\rceil$$

Proof. The proof is a straightforward generalization of the argument used in [GS99] and [KV03a]. Let $N_3(\Delta)$ denote the number of trivariate monomials whose weighted degree is at most Δ. We claim that there exists a nonzero polynomial of weighted degree at most Δ that passes through each point in $\mathcal{P}$ with multiplicity m, provided

$$N_3(\Delta) > n\,\frac{m(m+1)(m+2)}{6} \tag{3.7}$$

Indeed, referring to (3.3), it is clear that (3.4) is just a system of linear constraints on the coefficients of such a polynomial. The total number of linear constraints imposed by passing with multiplicity m through each of n points is given by the right-hand side of (3.7). The total number of unknowns (coefficients of the polynomial) is $N_3(\Delta)$. If the number of unknowns is strictly greater than the number of constraints, the linear system is guaranteed to have a nonzero solution. It follows that any Δ such that $N_3(\Delta)$ satisfies (3.7) is an upper bound on the weighted degree of the interpolation polynomial. It remains to estimate $N_3(\Delta)$. To this end, consider the correspondence between monomials in $\mathbb{F}_q[X,Y,Z]$ and unit cubes in $\mathbb{R}^3$, defined by $X^aY^bZ^c \mapsto K(a,b,c)$, where $K(a,b,c) = [a,a{+}1) \times [b,b{+}1) \times [c,c{+}1)$. Then $N_3(\Delta)$ is equal to the *volume* of a union of such cubes $K(a,b,c)$, the union being taken over all $a,b,c \in \mathbb{N}$ that satisfy

$$a + (k{-}1)b + (k{-}1)c \leqslant \Delta$$

Let $\mathscr{U} \subset \mathbb{R}^3$ denote this union of cubes, and consider the pyramid $\mathscr{P} \subset \mathbb{R}^3$ defined by the four half-planes: $x \geqslant 0$, $y \geqslant 0$, $z \geqslant 0$, and $x + (k{-}1)y + (k{-}1)z \leqslant \Delta$. It is easy to see that $\mathscr{P} \subset \mathscr{U}$, so $N_3(\Delta) = \mathrm{Vol}(\mathscr{U}) > \mathrm{Vol}(\mathscr{P})$. But $\mathrm{Vol}(\mathscr{P}) = \Delta^3/6(k{-}1)^2$ and the lemma follows. ▮

Lemma 3.4. *Let $\mathcal{Q}_m(X,Y,Z)$ be the interpolation polynomial with respect to the set $\mathcal{P}$ in* (3.6)*. Given a pair of polynomials $f(X)$ and $g(X)$ over $\mathbb{F}_q$, define*

$$\Big|(f,g)\Big|_{\mathcal{P}} \overset{\text{def}}{=} \Big|\{\, j \,:\, f(x_j) = y_j \text{ and } g(x_j) = z_j \}\Big| \tag{3.8}$$

Provided the degree of both $f(X)$ and $g(X)$ is at most $k-1$, and moreover

$$\Big|(f,g)\Big|_{\mathcal{P}} \geqslant \left\lceil \sqrt[3]{n\,(k-1)^2\Big(1+\frac{1}{m}\Big)\Big(1+\frac{2}{m}\Big)} + \frac{1}{m} \right\rceil$$

we have $\mathcal{Q}_m\big(X,f(X),g(X)\big) \equiv 0$. Namely, the univariate polynomial $p(X)$ defined as $\mathcal{Q}_m\big(X,f(X),g(X)\big)$ is identically zero.

Proof. If $\deg f(X)$, $\deg g(X) \leqslant k-1$, it follows from the definition of the weighted degree in (3.5) that

$$\deg p(X) \leqslant {}_{\mathbf{W}}\!\deg \mathcal{Q}_m(X,Y,Z) \tag{3.9}$$

On the other hand, for every integer $j \in \{1,2,\ldots,n\}$ that is counted in (3.8), the interpolation polynomial $\mathcal{Q}_m(X,Y,Z)$ has a zero of multiplicity m at the point

$\big(x_j, f(x_j), g(x_j)\big)$. Hence the total number of zeros of $p(X)$ in $\mathbb{F}_q$, counted with multiplicity, is lower-bounded by

$$\#\text{ zeros of } p(X) \;\geqslant\; m\left|(f,g)\right|_{\mathcal{P}} \tag{3.10}$$

If the right-hand side of (3.10) is strictly greater than the right-hand side of (3.9), then $p(X)$ must be the all-zero polynomial. The lemma now follows from Lemma 3.3. ∎

Before concluding this section, we need to establish one more technical lemma.

Lemma 3.5. *A polynomial $Q(X,Y,Z) \in \mathbb{F}_q[X,Y,Z]$ satisfies $Q\big(X, f(X), g(X)\big) \equiv 0$ if and only if it belongs to the ideal generated by $Y - f(X)$ and $Z - g(X)$, that is, if and only if there exist some polynomials $\mathcal{A}, \mathcal{B} \in \mathbb{F}_q[X,Y,Z]$ such that*

$$Q(X,Y,Z) \;=\; \mathcal{A}\big(Y - f(X)\big) + \mathcal{B}\big(Z - g(X)\big) \tag{3.11}$$

Proof. It should be obvious that if $Q(X,Y,Z)$ can be expressed as in (3.11), then $Q\big(X, f(X), g(X)\big) \equiv 0$. To prove the converse, fix a monomial order $\prec$ such that $Y \succ f(X)$ and $Z \succ g(X)$, then divide $Q(X,Y,Z)$ by $Y - f(X)$ and $Z - g(X)$ to express it as follows:

$$Q(X,Y,Z) \;=\; \mathcal{A}\big(Y - f(X)\big) + \mathcal{B}\big(Z - g(X)\big) + R \tag{3.12}$$

The polynomial division algorithm guarantees that none of the monomials in the remainder polynomial $R \in \mathbb{F}_q[X,Y,Z]$ is divisible by the leading term of $Y - f(X)$ or $Z - g(X)$. But since $Y \succ f(X)$ and $Z \succ g(X)$, this simply means that $R(X,Y,Z) = R(X)$. This, in conjunction with (3.12) and the assumption that $Q\big(X, f(X), g(X)\big) \equiv 0$, then implies that $R(X) \equiv 0$, and the lemma follows. ∎

3.3 Simple trivariate codes and their decoding

We are now ready to describe the construction of our codes for the simplest nontrivial case: $M = 2$ (trivariate interpolation) with the encoding rule of (3.2).

This is done in the next subsection. Then, in Section 3.3.2, we describe a decoding algorithm for these codes that achieves the error-correction claimed in Theorem 3.1 for the special case of $M = 2$.

3.3.1 Code parameters and encoder mapping

A specific code $\mathbb{C}$ in the family constructed in this section is specified by the following set of parameters.

Underlying field order: An integer q, which is a power of a prime; determines the message symbol alphabet.

Length: A positive integer $n \leqslant q$, which determines the length of the code (a good choice is $n = q$).

Evaluation set: A set of n elements $x_1, x_2, \ldots, x_n$ of $\mathbb{F}_q$, that must be distinct but are otherwise arbitrary.

Rate parameter: A positive integer k which, along with n, determines the rate of the code.

Basis: A fixed basis $\{1, \alpha\}$ for $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$.

Multiplicity parameter: An arbitrary positive integer m, which determines decoding interpolation multiplicity.

Exponentiation modulus: A polynomial $e(X) \in \mathbb{F}_q[X]$ of degree k, that is irreducible over $\mathbb{F}_q$ but otherwise arbitrary; determines the encoder mapping.

Exponentiation degree: A positive integer a that satisfies

$$a \geqslant \left\lceil m \sqrt[3]{\frac{n}{k-1}\left(1+\frac{1}{m}\right)\left(1+\frac{2}{m}\right)} + \frac{1}{k-1} \right\rceil \tag{3.13}$$

but is otherwise arbitrary; along with $e(X)$, determines the encoder mapping, also determines the list-size.

We assume that all these parameters are fixed in advance, and known to both the encoder and the decoder for $\mathbb{C}$.

Our code $\mathbb{C}$ is a subset of the set of vectors of length n over $\mathbb{F}_{q^2}$. We define $\mathbb{C}$ by describing a bijective encoder map $\mathcal{E}\colon \mathbb{F}_q^k \to \mathbb{C}$, as follows. Given k arbitrary message symbols $u_0, u_1, \ldots, u_{k-1} \in \mathbb{F}_q$, the encoder first constructs the polynomial $f(X) = \sum_{i=0}^{k-1} u_i X^i$, and then computes

$$g(X) \;=\; \big(f(X)\big)^a \mod e(X) \tag{3.14}$$

over $\mathbb{F}_q$. The codeword $(c_1, c_2, \ldots, c_n) \in \mathbb{C}$ corresponding to the message $u_0, \ldots, u_{k-1}$ is then given by

$$c_j \;=\; f(x_j) + \alpha g(x_j) \qquad \text{for } j = 1, 2, \ldots, n \tag{3.15}$$

It is obvious from the above description that the rate of $\mathbb{C}$ is $R = \log_{q^2}|\mathbb{C}|/n = k/2n$. The minimum distance of $\mathbb{C}$ is at least $n-k+1$, since $\mathbb{C}$ is a subset of a Reed-Solomon code $\mathscr{C}$ of length n and dimension k over $\mathbb{F}_{q^2}$. We leave the proof of this fact as an exercise for the reader.

It is interesting to observe that in most cases $\mathbb{C}$ is not a *linear subspace* of $\mathscr{C}$ because the computation in (3.14) is, in general, not a linear operation. However, if $q = p^s$ for a prime p and we take the exponentiation degree a to be of the form $a = p^b$, where b is a multiple of s, then the code $\mathbb{C}$ becomes $\mathbb{F}_q$-linear. It is still not a linear code in the usual sense of the word, since it is not linear over the field $\mathbb{F}_{q^2}$ over which it is defined. Making it linear over $\mathbb{F}_{q^2}$ requires $a-1$ to be a multiple of q^k-1, and leads to the degenerate case where $g(X) \equiv f(X)$ since the modular exponentiation in (3.14) then becomes the identity operation.

Finally, we need to show that the encoder map in (3.14) and (3.15) can be computed in polynomial time. To this end, it is convenient to think of the polynomials $f(X)$ and $g(X)$ of degree $\leqslant k-1$ as elements in the extension field

$$\mathbb{K} \overset{\text{def}}{=} \mathbb{F}_q[X] \,/\, \langle e(X) \rangle \tag{3.16}$$

of order q^k. Let β and γ denote the elements of $\mathbb{K}$ that correspond to $f(X)$ and $g(X)$, respectively. Then the computation in (3.14) becomes simply $\gamma = \beta^a$ in

$\mathbb{K}$. We can now use a result of Von zur Gathen and Nöcker [GN97], who show how to compute β^a with $O(k^2 \log\log k)$ operations in the *ground field* $\mathbb{F}_q$, using storage for only $O(k/\log^2 k)$ elements of $\mathbb{K}$.

3.3.2 Simple trivariate interpolation decoding

We have seen in the foregoing subsection that the construction of our codes is conceptually quite simple. Their decoding is equally simple. Given a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ over $\mathbb{F}_{q^2}$, we first decompose each v_j into its two components over $\mathbb{F}_q$, using the fixed basis $\{1, \alpha\}$. Thus we write

$$v_j \;=\; y_j + \alpha z_j \qquad \text{for } j = 1, 2, \ldots, n$$

where y_j and z_j are in $\mathbb{F}_q$. We then set up the interpolation set $\mathcal{P}$ exactly as in (3.6), namely

$$\mathcal{P} \;=\; \Big\{ (x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n) \Big\}$$

where $x_1, x_2, \ldots, x_n$ is the evaluation set for $\mathbb{C}$. As observed in Section 3.2, the corresponding interpolation polynomial $\mathcal{Q}_m(X, Y, Z)$ can be computed in time $O\big(n^{8/3} m^8 / k^{2/3}\big)$.

Lemma 3.6. *Suppose that a codeword $\boldsymbol{c} \in \mathbb{C}$ differs from the given vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ in at most*

$$t \;=\; \left\lfloor n - \sqrt[3]{n\,(k-1)^2 \left(1 + \frac{1}{m}\right)\left(1 + \frac{2}{m}\right)} \;-\; \frac{1}{m} \right\rfloor \tag{3.17}$$

positions, and let $f(X)$, $g(X)$ denote the message polynomial(s) that produce $\boldsymbol{c}$ according to the mapping (3.14) – (3.15). *Then $\mathcal{Q}_m\big(X, f(X), g(X)\big)$ is equivalent to zero.*

Proof. This follows immediately from Lemma 3.4, upon observing that $\big|(f,g)\big|_{\mathcal{P}} \geqslant n - t$. ∎

It remains to show how to recover the message $f(X)$ from the interpolation polynomial $\mathcal{Q}_m(X, Y, Z)$, assuming that $\mathcal{Q}_m\big(X, f(X), g(X)\big) \equiv 0$. To this end, we first reduce $\mathcal{Q}_m(X,Y,Z)$ modulo $e(X)$. That is, we compute

$$P(Y,Z) \overset{\text{def}}{=} \mathcal{Q}_m(X,Y,Z) \mod e(X) \tag{3.18}$$

To see that $P(Y,Z)$ is indeed a bivariate polynomial, first write $\mathcal{Q}_m(X,Y,Z)$ as an element of $\mathbb{F}_q[X][Y,Z]$, namely

$$\mathcal{Q}_m(X,Y,Z) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} q_{i,j}(X)\, Y^i Z^j \tag{3.19}$$

and let $p_{i,j}(X) = q_{i,j}(X) \mod e(X)$. Then clearly $p_{i,j}(X)$ can be regarded as an element of the extension field $\mathbb{K}$ defined in (3.16), and therefore $P(Y,Z) \in \mathbb{K}[Y,Z]$. We point out that this is mainly a syntactic trick, but it greatly simplifies the derivation in what follows.

Lemma 3.7. *The polynomial $P(Y,Z)$ defined by* (3.18) *is not the all-zero polynomial.*

Proof. Assume to the contrary that $P(Y,Z) \equiv 0$, or equivalently that $e(X)$ is a factor of $q_{i,j}(X)$ for all i and j, where $q_{i,j}(X)$ are the coefficients in (3.19). But this means that $e(X)$ is a factor of $\mathcal{Q}_m(X,Y,Z)$, so we can define the polynomial

$$\mathcal{Q}'(X,Y,Z) \overset{\text{def}}{=} \frac{\mathcal{Q}_m(X,Y,Z)}{e(X)} \tag{3.20}$$

Using the fact that $e(X)$ is irreducible, it is not difficult to show that if $\mathcal{Q}_m(X,Y,Z)$ satisfies all the interpolation constraints, then so does $\mathcal{Q}'(X,Y,Z)$. Furthermore, we have ${}_{\mathbf{w}}\deg \mathcal{Q}'(X,Y,Z) = {}_{\mathbf{w}}\deg \mathcal{Q}_m(X,Y,Z) - \deg e(X)$ in view of (3.20). But this contradicts the definition of the interpolation polynomial $\mathcal{Q}_m(X,Y,Z)$ as the *least weighted-degree* polynomial satisfying the interpolation constraints. ∎

Lemma 3.8. *Suppose that $\mathcal{Q}_m\big(X,f(X),g(X)\big) \equiv 0$, and let β and γ denote the elements of $\mathbb{K}$ that correspond to $f(X)$ and to $g(X)$, respectively. Then $P(\beta,\gamma) = 0$.*

Proof. If $\mathcal{Q}_m\big(X,f(X),g(X)\big) \equiv 0$, then by Lemma 3.5 it can be written in the form

$$\mathcal{A}(X,Y,Z)\big(Y - f(X)\big) + \mathcal{B}(X,Y,Z)\big(Z - g(X)\big)$$

for some polynomials $\mathcal{A}(X,Y,Z)$, $\mathcal{B}(X,Y,Z)$ in $\mathbb{F}_q[X,Y,Z]$. Therefore $P(Y,Z)$ can be written in the form

$$P(Y,Z) \;=\; \widetilde{\mathcal{A}}(Y,Z)\big(Y-\beta\big) \,+\, \widetilde{\mathcal{B}}(Y,Z)(Z-\gamma) \tag{3.21}$$

where $\widetilde{\mathcal{A}}(Y,Z)$ and $\widetilde{\mathcal{B}}(Y,Z)$ in $\mathbb{K}[Y,Z]$ are the remainders of $\mathcal{A}(X,Y,Z)$ and $\mathcal{B}(X,Y,Z)$ upon division by $e(X)$. It is now obvious from (3.21) that $P(\beta,\gamma)=0$. ∎

Here is the crucial part of our decoding algorithm. From the encoding rule (3.14), we know *a priori* that $\gamma = \beta^a$ in $\mathbb{K}$. Hence $P(\beta,\gamma)=0$ implies that β is a root of the polynomial $H(Y) = P(Y,Y^a)$. But $H(Y)$ is a univariate polynomial, so all of its roots can be found using well-known techniques. First, however, we should make sure that $H(Y) \not\equiv 0$.

Lemma 3.9. *The polynomial $H(Y) = P(Y,Y^a)$ is not the all-zero polynomial.*

Proof. Assume to the contrary that $H(Y) = P(Y,Y^a) \equiv 0$. This happens if and only if $Z - Y^a$ is a factor of $P(Y,Z)$. Clearly, this cannot be the case if $\deg_Y P(Y,Z)$ is smaller than a. But it follows from (3.18) and (3.5) that

$$\deg_Y P(Y,Z) \;\leqslant\; \frac{{}_{\mathbf{w}}\!\deg\, \mathcal{Q}_m(X,Y,Z)}{k-1} \tag{3.22}$$

Lemma 3.3 provides an upper bound on ${}_{\mathbf{w}}\!\deg\, \mathcal{Q}_m(X,Y,Z)$, and our condition for a in (3.13) uses this bound to guarantee that the value of a is strictly greater than the RHS of (3.22). ∎

Based upon Lemmas 3.6, 3.7, 3.8, and 3.9, we can now summarize the entire decoding algorithm as follows:

$\boxed{1}$ Given a vector $(v_1,v_2,\ldots,v_n)$ over $\mathbb{F}_{q^2}$, set up the interpolation set $\mathcal{P}$ and compute the interpolation polynomial $\mathcal{Q}_m(X,Y,Z)$ in $\mathbb{F}_q[X,Y,Z]$.

$\boxed{2}$ Compute $P(Y,Z) = \mathcal{Q}_m(X,Y,Z) \mod e(X)$, interpreted as an element of $\mathbb{K}[Y,Z]$.

$\boxed{3}$ Compute the univariate polynomial $H(Y) = P(Y,Y^a)$.

$\boxed{4}$ Find all the roots of $H(Y)$ in $\mathbb{K}$, and output this list.

It should be obvious that this algorithm runs in polynomial time. The only potential quibble is at Step 4, where we are required to find roots of a polynomial over $\mathbb{K}$, which is an exponentially large field. Note, however, that while the order of $\mathbb{K}$ is q^k its characteristic is still equal to that of $\mathbb{F}_q$. Shoup [Sho91] gives a deterministic algorithm for factoring polynomials over large finite fields of small characteristic. Using this algorithm, factoring a polynomial of degree D over a field of order p^K (for a prime p) takes only

$$\widetilde{O}\big(D^2K^2\big)\log p \;+\; \widetilde{O}\big(D^{3/2}K^{3/2}\big)\log p\sqrt{p}$$

operations in $\mathbb{F}_p$. Thus there are no problems with Step 4 as well. The following theorem uses all of the above to establish Theorem 3.1 for the special case of $M = 2$.

Theorem 3.10. *Given a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ over $\mathbb{F}_{q^2}$, the algorithm above outputs in polynomial time the list of all codewords of $\mathbb{C}$ that differ from $\boldsymbol{v}$ in at most*

$$t \;=\; \left\lfloor n - n\sqrt[3]{4R^2\Big(1+\frac{1}{m}\Big)\Big(1+\frac{2}{m}\Big)} - \frac{1}{m}\right\rfloor \tag{3.23}$$

positions, where $m \geqslant 1$ is an arbitrary multiplicity parameter. The size of this list is at most L^2, where

$$L \;=\; \left\lceil m\sqrt[3]{\frac{(1+\frac{1}{m})(1+\frac{2}{m})}{2R}}\;\right\rceil + 1 \tag{3.24}$$

Proof. The expression for t in (3.23) follows immediately from (3.17), when taking into account that $R = k/2n$. The list-size is obviously bounded by the degree of $H(Y)$. It is clear that $\deg H(Y) \leqslant a\,\deg_{\text{tot}} P(Y,Z)$, where $\deg_{\text{tot}}$ is the total degree. But $\deg_{\text{tot}} P(Y,Z)$ is bounded by the RHS of (3.22), whence (3.24) now follows by Lemma 3.3 and (3.13).

In general, we will always choose a to be just larger than the RHS of (3.22). Hence, the list size is essentially bounded by the square of

$$a \simeq {}_{\mathbf{w}}\!\deg \mathcal{Q}_m(X,Y,Z)/(k-1). \;\blacksquare$$

3.4 General trivariate codes and their decoding

We now show that the codes constructed in the foregoing section are, in fact, just one member of a much more general family of codes. All the codes in this family can be list decoded in essentially the same way as the codes of Section 3.3.

3.4.1 Code parameters and encoder mapping

To the list of code parameters detailed in Section 3.3.1, we now add one more ingredient:

Encoder polynomial: A polynomial $\mathcal{T}(Y,Z) \in \mathbb{K}[Y,Z]$ of total degree a, that is irreducible over $\mathbb{K}$ and has positive degree in both Y and Z, but is otherwise arbitrary.

We then generalize the encoder mapping of Section 3.3.1 as follows. Let $\mathscr{X}$ be the set of zeros of $\mathcal{T}(Y,Z)$ in $\mathbb{K}^2$, namely

$$\mathscr{X} \overset{\text{def}}{=} \{\, (\beta,\gamma) \in \mathbb{K} \times \mathbb{K} \,:\, \mathcal{T}(\beta,\gamma) = 0 \,\} \tag{3.25}$$

We shall assume for the time being (but see the remark at the end of this section) that there is a way to enumerate the rational points of $\mathcal{T}(Y,Z)$ in polynomial time. That is, we assume that there is a bijection from $\{1,2,\ldots,|\mathscr{X}|\}$ to $\mathscr{X}$ that can be computed in time polylog $|\mathscr{X}|$. We can then think of an encoder for our code as a mapping $\mathcal{E}\colon \mathscr{X} \to \mathbb{C}$ from rational points of $\mathcal{T}(Y,Z)$ to codewords.

We now describe this mapping. Given a $(\beta,\gamma) \in \mathscr{X}$, we first write β and γ as

$$\beta = f_0 + f_1\zeta + \cdots + f_{k-1}\zeta^{k-1}, \quad \text{with } f_i \in \mathbb{F}_q \tag{3.26}$$

$$\gamma = g_0 + g_1\zeta + \cdots + g_{k-1}\zeta^{k-1}, \quad \text{with } g_i \in \mathbb{F}_q \tag{3.27}$$

where ζ is a zero of $e(X)$ in $\mathbb{K}$ and, hence, $\{1,\zeta,\ldots,\zeta^{k-1}\}$ is a polynomial basis for $\mathbb{K}$ over $\mathbb{F}_q$. Then $f_0, f_1, \ldots, f_{k-1}$ and $g_0, g_1, \ldots, g_{k-1}$ define polynomials $f(X)$ and $g(X)$ in $\mathbb{F}_q[X]$ of degree at most $k-1$. Henceforth, the encoder proceeds as in (3.15). Namely, the codeword $(c_1, c_2, \ldots, c_n) \in \mathbb{C}$ corresponding to the rational point (β, γ) is given by

$$c_j \;=\; f(x_j) + \alpha g(x_j) \qquad \text{for } j = 1, 2, \ldots, n \tag{3.28}$$

It should be obvious that the codes constructed in Section 3.3 are just a special case of the above, which results by taking $\mathcal{T}(Y, Z)$ as the absolutely irreducible polynomial $Z - Y^a$.

Do we gain anything from this generalization? As we shall see, the error-correction radius of the general codes is still given by (3.17). But their rate is no longer simply $k/2n$.

Proposition 3.11. *Let $\mathfrak{g}$ denote the genus of $\mathcal{T}(Y, Z)$, and assume that $\mathcal{T}(Y, Z)$ is absolutely irreducible. Then the rate R of the code $\mathbb{C}$ defined by* (3.25) – (3.28) *is bounded as follows*

$$R \geqslant \frac{k}{2n} \;+\; \frac{1}{2n}\left(\log_q\!\left(1 - \frac{2\mathfrak{g}}{q^{k/2}}\right) - \log_q a\right) \tag{3.29}$$

$$R \leqslant \frac{k}{2n} \;+\; \frac{1}{2n}\left(\log_q\!\left(1 + \frac{2\mathfrak{g}}{q^{k/2}}\right)\right) \tag{3.30}$$

Proof. Clearly, the rate of $\mathbb{C}$ is given by $R = \log_{q^2}|\mathscr{X}|/n$. By the Hasse-Weil bound [Has49], the number of rational points of $\mathcal{T}(Y, Z)$, counted with multiplicity, is bounded by

$$\Big|\#\,\text{zeros}[\mathcal{T}](\mathbb{K}) \;-\; (1 + q^k)\Big| \;\leqslant\; 2\mathfrak{g}\sqrt{q^k} \tag{3.31}$$

Since the total degree of $\mathcal{T}(Y, Z)$ is a, the multiplicity of any rational point of $\mathcal{T}(Y, Z)$ is at most a. The proposition now follows from (3.31), upon straightforward manipulations. ▮

What Proposition 3.11 tells us is that, in principle, it should be possible to improve upon the simple trivariate codes of Section 3.3 by choosing an irreducible polynomial $\mathcal{T}(Y, Z)$ with many rational points. This would yield codes with the same error-correction radius whose rate is higher than $k/2n$. But not by much. For example, in the (reasonable) regime where $n = q$, $k = \Theta(n)$, and $m = O(1)$, it follows from Proposition 3.11 and (3.13) that $R \to k/2n$ as $n \to \infty$, *regardless of the choice* of the encoder polynomial $\mathcal{T}(Y, Z)$.

3.4.2 General trivariate interpolation decoding

The decoding commences exactly as before: we set up the interpolation set $\mathcal{P}$, compute the corresponding interpolation polynomial $\mathcal{Q}_m(X,Y,Z)$, then reduce it modulo $e(X)$ to obtain the polynomial $P(Y,Z) \in \mathbb{K}[Y,Z]$. In other words, Steps 1 and 2 of the decoding algorithm of Section 3.3.2 remain unchanged, and Lemmas 3.6, 3.7, 3.8 still apply.

In view of Lemma 3.8, we are now interested in all pairs $(\beta,\gamma) \in \mathbb{K}^2$ such that $P(\beta,\gamma) = 0$. However, as suggested by (3.31), there could be an *exponential number* of such pairs, at least in certain cases. The key to our decoding algorithm is, again, the *a priori* relationship between β and γ embedded into the encoder. We know *a priori* that $\mathcal{T}(\beta,\gamma) = 0$, so (β,γ) is a solution to the following system of equations

$$P(Y,Z) = 0 \quad \text{and} \quad \mathcal{T}(Y,Z) = 0 \tag{3.32}$$

The standard way to solve such a system of polynomial equations is to use *resultants*. Thus we compute

$$H(Y) \overset{\text{def}}{=} \operatorname{Res}(P,\mathcal{T};Z) \tag{3.33}$$

The resultant of two polynomials is the determinant of their Sylvester matrix; for a precise definition of the resultant operation $\operatorname{Res}(\cdot,\cdot;Z)$ see e.g. [CLO96, p.158]. We will furthermore make use of the following lemma from [CLO96, Chapter 3].

Lemma 3.12. *Let $P(Y,Z)$ and $\mathcal{T}(Y,Z)$ be arbitrary polynomials over a field $\mathbb{K}$, both having positive degree in Z, and let $H(Y) = \operatorname{Res}(P,\mathcal{T};Z)$. Then*

a. *The polynomial $H(Y)$ belongs to the ideal generated by $P(Y,Z)$ and $\mathcal{T}(Y,Z)$. That is, there exist polynomials $\mathcal{A}(Y,Z)$ and $\mathcal{B}(Y,Z)$ in $\mathbb{K}[Y,Z]$ such that*

$$H(Y) = \mathcal{A}(Y,Z)P(Y,Z) + \mathcal{B}(Y,Z)\mathcal{T}(Y,Z) \tag{3.34}$$

b. *The polynomial $H(Y)$ is the all-zero polynomial if and only if the polynomials $P(Y,Z)$, $\mathcal{T}(Y,Z)$ have a common factor in $\mathbb{K}[Y,Z]$ which has positive degree in Z.*

Lemma 3.13. *The resultant polynomial $H(Y)$ given by* (3.33) *is not the all-zero polynomial. Moreover, if (β, γ) is a solution of the system* (3.32)*, then $H(\beta) = 0$.*

Proof. This follows easily from Lemma 3.12. Strictly speaking, in order to apply this lemma, we need to make sure that $P(Y, Z)$ has positive degree in Z. But we can assume this w.l.o.g., otherwise compute the resultant in (3.33) with respect to Y rather than Z. Now, the fact that $H(\beta) = 0$ follows by evaluating both sides of (3.34) at the point (β, γ) and using (3.32). Since $\mathcal{T}(Y, Z)$ is irreducible, it follows from Lemma 3.12.b that $H(Y) \equiv 0$ if and only if $\mathcal{T}(Y, Z)$ *divides* $P(Y, Z)$. Our condition for a in (3.13) again guarantees that $\deg_{\text{tot}}\mathcal{T}(Y, Z) > \deg_{\text{tot}}P(Y, Z)$, so this cannot happen. ∎

Given a $\beta \in \mathbb{K}$, we can find all $\gamma \in \mathbb{K}$ such that (β, γ) is a solution of (3.32) by computing the set of common roots of the univariate polynomials $P(\beta, Z)$ and $\mathcal{T}(\beta, Z)$. To summarize, decoding the general trivariate codes of Section 3.4.1 amounts to modifying the last two steps of the decoding algorithm of Section 3.3.2 as follows:

$\boxed{3'}$ Compute the resultant $H(Y) = \text{Res}(P, \mathcal{T}; Z) \in \mathbb{K}[Y]$.

$\boxed{4'}$ Find all the roots of $H(Y)$ in $\mathbb{K}$. For each such root β, find all $\gamma \in \mathbb{K}$ such that $P(\beta,\gamma) = 0$ and $\mathcal{T}(\beta,\gamma) = 0$. Output the resulting list of pairs (β, γ).

The error-correction radius of this algorithm is still given by (3.17) and (3.23), and the size of the resulting list is still bounded by a^2, where a is given by (3.13). The latter observation follows from Bézout's Theorem [CLO96, p. 420]: since $P(Y, Z)$ and $\mathcal{T}(Y, Z)$ have no common factors, the number of solutions to (3.32) is at most the product of their total degrees.

Remark. Observe that $\text{Res}(P, \mathcal{T}; Z) = P(Y, Y^a)$ in the special case $\mathcal{T}(Y, Z) = Z - Y^a$. Thus the decoding algorithm of Section 3.3.2 is just an instance of the general algorithm above. While there are no problems with decoding, the encoder assumes the existence of a polynomial-time algorithm for mapping information into rational points of $\mathcal{T}(Y, Z)$. In general, finding such an algorithm is *hard*. So far, we are not aware of any choices for $\mathcal{T}(Y, Z)$ that yield codes of rate $R > k/2n$ $\underline{\text{and}}$ come equipped with an efficient encoder.

3.5 Extension to multivariate interpolation

We finally discuss the general case $M \geqslant 2$, where the encoder uses $M-1$ irreducible polynomials $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{M-1}$. While, in principle, these polynomials can be essentially arbitrary (cf. Section 3.4), we will only consider the special case where $\mathcal{T}_i(Y, Z_i) = Z_i - Y^{a_i}$ for all i (cf. Section 3.3).

Code parameters and encoder mapping. Our code $\mathbb{C}$ will now be a subset of $\mathbb{F}_Q^n$, where $Q = q^M$. We modify the list of code parameters in Section 3.3.1 as follows:

Basis: A fixed basis $\{1, \alpha_1, \ldots, \alpha_{M-1}\}$ for $\mathbb{F}_Q$ over $\mathbb{F}_q$.

Exponentiation degrees: A sequence of $M-1$ positive integers $a_1, a_2, \ldots, a_{M-1}$ given by

$$a_i \overset{\text{def}}{=} \sum_{j=0}^{i} \left\lceil \sqrt[M+1]{\frac{n}{k-1} \prod_{\ell=0}^{M} (m+\ell)} \right\rceil^{j} \tag{3.35}$$

All other parameters remain unchanged. Given k arbitrary symbols $u_0, u_1, \ldots, u_{k-1} \in \mathbb{F}_q$, the encoder first constructs the polynomial $f(X) = \sum_{i=0}^{k-1} u_i X^i$, and then computes

$$g_i(X) = \big(f(X)\big)^{a_i} \mod e(X) \tag{3.36}$$

for $i = 1, 2, \ldots, M-1$. The codeword $(c_1, c_2, \ldots, c_n) \in \mathbb{C}$ corresponding to the message $u_0, u_1, \ldots, u_{k-1}$ is given by

$$c_j = f(x_j) + \sum_{i=1}^{M-1} \alpha_i g_i(x_j) \qquad \text{for } j = 1, 2, \ldots, n \tag{3.37}$$

It is obvious from the above description that the rate of $\mathbb{C}$ is $R = \log_Q |\mathbb{C}|/n$ which is equal to k/Mn. The minimum distance of $\mathbb{C}$ is at least $n-k+1$, since $\mathbb{C}$ is again a subset of a Reed-Solomon code of length n and dimension k over $\mathbb{F}_Q$.

Multivariate Decoding. Given a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ over $\mathbb{F}_Q$, we decompose each v_j into its components over $\mathbb{F}_q$ using the basis $\{1, \alpha_1, \ldots, \alpha_{M-1}\}$. Thus we write

$$v_j = y_j + \sum_{i=1}^{M-1} \alpha_i z_{i,j} \qquad \text{for } j = 1, 2, \ldots, n$$

where y_j and $z_{i,j}$ are in $\mathbb{F}_q$. We then set up the interpolation set $\mathcal{P}$ consisting of the n points $(x_j, y_j, z_{1,j}, \ldots, z_{M-1,j})$. As before, the first decoding step consists of computing the *interpolation polynomial* $\mathcal{Q}_m(X,Y,Z_1,\ldots,Z_{M-1})$, defined as the least $(1,k-1,k-1,\ldots,k-1)$-weighted degree polynomial that has a zero of multiplicity m at each point of $\mathcal{P}$.

Lemma 3.14. *Suppose that a codeword* $\boldsymbol{c} \in \mathbb{C}$ *differs from the given vector* $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ *in at most*

$$t = \left\lfloor n - n \sqrt[M+1]{\left(\frac{k-1}{n}\right)^{M}\left(1+\frac{1}{m}\right)\cdots\left(1+\frac{M}{m}\right)} - \frac{1}{m} \right\rfloor$$

positions, and let $f(X), g_1(X), g_2(X), \ldots, g_{M-1}(X)$ *denote the message polynomial(s) that produce* $\boldsymbol{c}$ *according to the mapping* (3.36) – (3.37). *Then the interpolation polynomial satisfies* $\mathcal{Q}_m\big(X, f(X), g_1(X), \ldots, g_{M-1}(X)\big) \equiv 0$.

Proof. The key observation is that the weighted-degree of the interpolation polynomial is at most

$$\Delta_M \overset{\text{def}}{=} \left\lceil \sqrt[M+1]{n(k-1)^M \prod_{\ell=0}^{M}(m+\ell)} \right\rceil \tag{3.38}$$

Since the number of interpolation constraints is now given by $n(M+m)!/(M+1)!(m-1)!$, this follows from a simple geometric argument as in Lemma 3.3. The rest is an easy generalization of Lemma 3.4 and Lemma 3.6. ∎

Next, as in (3.18), we reduce the interpolation polynomial mod $e(X)$ to obtain the polynomial $P(Y, Z_1, \ldots, Z_{M-1})$ in the ring $\mathbb{K}[Y, Z_1, \ldots, Z_{M-1}]$, where $\mathbb{K}$ is defined by (3.16). Then exactly the same argument as in Lemma 3.7 shows that $P(Y, Z_1, \ldots, Z_{M-1})$ is not the all-zero polynomial. Moreover, under the conditions of Lemma 3.14, we have

$$P(\beta, \gamma_1, \ldots, \gamma_{M-1}) = P(\beta, \beta^{a_1}, \ldots, \beta^{a_{M-1}}) = 0 \tag{3.39}$$

where β and $\gamma_1, \gamma_2 \ldots, \gamma_{M-1}$ are the elements of $\mathbb{K}$ corresponding to $f(X)$ and to $g_1(X), g_2(X), \ldots, g_{M-1}(X)$, respectively. We omit the proof of (3.39), since it is a straightforward generalization of the arguments in Lemma 3.5 and Lemma 3.8. The third decoding step consists of computing

$$H(Y) \stackrel{\text{def}}{=} P(Y, Y^{a_1}, Y^{a_2}, \ldots, Y^{a_{M-1}}) \tag{3.40}$$

The fourth and final decoding step is exactly as before: find all the roots of $H(Y)$ in $\mathbb{K}$, and output this list. It is obvious from (3.39) that, under the conditions of Lemma 3.14, β is indeed a root of $H(Y)$. There is only one thing left to prove.

Lemma 3.15. *The polynomial $H(Y)$ defined in* (3.40) *is not the all-zero polynomial.*

Proof. Consider the M polynomials $P_0, P_1, \ldots, P_{M-1}$ defined as follows: P_i is a polynomial in the $M - i$ variables $Y, Z_{i+1}, Z_{i+2}, \ldots, Z_{M-1}$ defined by the recursive rule

$$P_i(Y, Z_{i+1}, \ldots, Z_{M-1}) \stackrel{\text{def}}{=} P_{i-1}(Y, Y^{a_i}, Z_{i+1}, \ldots, Z_{M-1})$$

To initialize this recursion, set $P_0 = P(Y, Z_1, \ldots, Z_{M-1})$. $P(Y, Z_1, \ldots, Z_{M-1})$ is the interpolation polynomial modulo $e(X)$, as above. It is then clear that $H(Y) = P_{M-1}$. To establish the lemma, we will show by induction on i that none of the polynomials $P_0, P_1, \ldots, P_{M-1}$ is the all-zero polynomial. As induction hypothesis, assume that P_{i-1} is non- zero. As noted above (see also Lemma 3.7), this is indeed true for $i = 1$. By induction hypothesis, P_i is the all-zero polynomial if and only if $Z_i - Y^{a_i}$ is a factor of P_{i-1}. Thus it would suffice to show that $a_i > \deg_{\text{tot}} P_{i-1}$. Observe that

$$\deg_{\text{tot}} P_{i-1} \leqslant \max\{a_1, a_2, \ldots, a_{i-1}\} \deg_{\text{tot}} P_0 \tag{3.41}$$

and since $a_1 < a_2 < \cdots < a_{M-1}$ in view of (3.35), it follows that $\deg_{\text{tot}} P_{i-1} \leqslant a_{i-1} \deg_{\text{tot}} P_0$. Thus it remains to show that $a_i > a_{i-1} \deg_{\text{tot}} P_0$. But $\deg_{\text{tot}} P_0 \leqslant \Delta_M/(k-1)$, as in Lemma 3.9, where Δ_M is defined by (3.38). The induction step now follows by observing that $a_i/a_{i-1} > \Delta_M/(k-1)$. ∎

As a by-product of the proof of Lemma 3.15, we also get an upper bound on the degree of $H(Y)$. Applying (3.41) for the special case $i - 1 = M - 1$, we find that

$$\deg H(Y) \leqslant a_{M-1} \deg_{\text{tot}} P_0 \leqslant \frac{\Delta_M a_{M-1}}{k-1} \tag{3.42}$$

Since the list-size of our decoder is obviously bounded by $\deg H(Y)$, this completes the proof of Theorem 3.1. Using the fact that $R = k/Mn$, the expression for the error-correction radius t follows readily from Lemma 3.14 while the expression for the list-size L follows from (3.35), (3.38), and (3.42).

3.6 Explicit capacity achieving codes

This section is based on the results of Guruswami and Rudra [GR06c,GR06b] on decoding of *folded Reed-Solomon* codes. We pretty much follow the structure of [GR06b] here.

In Chapter 2, we observed that multivariate interpolation decoding can decode beyond the $1 - \sqrt{R}$ bound when errors are random. However, the provable performance of the algorithm coincides with the $1 - \sqrt{R}$ bound under adversarial model. The key obstacle in improving this bound is the following: for the case when the messages are pairs of $f(X)$ and $g(X)$ polynomials with degree $< k$, two algebraically independent relations is needed in the adaptive recovery algorithm for decoding. The interpolation polynomial only provides one such relation. So we used several interpolation polynomials in Section 2.5 to decode, but there is no guarantee that these interpolation polynomials are algebraically independent.

Then, in Chapter 3, we used the idea of obtaining the extra algebraic relation by enforcing it as an *a priori* condition satisfied at the encoder. Specifically, instead of letting the second polynomial $g(X)$ to be an arbitrarily polynomial of degree $< k$, we make it correlated with $f(X)$ by a specific algebraic condition, such as $g(X) = f(X)^a \mod E(X)$ for some integer a and an irreducible polynomial $E(X)$ of degree k.

The modification of using algebraic relation at the encoder does not come for free. In particular, since we send at least twice as much information as in the original RS code, there is no way to construct codes with rate more than ½. The central idea of Guruswami and Rudra is to avoid this rate loss by making the correlated polynomial $g(X)$ essentially identical to $f(X)$ (say $g(X) = f(\gamma X)$). 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 in the encoding. The folded RS encoding of $f(X)$ compresses all the needed information, without extra redundancy for $g(X)$.

3.6.1 Folded Reed-Solomon codes

Guruswami and Rudra use a simple variant of Reed-Solomon codes called *folded Reed-Solomon* codes for which, by choosing parameters suitably, they can decode close to the optimal fraction of $1 - R$ of errors.

Consider a Reed-Solomon code $\mathcal{C}(n,k)$ consist of evaluations of degree $< k$ polynomials over $\mathbb{F}_q$. Let $q = n + 1$ and γ be a primitive element of $\mathbb{F}_q$. Assume that the evaluation points are ordered as $1, \gamma, \gamma^2, \ldots, \gamma^{n-1}$. Notice that in construction of Reed-Solomon codes usually all the nonzero elements of $\mathbb{F}_q$ is being used. Let $S \leqslant 1$ be an integer parameter called the *folding parameter*. We assume that S divides $n = q - 1$. Also let $\{\beta_0, \beta_1, \ldots, \beta_{S-1}\}$ be a fixed basis of $\mathbb{F}_q^S$ over $\mathbb{F}_q$.

Definition 3.1. *The S-folded version of the Reed-Solomon code $\mathcal{C}(n,k)$ is a code of block length $N = n/S$ over $\mathbb{F}_q^S$. The encoding of a message $f(X)$, a polynomial of degree $< k$ over $\mathbb{F}_q$, is $(c_0, c_1, \ldots, c_{n/S-1}) \in \mathbb{F}_{q^S}^N$ where*

$$c_j \overset{\text{def}}{=} \sum_{i=0}^{S-1} \beta_i f(\gamma^{jS+i}), \qquad \text{for } j = 0, 1, \ldots, \frac{n}{S} - 1$$

In other words, the codewords of the folded Reed-Solomon code are in one-to-one correspondence with $\mathcal{C}(n,k)$ and are obtained by bundling together consecutive S-tuples of symbols in codewords of $\mathcal{C}$.

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

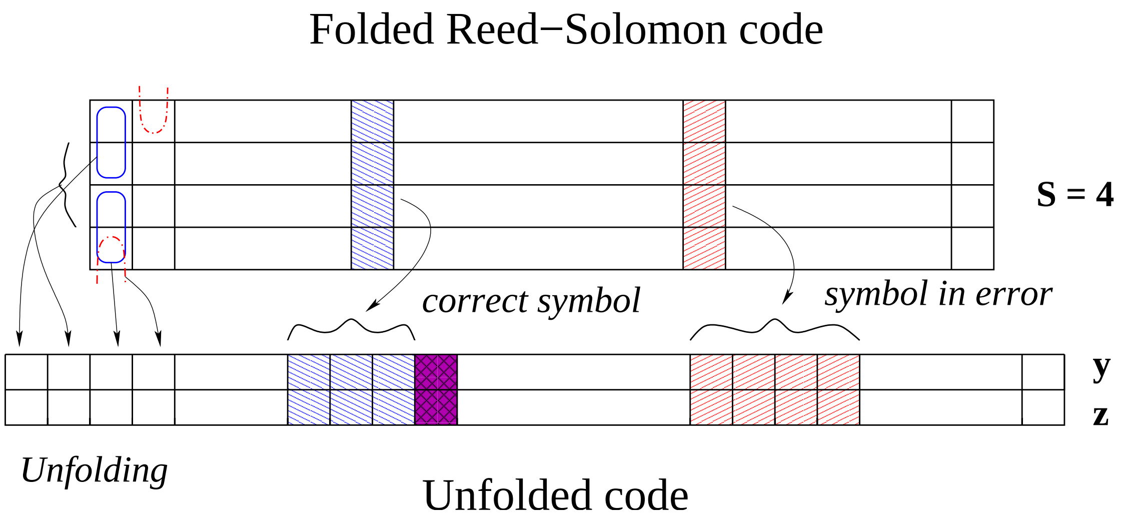


Figure 3.1: Unfolding a folded Reed-Solomon code

3.6.2 Using trivariate interpolation for folded RS codes

Let us see how trivariate interpolation can be used in the context of decoding the folded Reed-Solomon code. Given a received word $\boldsymbol{v} = (v_1, v_2, \ldots, v_N)$ in $(\mathbb{F}_q^S)^N$, we construct $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{F}_q^n$ to be the corresponding "unfolded" received word. (Formally, let j-th symbol of $\boldsymbol{v}$ be $(v_{j,0}, v_{j,1}, \ldots, v_{j,S-1})$ for $j = 0, 1, \ldots, N-1$. Let vector $\boldsymbol{y} \in \mathbb{F}_q^n$ to be the corresponding "unfolded" received word. (Specifically, let $y_{jS+\ell}$ to be equal to $\boldsymbol{v}_{j,\ell}$ for $j = 0, 1, \ldots, N-1$ and $\ell = 0, 1, \ldots, S-1$.) We also, define the vector $\boldsymbol{z}$ to be the cyclic shift of $\boldsymbol{y}$ to the left, formally, $z_i := y_{i+1}$ for $i = 0, 1, \ldots, n$ (with the convention $y_0 = y_n$).

The obvious but very important fact after unfolding is the following: if t symbols of $\boldsymbol{v} \in \mathbb{F}_{q^S}^N$ are not in error then at least $(S-1)t$ pairs of (y_i, z_i) are also not in error for $i = 0, 1, \ldots, n-1$ of the unfolded received word (cf. Figure 3.1). Suppose, $\boldsymbol{v}$ is the received word of the encoded version of $f(X)$. Define $g(X) = f(\gamma X)$. Now if we consider the n triples (γ^i, y_i, z_i) then for at least $(S-1)t$ triples we have $f(\gamma^i) = y_i$ and $g(\gamma^i) = z_i$. This suggest that, using Lemma 3.3, there is a trivariate polynomial $\mathcal{Q}(X, Y, Z)$ that passes through

$$\mathcal{P} = \left\{ (\gamma^0, y_0, z_0), (\gamma^1, y_1, z_1), \cdots, (\gamma^{n-1}, y_{n-1}, z_{n-1}) \right\}$$

with multiplicity m and has $(1, k-1, k-1)$-weighted degree smaller than or equal to

$$\text{w}\deg Q(X,Y,Z) \leqslant \left\lceil \sqrt[3]{n(k-1)^2 m(m+1)(m+2)} \right\rceil$$

Lemma 3.16. *Suppose that a codeword of the folded Reed-Solomon code $\mathbf{c} \in \mathbb{F}_{q^S}^N$ differs from the given vector $\mathbf{v} \in \mathbb{F}_{q^S}^N$ in at most*

$$t = \left\lfloor N - N\frac{S}{S-1}\sqrt[3]{R^2\left(1+\frac{1}{m}\right)\left(1+\frac{2}{m}\right)} - \frac{1}{m(S-1)} \right\rfloor \tag{3.43}$$

position, and let $f(X)$ denote the message polynomial that produces $\mathbf{c}$. Then the interpolation polynomial satisfies $Q(X, f(X), g(X)) \equiv Q(X, f(X), f(\gamma X)) \equiv 0$.

Proof. The proof simply follows from the Lemma 3.4 and Lemma 3.16 and the fact that $N = Sn$. ∎

Lemma 3.17. *Let γ to be a primitive element of $\mathbb{F}_q$ and define $E(X) \overset{\text{def}}{=} X^{q-1} - \gamma$. Then we have the following two facts:*

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

2) *For every polynomial $f(X) \in \mathbb{F}_q[X]$ of degree $< q-1$ we have*

$$f(\gamma X) = f(X)^q \mod E(X).$$

Proof. Suppose $E(X)$ is not irreducible and some irreducible polynomial $h(X)$ in $\mathbb{F}_q[X]$ of degree b, $1 \leqslant b < q-1$, divides $E(X)$. Let ξ be a root of $h(X)$ in the extension field $\mathbb{F}_{q^b}$. We have $\xi^{q^b-1} = 1$. Also, $h(\xi) = 0$ implies $\xi^{q-1} = \gamma$. Thus, $\gamma^{\frac{q^b-1}{q-1}} = 1$. γ is a primitive element of $\mathbb{F}_q$ and $\gamma^\alpha = 1$ iff α divides $q-1$. We conclude that $q-1$ must divide $1 + q + \cdots + q^{b-1}$. This is, however, impossible since $1 + q + \cdots + q^{b-1} \equiv b \mod q-1$ and $0 < b < q-1$. This contradiction proves that $E(X)$ has no factor of degree less than $q-1$, and therefore is irreducible. For a precise characterization of all irreducible polynomials of the form $X^a - c$, see for instance [LN86, Chapter 3, Section 5].

For the second part, we use the identity $f(X)^q = f(X^q)$ for all polynomials in $\mathbb{F}_q[X]$. Thus, $f(X)^q - f(\gamma X) = f(X^q) - f(\gamma X)$. The latter polynomial is divisible by $X^q - \gamma X$ and thus also by $X^{q-1} - \gamma$. Hence, $f(X)^q \equiv f(\gamma X) \mod E(X)$. Degree of $f(\gamma X)$ is less than $q-1$ and that completes the proof. ▮

Lemma 3.18. *Let γ to be a primitive element of $\mathbb{F}_q$. There is a deterministic algorithm that for a nonzero polynomial $\mathcal{Q}(X,Y,Z) \in \mathbb{F}_q[X,Y,Z]$ of degree less than q in Y, outputs a list of all polynomials $f(X)$ of degree $< k$ that satisfying the condition $\mathcal{Q}(X, f(X), f(\gamma X)) \equiv 0$. The algorithm runs in polynomial time in q.*

Proof. Let $E(X) = X^{q-1} - \gamma$. We know from Lemma 3.17 that $E(X)$ is irreducible. We first divide out the largest power of $E(X)$ that divides $\mathcal{Q}(X,Y,Z)$ to obtain $\mathcal{Q}_0(X,Y,Z)$ where $\mathcal{Q}(X,Y,Z) = E(X)^b \mathcal{Q}_0(X,Y,Z)$. Clearly, if $f(X)$ satisfies $\mathcal{Q}(X, f(X), f(\gamma X)) \equiv 0$ then it also satisfies $\mathcal{Q}_0(X, f(X), f(\gamma X)) \equiv 0$. Now define $T(Y,Z) \in \mathbb{K}[Y,Z]$ to be $\mathcal{Q}_0(X,Y,Z) \mod E(X)$, where $\mathbb{K} \overset{\text{def}}{=} \mathbb{F}_q[X]/E(X)$. Also, let α to denote the representation of $f(X)$ in $\mathbb{K}$. Therefore, for any $f(X)$ we know $f(\gamma X) = f(X)^q \mod E(X)$ and so $f(\gamma X)$ will be represented as α^q in $\mathbb{K}$. Also, for any $f(X)$ that satisfies $\mathcal{Q}(X, f(X), f(\gamma X)) \equiv 0$ we have $T(\alpha, \alpha^q) = 0$. Define $H(Y) = T(Y, Y^q)$. Then, for all the α's that $T(\alpha, \alpha^q)$ is zero $H(\alpha)$ becomes zero as well. Degree of $H(Y)$ is at most dq where d is the total degree of $\mathcal{Q}(X,Y,Z)$. The characteristic of $\mathbb{K}$ is at most q, and its degree over the base field is at most $q \log q$. Therefore, we can find all roots of $H(Y)$ in $\mathbb{K}$ by deterministic algorithm running in time polynomial in d, q [Ber70]. 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 $f(X)$ with degree $< k$ that satisfy $\mathcal{Q}(X, f(X), f(\gamma X)) \equiv 0$

Notice that $H(Y)$ is a nonzero polynomial since $H(Y) \equiv 0$ iff $Z - Y^q$ divides $T(Y,Z)$, and this cannot happen as degree of $T(Y,Z)$ in Y is less than q. ▮

Theorem 3.19. *For every $\varepsilon > 0$ and R, $0 < R < 1$, there is a family of S-folded Reed-Solomon codes for $S = O(1/\varepsilon)$ that have rate at least R and can be list* *decoded up to a fraction of* $1-(1+\varepsilon)R^{2/3}$ *of errors in time polynomial in the block length and* $1/\varepsilon$.

Given that trivariate interpolation improves the decoding radius 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. Here we just spell out the final result. An interested reader can check [GR06c, GR06b] for the details.

Theorem 3.20. *For every* $\varepsilon > 0$*, integer* $M \geqslant 1$ *and* $0 < R < 1$*, there is a family of* S*-folded Reed-Solomon codes for* $S = O(M/\varepsilon)$ *that have rate at least* R *and can be list decoded up to a fraction of* $1-(1+\varepsilon)R^{M/(M+1)}$ *of errors in time* $(NS)^{O(M)}(1/\varepsilon)^{O(1)}$ *where* N *is the block length of the code. The alphabet size of the code as a function of block length* N *is* $(NS)^{O(S)}$.

In the limit of large M, the decoding radius approaches the list decoding radius of $1-R$.

Theorem 3.21. *For every* $\varepsilon > 0$ *and* $0 < R < 1$*, there is a family of folded Reed-Solomon codes that have rate at least* R *an can be list decoded up to a fraction of* $1-R-\varepsilon$ *of errors in time* $(N/\varepsilon)^{O(1/\varepsilon)}$ *where* N *is the block length of the code. The alphabet size of the code as a function of the block length* N *is* $(N/\varepsilon)^{O(1/\varepsilon^2)}$.

The material of Chapter 3 has been presented, in full, at the 46th *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Parvaresh, Farzad; Vardy, Alexander. The dissertation author was the primary investigator and author of this paper.

CHAPTER 4

Multiplicity assignment

A soft-decision decoding algorithm for Reed-Solomon codes, based upon the algebraic interpolation and factorization techniques of Sudan [GS99,Sud97], was recently proposed by Koetter and Vardy [KV03a]. The key to the soft-decoding algorithm of [KV03a] is the conversion of probabilities observed at the channel output into algebraic interpolation conditions, specified in terms of a multiplicity matrix M. Koetter and Vardy [KV03a] show that the probability of decoding failure is given by $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$, where $\mathcal{S}_M$ is a random variable whose distribution depends on the channel observations and $\Delta(M)$ is a known function of M. They then derive an efficient algorithm to compute the multiplicity matrix M that maximizes the *expected value* of $\mathcal{S}_M$. Here, we attempt to *directly minimize* the probability of decoding failure $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$.

First, we recast this optimization problem into a geometrical framework and deduce the salient properties of the resulting geometric structure. Using this geometric insight, we propose a simple modification to the Koetter-Vardy algorithm which results in a provably better multiplicity assignment M.

Finding the multiplicities that minimizes the probability of failure in general is a hard task. Instead, we use Chebyshev bound to set an upper bound on the probability of failure $\Pr\{S_M \leqslant \Delta(M)\}$ and we minimize the upper bound. This leads us to an iterative and analytical algorithms to minimize approximately $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$. This approach leads to coding gains of up to 0.20 dB for RS codes of length 255 and up to 0.75 dB for RS codes of length 15, as compared to the Koetter-Vardy algorithm.

Finally we show how the multiparities can be computed for the finite case by using both the geometrical framework theorems and the Chebyshev upper bound on the probability of failure.

4.1 Introduction

The possibility of *weighted* interpolation was mentioned by Guruswami and Sudan in [GS99], however the question of how such interpolation weights (or *multiplicities*) should be chosen was not discussed. This problem has been later investigated in a number of papers. Nielsen [Nie03] studies the setting of interpolation multiplicities in the context of concatenated codes so as to maximize the overall Hamming weight of a correctable error pattern. Pecquet [Pec02, Pec01] considers the so-called λ-distance and s-similarity, and then proposes a logarithmic multiplicity assignment. The problem of assigning interpolation multiplicities so as to maximize the cost of a correctable error pattern with respect to a general additive cost structure (which includes [Nie03, Pec02] as special cases) was introduced and studied in [KV02].

An entirely different approach to the problem of determining interpolation multiplicities was taken by Koetter and Vardy in [KV03a]. The goal of the multiplicity assignment scheme of [KV03a] is to maximize the *probability* of correct decoding. Koetter and Vardy [KV03a] show that, for a given multiplicity assignment M, this probability is essentially given by $\Pr\{\mathcal{S}_M > \Delta(M)\}$, where $\mathcal{S}_M$ is a random variable (called the *score*), whose distribution depends on the channel output, and $\Delta(M)$ is a known deterministic function of M. The goal, then, is to compute M so as to maximize $\Pr\{\mathcal{S}_M > \Delta(M)\}$. This goal is achieved in [KV03a] only in the asymptotic sense. Koetter and Vardy [KV03a] derive an efficient multiplicity assignment scheme that maximizes the *mean* of $\mathcal{S}_M$ for a fixed $\Delta(M)$. They moreover show that as the length n of the code becomes large, the random variable $\mathcal{S}_M$ concentrates about its mean. Thus for $n \to \infty$,

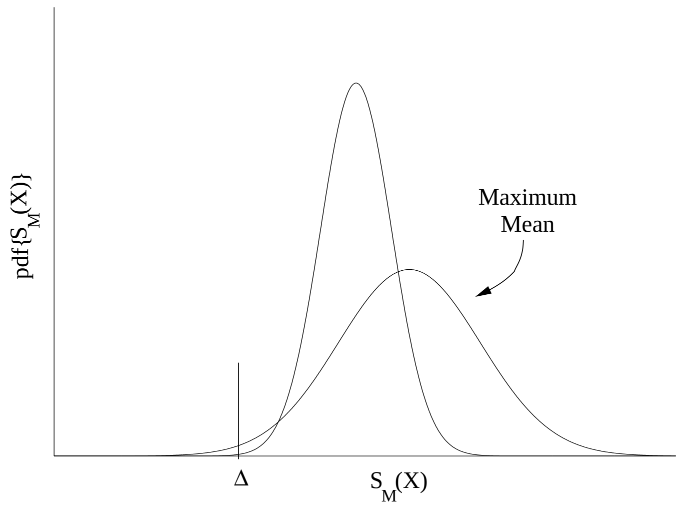


Figure 4.1: Score distributions for a fixed n

maximizing the mean of $\mathcal{S}_M$ is the optimal strategy. In particular, for $n \to \infty$, this allows transmission with an arbitrarily small probability of error at a maximum possible rate [KV03a].

However, for each fixed n, maximizing the mean of $\mathcal{S}_M$ (the *expected score*) may be suboptimal. This situation is illustrated in Figure 4.1: the score distribution on the right represents the Koetter-Vardy [KV03a] multiplicity assignment, whereas the distribution on the left corresponds to a different multiplicity assignment, with a smaller expected score, which nevertheless leads to a higher probability of correct decoding.

Our goal here is to devise multiplicity assignments based on directly minimizing the probability of decoding failure $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$. To this end, we first recast the problem of minimizing $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$ into a geometrical framework. While we are as yet unable to solve the corresponding geometric problem, we establish several key properties of the resulting geometric structure. Using the insight thereby gained, we propose a simple modification to the Koetter-Vardy algorithm which produces a provably better multiplicity assignment. The results are summarized in Section 4.3.

A completely different approach is taken in Section 4.4. We find an upper bound on the probability of failure using the Chebyshev bound. Then, instead of minimizing the actual probability of failure of the decoder, we minimize the upper bound on the probability of failure. Under this second-order approximation, we assume

$$\operatorname*{argmin}_M \Pr\{\mathcal{S}_M \leqslant \Delta(M)\} \;\simeq\; \operatorname*{argmax}_M \frac{\mu(\mathcal{S}_M) - \Delta(M)}{\sigma(\mathcal{S}_M)}$$

where $\mu(\mathcal{S}_M)$ is the mean and $\sigma(\mathcal{S}_M)$ is the standard deviation of $\mathcal{S}_M$. We develop an iterative algorithm that solves the optimization. The iterative algorithm is computationally intensive than the Koetter-Vardy algorithm and/or the modification thereof developed in Section 4.3. However, it also results in higher coding gains: up to 0.20 dB for RS codes of length 255 and up to 0.75 dB for RS codes of length 15, as compared to the Koetter-Vardy algorithm. Finally, we derive an analytical solution for the second-order approximation optimization problem in high SNR cases which has almost the same complexity as that of Koetter-Vardy but has the same performance as compare to iterative algorithm. (see Figures 4.5 and 4.4).

In Section 4.5, we use both ideas developed in Section 4.3 and Section 4.4 to solve the optimization problem for the finite cost. The idea is to transform the finite cost optimization into infinite cost optimization. Using the theorems from geometrical framework we show that these two optimization are equivalent. Then, we exploit the ideas of the Section 4.4 to solve the new optimization problem. We see that the finite cost optimization is essentially the same as infinite cost optimization subject to an extra condition on the summation of multiplicities.

4.2 Background and notation

We first recall some of the notation and auxiliary results from [KV03a], that will be used throughout this chapter. Let $\mathbb{F}_q$ be the finite field with q elements.

A Reed-Solomon code of length n and dimension k over $\mathbb{F}_q$ is defined by

$$\mathbb{C}_q(n,k) \overset{\text{def}}{=} \{\, (f(x_1), f(x_2), \ldots, f(x_n)) \;:\; x_1, x_2, \ldots, x_n \text{ distinct elements of } \mathbb{F}_q,\ f(X) \in \mathbb{F}_q[X],\ \deg f(X) < k \,\Big\}$$

Given a bivariate polynomial $\mathcal{A}(X,Y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} a_{i,j} X^i Y^j$ over $\mathbb{F}_q$ and integers w_X, w_Y, the (w_X, w_Y)-***weighted degree*** of $\mathcal{A}(X,Y)$ is defined as $\deg_{w_X,w_Y} \mathcal{A}(X,Y) = \max_{a_{i,j} \neq 0}\{i w_X + j w_Y\}$. Let $N_{w_X,w_Y}(\delta)$ denote the number of monomials $X^i Y^j$ of (w_X, w_Y)-weighted degree at most δ, and define

$$\Delta_{w_X,w_Y}(\nu) \overset{\text{def}}{=} \min \{\, \delta \in \mathbb{Z} \;:\; N_{w_X,w_Y}(\delta) > \nu \,\}$$

It is shown in [GS99,KV03a] that $N_{1,k-1}(\delta) > \delta^2/(k-1)$, and therefore $\Delta_{1,k-1}(\nu) \leqslant \sqrt{2(k-1)\nu}$. Furthermore, for large δ, we have $N_{1,k-1}(\delta) \simeq \delta^2/(k-1)$, so that $\Delta^2_{1,k-1}(\nu) \simeq 2(k-1)\nu$ for large ν. Given a $q \times n$ matrix $M = [m_{i,j}]$, we define

$$\mathcal{C}(M) \overset{\text{def}}{=} \sum_{i=1}^{q} \sum_{j=1}^{n} \frac{m_{i,j}(m_{i,j}+1)}{2} \tag{4.1}$$

If M is a multiplicity assignment matrix, then $m_{i,j} \geqslant 0$ and $\mathcal{C}(M)$ is called the ***cost of*** M. The function $\Delta(M)$ that plays a crucial role in soft-decision decoding of Reed-Solomon codes (see Section 4.1) is given by

$$\Delta(M) \overset{\text{def}}{=} \Delta_{1,k-1}(\mathcal{C}(M)) = \min \{\, \delta \in \mathbb{Z} \;:\; N_{1,k-1}(\delta) > \mathcal{C}(M) \,\} \simeq \sqrt{2(k-1)\mathcal{C}(M)} \tag{4.2}$$

where the approximation becomes valid for large costs. Next, we need to define the *score* $\mathcal{S}_M$. To this end, it will be convenient to identify vectors in $\mathbb{F}_q^n$ with $q \times n$ matrices over the reals. Specifically, let us list the elements of $\mathbb{F}_q$ in some arbitrary, but fixed, order $\alpha_1, \alpha_2, \ldots, \alpha_q$. Then a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{F}_q^n$ can be represented by the $q \times n$ real-valued matrix $[\boldsymbol{v}]$, defined as follows: $[\boldsymbol{v}]_{i,j} = 1$ if $v_j = \alpha_i$, and $[\boldsymbol{v}]_{i,j} = 0$ otherwise. We will often think of $q \times n$ matrices over $\mathbb{R}$ simply as vectors in $\mathbb{R}^{qn}$. In particular, we define the inner product of two such matrices A and B as

$$\langle A, B \rangle \overset{\text{def}}{=} \sum_{i=1}^{q} \sum_{j=1}^{n} a_{i,j} b_{i,j} = \mathsf{trace}(AB^T)$$

Now, the ***score*** of a vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ over $\mathbb{F}_q$ with respect to a given multiplicity matrix M can be defined as the inner product $\mathcal{S}_M(\boldsymbol{v}) = \langle M, [\boldsymbol{v}] \rangle$. The following theorem is one of the key results of [GS99, KV03a, Sud97].

Theorem 4.1. *Let M be a given $q \times n$ multiplicity assignment matrix. Then there exists a polynomial-time decoding algorithm for $\mathbb{C}_q(n,k)$ that computes a list of all codewords $\boldsymbol{c} \in \mathbb{C}_q(n,k)$ such that $\mathcal{S}_M(\boldsymbol{c}) > \Delta(M)$.*

In view of Theorem 4.1, the remaining task of the decoder is to compute the matrix M that maximizes the score of the transmitted codeword, subject to the constraint $\Delta(M) = \Delta$. However, the transmitted codeword is unknown to the decoder, only some probabilistic information about it is available through the channel observations. Thus one should think of the transmitted codeword as a random vector $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n)$, and the score of this codeword is a random variable

$$\mathcal{S}_M \overset{\text{def}}{=} \langle M, [\underline{\boldsymbol{\mathcal{X}}}] \rangle \tag{4.3}$$

For Reed-Solomon codes, the random variables $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n$ at the channel input are *a priori* uniform over $\mathbb{F}_q$, and we assume, as in [KV03a], that they are independent (in fact, they are not, since $\boldsymbol{\mathcal{X}}$ is drawn *a priori* from $\mathbb{C}_q(n,k)$ rather than uniformly from $\mathbb{F}_q^n$; however, as shown in [KV03a], taking this into account leads to an intractable optimization problem). Let $\mathscr{Y}$ denote the channel output alphabet, let $\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_n \in \mathscr{Y}$ denote the random variables at the channel output, and let $(y_1, y_2, \ldots, y_n) \in \mathscr{Y}^n$ be the specific channel output vector observed by the decoder. Then, for a memoryless channel, the *a posteriori* distribution of the random vector $\boldsymbol{\mathcal{X}}$ is given by

$$\Pr\{\boldsymbol{\mathcal{X}} = (x_1, x_2, \ldots, x_n)\} = \prod_{j=1}^{n} \Pr(\mathcal{X}_j = x_j | \, \mathcal{Y}_j = y_j) \tag{4.4}$$

where $(x_1, x_2, \ldots, x_n)$ ranges over $\mathbb{F}_q^n$. For $i = 1, 2, \ldots, q$ and $j = 1, 2, \ldots, n$, we define $\pi_{i,j} = \Pr(\mathcal{X}_j = \alpha_i | \, \mathcal{Y}_j = y_j)$ and call the $q \times n$ matrix $\Pi = [\pi_{i,j}]$ the ***reliability matrix***. Observe that the *a posteriori* distribution of the score $\mathcal{S}_M$ is completely determined by Π and M. The decoder is given the reliability matrix

Π by the channel, but is free to choose the multiplicity matrix M. The question is: given Π, how should M be chosen so as to minimize $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$?

4.3 Geometrical framework

Our objective in this section is to recast the optimization problem above into a geometrical framework. To this end, let us think of the $q \times n$ multiplicity matrix M as a point in the qn-dimensional Euclidean space $\boldsymbol{\mathcal{V}} = \mathbb{R}^{qn}$. In this space, we assume that the multiplicity matrix could get not only integer values but also real values. This assumption relaxes the problem of multiplicity assignment form *integer value* assignment into *real value* multiplicity assignment. In Section 4.5, how to modify the real value multiplicity matrix into an integer value multiplicity matrix. Given a positive integer $\mathcal{C}$, what is the set of all points $M \in \boldsymbol{\mathcal{V}}$ such that $\mathcal{C}(M) = \mathcal{C}$? It turns out that this set has a simple characterization.

Theorem 4.2. *The set of all points $M \in \boldsymbol{\mathcal{V}}$ with a given constant cost $\mathcal{C}(M) = \mathcal{C}$ is a sphere of radius $r_{\mathcal{C}} \overset{\text{def}}{=} \sqrt{2\mathcal{C} + \frac{qn}{4}}$ centered about the point $(-1/2, \ldots, -1/2) \in \boldsymbol{\mathcal{V}}$.*

Proof. It follows from the definition (4.1) that $\mathcal{C}(M)$ is equal to $\mathcal{C}$ if and only if $\sum_{i=1}^{q} \sum_{j=1}^{n} \left(m_{i,j} + 1/2\right)^2 = 2\mathcal{C} + \frac{qn}{4}$. ▮

We denote the sphere in Theorem 4.2 by $\mathscr{S}_{\mathcal{C}}$ and call it the **cost sphere**. For a given $\mathcal{C} \in \mathbb{Z}^+$, let $\Delta = \Delta_{1,k-1}(\mathcal{C})$. Then it is obvious from (4.2) that $\Delta(M) = \Delta$ for all $M \in \mathscr{S}_{\mathcal{C}}$. Thus the constraint $\Delta(M) = \Delta$ is equivalent to $M \in \mathscr{S}_{\mathcal{C}}$ in $\boldsymbol{\mathcal{V}}$.

Next, we derive a generating function for the probability distribution of the score $\mathcal{S}_M$, for a given multiplicity assignment matrix $M = [m_{i,j}]$ and reliability matrix $\Pi = [\pi_{i,j}]$. Define $\mathcal{G}(T) = \prod_{j=1}^{n} \mathcal{G}_j(T)$, where

$$\mathcal{G}_j(T) \overset{\text{def}}{=} \pi_{1,j}T^{m_{1,j}} + \pi_{2,j}T^{m_{2,j}} + \cdots + \pi_{q,j}T^{m_{q,j}} = \sum_{i=1}^{q} \pi_{i,j}T^{m_{i,j}}, \text{ for } j = 1, 2, \ldots, n \tag{4.5}$$

Theorem 4.3. *Let the reliability matrix Π and the multiplicity matrix M be* *given. Then for all numbers a, the* a posteriori *probability* $\Pr\{\mathcal{S}_M = a\}$ *is equal to the coefficient of* T^a *in* $\mathcal{G}(T)$.

Proof. Given $\alpha_i \in \mathbb{F}_q$, let $M(\alpha_i, j)$ denote the entry in row i and column j of M, that is $M(\alpha_i, j) = m_{i,j}$. We can then define the random variables $\mathcal{S}_j = M(\mathcal{X}_j, j)$ for $j = 1, 2, \ldots, n$. The polynomial $\mathcal{G}_j(T)$ in (4.5) is precisely the generating function for the probability distribution of $\mathcal{S}_j$. It is easy to see from (4.4) that the random variables $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$ are independent, and it follows from (4.3) that $\mathcal{S}_M = \mathcal{S}_1 + \mathcal{S}_2 + \cdots + \mathcal{S}_n$. ∎

In general, the product $\mathcal{G}(T) = \prod_{j=1}^n \mathcal{G}_j(T)$ has q^n additive terms[1], one for each $\boldsymbol{x} \in \mathbb{F}_q^n$. These terms are of the form $p_{\boldsymbol{x}}\, T^{\langle M, [\boldsymbol{x}]\rangle}$, where $p_{\boldsymbol{x}} = e^{\langle \ln \Pi, [\boldsymbol{x}]\rangle}$ (here, $\ln \Pi$ denotes the $q \times n$ matrix $[\ln \pi_{i,j}]$, with the understanding that $\ln 0 = -\infty$ and $e^{-\infty} = 0$). To summarize all of the above, we have

$$\mathcal{G}(T) \;=\; \prod_{j=1}^{n} \mathcal{G}_j(T) \;=\; \sum_a \Pr\{\mathcal{S}_M = a\}\, T^a \;=\; \sum_{\boldsymbol{x}\,\in\,\mathbb{F}_q^n} p_{\boldsymbol{x}}\, T^{\langle M, [\boldsymbol{x}]\rangle} \tag{4.6}$$

Observe that the coefficients $p_{\boldsymbol{x}}$ in (4.6) are completely determined by the reliability matrix Π, so that the choice of the multiplicity assignment M affects only the corresponding exponents $T^{\langle M, [\boldsymbol{x}]\rangle}$. In light of (4.6), we now assign a ***value*** $\varphi_\Delta(M)$ to each point $M \in \boldsymbol{\mathcal{V}}$. The value function $\varphi_\Delta : \boldsymbol{\mathcal{V}} \to [0, 1]$ is defined as follows:

$$\varphi_\Delta(M) \;\overset{\text{def}}{=}\; \sum_{\boldsymbol{x}\,\in\,\mathbb{F}_q^n} p_{\boldsymbol{x}}\, \mathcal{I}_\Delta(M, \boldsymbol{x}) \tag{4.7}$$

where the indicator function $\mathcal{I}_\Delta(M, \boldsymbol{x})$ is defined by $\mathcal{I}_\Delta(M, \boldsymbol{x}) = 1$ if $\langle M, [\boldsymbol{x}]\rangle > \Delta$, and $\mathcal{I}_\Delta(M, \boldsymbol{x}) = 0$ otherwise. Combining the derivation of (4.6) with the definition of φ_Δ in (4.7), we have the following result.

Theorem 4.4. *Let the reliability matrix* Π *and an integer* $\Delta > 0$ *be given. Then* $\Pr\{\mathcal{S}_M > \Delta\} = \varphi_\Delta(M)$ *for all* $M \in \boldsymbol{\mathcal{V}}$.

[1]In practice, only η_j entries in the j-th column of Π are nonzero, where η_j is usually a small number — most often $\eta_j = 1$ or $\eta_j = 2$. Thus the number of *nonzero* additive terms in $\mathcal{G}(T)$ is $\prod_{j=1}^n \eta_j$, which is usually less than $2^n \ll q^n$.

Thus our goal is to find a point M on the cost sphere $\mathscr{S}_{\mathcal{C}}$ that maximizes the value $\varphi_\Delta(M)$ for $\Delta = \Delta_{1,k-1}(\mathcal{C})$. Note that the value function $\varphi_\Delta(M)$ partitions the space $\boldsymbol{\mathcal{V}}$ into s convex regions $R_1, R_2, \ldots, R_s$ (where s is at most 2^{q^n}), with $\varphi_\Delta(M)$ being constant on each region. To see this, observe that $\langle M, [\boldsymbol{x}] \rangle$ is just a summation of some n coordinates of the point M, so the set of all points in $\boldsymbol{\mathcal{V}}$ that satisfy $\langle M, [\boldsymbol{x}] \rangle > \Delta$ is a half-space with the hyperplane

$$\mathcal{H}_{\boldsymbol{x}} \overset{\text{def}}{=} \{\, M \in \boldsymbol{\mathcal{V}} \;:\; \langle M, [\boldsymbol{x}] \rangle \;=\; M(x_1, 1) + M(x_2, 2) + \cdots + M(x_n, n) \;=\; \Delta \,\} \tag{4.8}$$

as its boundary. For a given hyperplane $\mathcal{H}_{\boldsymbol{x}}$, we say that a point M lies above the hyperplane (or, equivalently, $\mathcal{H}_{\boldsymbol{x}}$ lies under M) if $\langle M, [\boldsymbol{x}] \rangle > \Delta$. Conversely, if $\langle M, [\boldsymbol{x}] \rangle \leqslant \Delta$, we say that M is under $\mathcal{H}_{\boldsymbol{x}}$. Now let us assign to each hyperplane $\mathcal{H}_{\boldsymbol{x}}$ a value equal to $p_{\boldsymbol{x}}$. Then the value $\varphi_\Delta(M)$ of a point $M \in \boldsymbol{\mathcal{V}}$ is precisely the sum of the values of those hyperplanes $\mathcal{H}_{\boldsymbol{x}}$ that lie under M. Clearly, the hyperplanes $\mathcal{H}_{\boldsymbol{x}}$ partition $\boldsymbol{\mathcal{V}}$ into at most 2^{q^n} convex regions, with $\varphi_\Delta(M)$ being constant on each region.

In the following, we drive some of the properties of geometrical framework. In Proposition 4.5 we show that in large-cost asymptotic case the framework is invariance of scaling. With Lemma 4.6, we mark a *singular* point in space $\boldsymbol{\mathcal{V}}$ which is the intersection of all the hyperplanes in this space and by Lemma 4.7 and Lemma 4.9 we distinguish movements in space $\boldsymbol{\mathcal{V}}$ that always keep or increase the value of points conducted by these movements. These two lemmas are bases of the main theorem of this section, Theorem 4.10, which reduces the searching space of multiplicity assignment over cost sphere to a set which has lower dimension than cost sphere.

Proposition 4.5 *In asymptotic cases, the geometrical framework is invariance of scaling, meaning that* $\varphi_{\Delta(\lambda M)}(\lambda M) = \varphi_{\Delta(M)}(M)$.

Proof. In the large-cost asymptotic case $\Delta(\lambda M) = \sqrt{(k-1)\,\langle \lambda M, \lambda M \rangle} = \lambda \Delta(M)$. Thus $\langle \lambda M, [\boldsymbol{x}] \rangle > \Delta(\lambda M)$ if and only if $\langle M, [\boldsymbol{x}] \rangle > \Delta(M)$. It follows that $\Pr\{\mathcal{S}_{\lambda M} \leqslant \Delta(\lambda M)\} = \Pr\{\mathcal{S}_M \leqslant \Delta M\}$, in view of (4.8) and Theorem 4.4. In other word only the *direction* of the matrix M in the space $\boldsymbol{\mathcal{V}}$, and not the norm, is relevant for the large costs. ▮

Lemma 4.6. *All the hyperplanes $\mathcal{H}_{\boldsymbol{x}}$ intersect at the point $\xi = (\frac{\Delta}{n}, \frac{\Delta}{n}, \ldots, \frac{\Delta}{n}) \in \boldsymbol{\mathcal{V}}$.*

Proof. Clearly $\langle \xi, [\boldsymbol{x}] \rangle = \Delta$ for all $\boldsymbol{x} \in \mathbb{F}_q^n$, which proves the lemma. ▮

Lemma 4.7. *Let a point $M \in \boldsymbol{\mathcal{V}}$ be given, and consider the ray $\mathcal{R}_M \overset{\text{def}}{=} \{\lambda M + (1-\lambda)\xi \,:\, \lambda \in [0, \infty) \subset \mathbb{R}\}$, starting at ξ and passing through M. Then for all points $M_\lambda \in \mathcal{R}_M \backslash \{\xi\}$, we have $\varphi_\Delta(M_\lambda) = \varphi_\Delta(M)$.*

Proof. Fix an $\boldsymbol{x} \in \mathbb{F}_q^n$. We assume there exist points $P_1, P_2 \in \mathcal{R}_M \backslash \{\xi\}$ such that $\langle P_1, [\boldsymbol{x}] \rangle > \Delta$ and $\langle P_2, [\boldsymbol{x}] \rangle \leqslant \Delta$, then show a contradiction. Since $\langle M_\lambda, [\boldsymbol{x}] \rangle = \lambda \langle M, [\boldsymbol{x}] \rangle + (1-\lambda) \langle \xi, [\boldsymbol{x}] \rangle$ is a continuous function of λ, there must also exist a point $P_3 \in \mathcal{R}_M \backslash \{\xi\}$ such that $\langle P_3, [\boldsymbol{x}] \rangle = \Delta$. Thus $P_3 \in \mathcal{R}_M \cap \mathcal{H}_{\boldsymbol{x}}$. But the point $M_0 = \xi$ belongs to $\mathcal{R}_M \cap \mathcal{H}_{\boldsymbol{x}}$ by Lemma 4.6, so that $\mathcal{R}_M$ intersects the hyperplane $\mathcal{H}_{\boldsymbol{x}}$ in (at least) two distinct points. Since $\mathcal{R}_M$ is one-dimensional, this implies that $\mathcal{R}_M \subset \mathcal{H}_{\boldsymbol{x}}$. It follows that *all* the points of $\mathcal{R}_M$ lie under $\mathcal{H}_{\boldsymbol{x}}$, which contradicts the assumed existence of $P_1 \in \mathcal{R}_M$ with $\langle P_1, [\boldsymbol{x}] \rangle > \Delta$. This proves that for each of the q^n hyperplanes $\mathcal{H}_{\boldsymbol{x}}$ in (4.8), either all the points of $\mathcal{R}_M \backslash \{\xi\}$ lie under $\mathcal{H}_{\boldsymbol{x}}$ or all lie above $\mathcal{H}_{\boldsymbol{x}}$. Hence, the value function φ_Δ is constant on $\mathcal{R}_M \backslash \{\xi\}$. ▮

Corollary 4.8 *All the partition regions $R_1, R_2, \ldots, R_s$ are open, meaning that for each $i = 1, 2, \ldots, s$, there exists a line from ξ to infinity that is wholly contained in R_i.*

Proof. For a given point in R_i, consider the ray that start from ξ and passes through M, $\mathcal{R}_M = \{\lambda M + (1-\lambda)\xi \,:\, \lambda \in [0, \infty) \subset \mathbb{R}\}$. In Lemma 4.7 we have shown that, either all the points of $\mathcal{R}_M \backslash \{\xi\}$ lie under $\mathcal{H}_{\boldsymbol{x}}$ or all lie above $\mathcal{H}_{\boldsymbol{x}}$ for each hyperplane $\mathcal{H}_{\boldsymbol{x}}$ in (4.8). Thus all the point of $\mathcal{R}_M \backslash \{\xi\}$ belongs to same partition as M. Also, the ray $\mathcal{R}_M$ goes to infinity as λ goes to infinity, which completes the proof. ▮

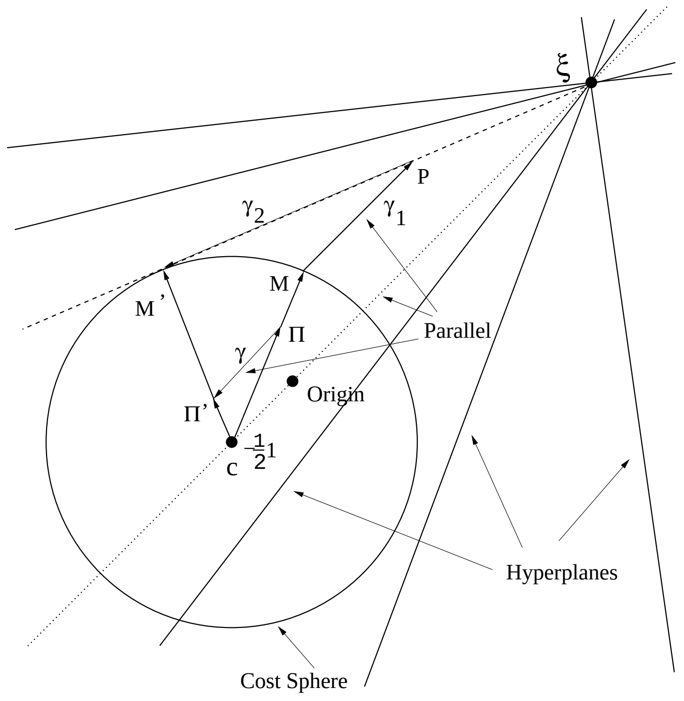


Figure 4.2: Geometrical framework

Lemma 4.9. *Let a point $M \in \boldsymbol{\mathcal{V}}$ be given, and consider the line $\mathcal{L}_M \overset{\text{def}}{=} \{M + \lambda\mathbf{1} : \lambda \in \mathbb{R}\}$ passing through M, where $\mathbf{1}$ denotes the vector $(1,1,\ldots,1) \in \boldsymbol{\mathcal{V}}$. Then the value $\varphi_\Delta(M{+}\lambda\mathbf{1})$ is monotonically nondecreasing with λ along this line.*

Proof. We need to show that $\varphi_\Delta(M{+}\lambda_1\mathbf{1}) - \varphi_\Delta(M{+}\lambda_2\mathbf{1}) \geqslant 0$ whenever $\lambda_1 > \lambda_2$. By the definition of φ_Δ in (4.7), we have:

$$\varphi_\Delta(M{+}\lambda_1\mathbf{1}) - \varphi_\Delta(M{+}\lambda_2\mathbf{1}) \;=\; \sum_{\boldsymbol{x} \in \mathbb{F}_q^n} p_{\boldsymbol{x}} \Big(\mathcal{I}_\Delta(M{+}\lambda_1\mathbf{1},\boldsymbol{x}) - \mathcal{I}_\Delta(M{+}\lambda_2\mathbf{1},\boldsymbol{x})\Big) \tag{4.9}$$

But if $\lambda_1 > \lambda_2$, then $\langle M{+}\lambda_1\mathbf{1},\boldsymbol{x}\rangle - \langle M{+}\lambda_2\mathbf{1},\boldsymbol{x}\rangle = (\lambda_1 - \lambda_2)n > 0$ for all $\boldsymbol{x} \in \mathbb{F}_q^n$. This implies that the the difference of the indicator functions $\mathcal{I}_\Delta$ on the right-hand side of (4.9) is always nonnegative, and the lemma follows. ▮

Theorem 4.10. *It is possible to move any given point M on the cost sphere* $\mathscr{S}_\mathcal{C}$ that does not lie on a tangent from ξ to $\mathscr{S}_\mathcal{C}$ *to a point $M' \in \mathscr{S}_\mathcal{C}$ such that $\varphi_\Delta(M') \geqslant \varphi_\Delta(M)$.*

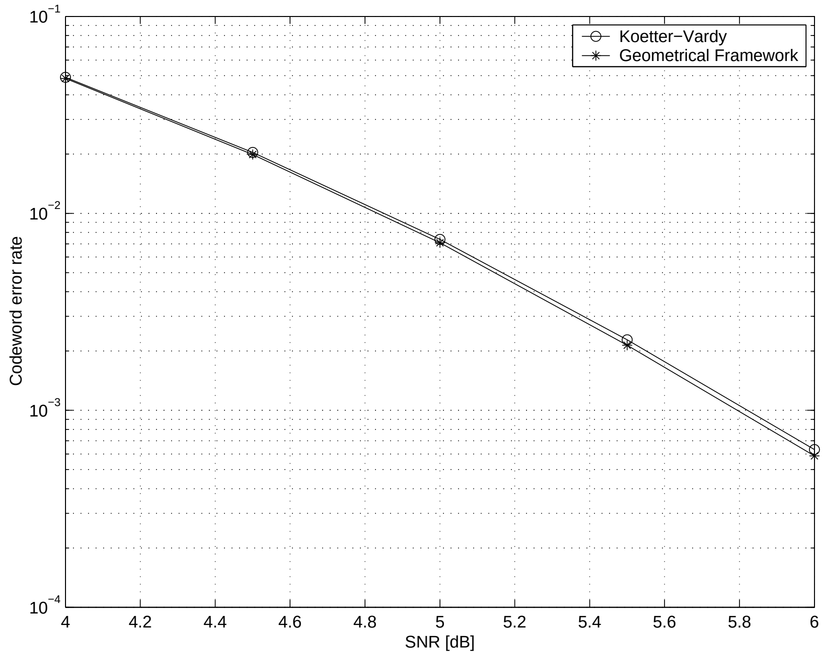


Figure 4.3: Performance of $\mathbb{C}_{16}(15, 11)$

Proof. For a given point $M \in \mathscr{S}_{\mathcal{C}}$, we move along the line $\mathcal{L}_M$ defined in Lemma 4.9. In the $2D$ plane that contains $\mathcal{L}_M$ and ξ, the tangent line from ξ to cost circle and $\mathcal{L}_M$ are not parallel, therefore $\mathcal{L}_M$ has intersection with the tangent line at a point P. We know $\Delta/n > -1/2$, so for some $\gamma_1 > 0$ we reach the point $P = M + \gamma_1\mathbf{1}$ on the tangent from ξ to $\mathscr{S}_{\mathcal{C}}$. If $M' \in \mathscr{S}_{\mathcal{C}}$ denotes the point of tangency, then $P = \gamma_2 M' + (1-\gamma_2)\xi$ for some $\gamma_2 \neq 0$. We can therefore move along the tangent ray $\mathcal{R}_{M'}$ back from P to the cost sphere $\mathscr{S}_{\mathcal{C}}$. By Lemma 4.7 and Lemma 4.9, we have $\varphi_\Delta(M') = \varphi_\Delta(P) \geqslant \varphi_\Delta(M)$, and if γ_1 is sufficiently large, then $\varphi_\Delta(M') > \varphi_\Delta(M)$. This transformation from M to M' is illustrated in Figure 4.2. ▮

Corollary 4.11. *Assume the multiplicity matrix is computed as $M = \lambda\Pi$ For a given vector $\Pi \in \boldsymbol{\mathcal{V}}$ and some $\lambda \in \mathbb{R}^+$. If the vector Π does not point toward the tangency point of cost sphere and ξ, then it is possible to offset Π by $\gamma\mathbf{1}$ such that the new vector $\Pi + \gamma\mathbf{1}$ points toward the tangency point, and by Theorem 4.10, for modified multiplicity assignment $M' = \lambda'(\Pi + \gamma\mathbf{1})$ we have*

$\varphi_\Delta(M') \geqslant \varphi_\Delta(M)$. *$\lambda'$ is chosen such that cost of M' to be equal to cost of M.*

Proof. For the proof check Figure 4.2. To compute value of γ we use the geometrical properties of the triangles $CM'\xi$ and $\Pi'C\Pi$. We get

$$\gamma = \sqrt{\frac{r_C^2}{d^2 - r_C^2}} \sqrt{\frac{\langle \Pi, \Pi \rangle}{qn} - \left(\frac{\langle \Pi, \mathbf{1} \rangle}{qn}\right)^2} - \frac{\langle \Pi, \mathbf{1} \rangle}{qn} - \frac{1}{2} \tag{4.10}$$

where $r_C = \sqrt{2C + qn/4}$ and $d = \sqrt{qn}(\Delta/n + 1\!/\!2)$. ▮

The Koetter-Vardy algorithm [KV03a] is equivalent to the following multiplicity assignment: $M_{\mathrm{KV}} = \lfloor \lambda \Pi \rfloor$ for a constant $\lambda \geqslant 1$ (see Lemma 16 of [KV03a]). In general, the point $\lfloor \lambda \Pi \rfloor$ is *not* on a tangent from ξ to the cost sphere. We can therefore improve the Koetter-Vardy multiplicity assignment by using result of Corollary 4.11. For Koetter-Vardy assignment $\langle \Pi, \mathbf{1} \rangle$ is equal to n. For large cost we have $\sqrt{\frac{r_C^2}{d^2 - r_C^2}} = \sqrt{\frac{n}{(k-1)q - n}}$. Thus, $M' = \lfloor \lambda(\Pi + \gamma \mathbf{1}) \rfloor$, where

$$\gamma \;=\; -\frac{1}{q}\left(1 - \sqrt{1 - \frac{1 - \frac{\langle \Pi, \Pi \rangle}{(k-1)}}{1 - \frac{n}{(k-1)q}}}\right) \tag{4.11}$$

While the improvement in performance is slight, Figure 4.3, it should be noted that it comes for "free". The computation required to find $M' = \lfloor \lambda(\Pi + \gamma \mathbf{1}) \rfloor$ is the same as that needed to compute $M_{\mathrm{KV}} = \lfloor \lambda \Pi \rfloor$. All we have to do is to add the constant γ in (4.10) to each $\pi_{i,j}$, and then (if desired) use the iterative algorithm of [KV03a].

4.4 Second-order approximation

We know that the best multiplicity assignment is the solution of the following optimization problem:

$$M^* = \operatorname*{argmin}_{M} \Pr\{\mathcal{S}_M > \Delta(M)\} \tag{4.12}$$

In this section, by using the Chebyshev bound, we find an upper bound on the $\Pr\{\mathcal{S}_M > \Delta(M)\}$ and then we try to minimize the upper bound instead of solving the actual optimization problem of (4.12). Working with the upper bound is computationally more tractable than trying to solve the actual optimization problem. From Chebyshev bound we know $\Pr\{|X - E[X]| > k\sigma[X]\} < 1/k^2$, where $E[X]$ is the expected value and $\sigma^2[X]$ is the variance of the random variable X. Thus for $\mathcal{S}_M$ we have

$$\Pr\{\mathcal{S}_M > \Delta(M)\} \leqslant \left(\frac{\sigma(\mathcal{S}_M)}{\Delta(M) - \mu(\mathcal{S}_M)}\right)^2 \tag{4.13}$$

where $\mu(\mathcal{S}_M)$ and $\sigma(\mathcal{S}_M)$ denote, respectively, the mean and the standard deviation of the score $\mathcal{S}_M$. Maximizing $\left(\frac{\sigma(\mathcal{S}_M)}{\Delta(M)-\mu(\mathcal{S}_M)}\right)^2$ is equivalent to minimizing $\left(\frac{\Delta(M)-\mu(\mathcal{S}_M)}{\sigma(\mathcal{S}_M)}\right)^2$, and in high SNR cases we know that $\mu(\mathcal{S}_M) > \Delta(M)$ so it becomes the same as maximizing

$$L(M) = \frac{\mu(\mathcal{S}_M) - \Delta(M)}{\sigma(\mathcal{S}_M)}. \tag{4.14}$$

The harder optimization problem of (4.12) is now transferred to a much easier optimization problem of

$$\widetilde{M}^* = \operatorname*{argmax}_M \frac{\mu(\mathcal{S}_M) - \Delta(M)}{\sigma(\mathcal{S}_M)} \tag{4.15}$$

We are expecting that the solution to (4.15) to be very close to (4.12) in high SNR cases. Simulation results show that the multiplicity assignment based on (4.15) improves upon Koetter-Vardy algorithm up to 0.2dB for decoding of Reed-Solomon codes of $\mathbb{C}(255, 239)$ for error rates around 10^{-10}.

Later on, in the papers [EM05,EMH04,RK05] authors show that by replacing the Chebyshev bound on the probability of failure with ***Chernoff bound*** we even can get multiplicity assignments with better coding gain. However, the complexity of the algorithms derived in [EM05, EMH04] is much higher than Koetter-Vardy algorithm or the one that we provide in the following section.

Before starting the section we just restate that it is established in [KV03a] that

$$\mu(\mathcal{S}_M) = \langle M, \Pi \rangle = \sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}\pi_{i,j} \tag{4.16}$$

$$\sigma^2(\mathcal{S}_M) = \sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}^2\pi_{i,j} - \sum_{j=1}^{n}\left(\sum_{i=1}^{q} m_{i,j}\pi_{i,j}\right)^2 \tag{4.17}$$

which eventually will be used on deriving the multiplicity assignment algorithms.

4.4.1 Iterative solution

We will pursue the maximization of $L(M)$ only for the asymptotic case when the cost of M is large. In this case, as explained in Section 4.2, we have $\Delta(M) \simeq \sqrt{2(k-1)\mathcal{C}(M)}$ and we can further simplify this to $\Delta(M) \simeq \sqrt{(k-1)\,\langle M, M\rangle}$ in view of (4.1). Combining this with (4.16) and (4.17) yields

$$L(M) \quad = \quad \frac{\sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}\pi_{i,j} \;-\; \sqrt{(k-1)\sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}^2}}{\sqrt{\sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}^2\pi_{i,j} - \sum_{j=1}^{n}\left(\sum_{i=1}^{q} m_{i,j}\pi_{i,j}\right)^2}} \tag{4.18}$$

Now our goal is to find $M = [m_{i,j}] \in \mathbb{R}^{qn}$ that maximizes the value of $L(M)$. Observe that in (4.18) if we replace $m_{i,j}$ by $\gamma m_{i,j}$ or M by γM where γ is a non-negative constant then $L(\gamma M) = L(M)$. In other words, the value of $L(M)$ is only a function of the direction of M in $\mathbb{R}^{qn}$ and not the norm of M. To find the optimum multiplicities we need the solution of the following system of equations

$$\frac{\partial}{\partial m_{i,j}} L(M) = 0 \qquad \text{for} \quad i = 1,2,\ldots,q \text{ and } j = 1,2,\ldots,n \tag{4.19}$$

After simplify the equations in (4.19) and using the fact that the norm of M is irrelevant in our optimization we get

$$m_{i,j} = \pi_{i,j}\,\frac{\sigma^2 - \mu_j(\mu - \sqrt{k-1}\,)}{\sigma^2\sqrt{k-1} - \pi_{i,j}(\mu - \sqrt{k-1}\,)} \tag{4.20}$$

$$\sum_{i=1}^{q}\sum_{j=1}^{n} m_{i,j}^2 = 1 \tag{4.21}$$

for $i = 1,2,\ldots,q$ and $j = 1,2,\ldots,n$. Here, μ and σ are basically $\mu(S_M)$ and $\sigma(S_M)$ defined in (4.16) and (4.17), and μ_j is defined as $\mu_j \overset{\text{def}}{=} \sum_{i=1}^{q} m_{i,j}\pi_{i,j}$. The equations (4.20) and (4.21) are nonlinear and we use iterative methods to find the solution.

Assume that the set of equations in (4.20) and (4.21) have a *fixed point*. We start with an initial value for $M = [m_{i,j}]$ for example the Koetter-Vardy assignment. Then we normalize the multiplicities by dividing each $m_{i,j}$ with

$\sqrt{\sum_{i=1}^{q}\sum_{j=1}^{n} m_{i,j}^2}$, and then we compute μ, σ and μ_j for $j = 1, 2, \ldots, n$. We plug in all the computed variables in to the RHS of (4.20) to get a new set of multiplicities. We repeat the procedure couple of times to get a stable set of solutions. Let say $M^{(0)} = [m_{i,j}^{(0)}], M^{(1)} = [m_{i,j}^{(1)}], \ldots, M^{(\ell)} = [m_{i,j}^{(\ell)}]$ are the multiplicities at the iteration $i = 0, 1, \ldots, \ell$. Also, let $\mu^{(s)}, \sigma^{(s)}$ and $\mu_j^{(s)}$ represent the value of μ, σ and μ_j for $j = 1, 2, \ldots, n$ at iteration s, respectively. Then the updates become as follow:

$$\widetilde{m}_{i,j} \;\leftarrow\; \pi_{i,j}\, \frac{\left(\sigma^{(s)}\right)^2 + \mu_j^{(s)}\left(\mu^{(s)} - \sqrt{k-1}\right)}{\left(\sigma^{(s)}\right)^2 \sqrt{k-1} \;-\; \pi_{i,j}\left(\sqrt{k-1} - \mu^{(s)}\right)} \tag{4.22}$$

$$m_{i,j}^{(s+1)} \;\leftarrow\; \frac{\widetilde{m}_{i,j}}{\sqrt{\sum_{j=1}^{n}\sum_{i=1}^{q}\left(\widetilde{m}_{i,j}\right)^2}}$$

for $s = 0, 1, \ldots, \ell - 1$. The initial value for the multiplicities can be taken as of normalized Koetter-Vardy multiplicity assignment:

$$m_{i,j}^{(0)} := \frac{\pi_{i,j}}{\sqrt{\sum_{i=1}^{q}\sum_{j=1}^{n}\pi_{i,j}^2}} \qquad \text{for } i = 1, 2, \ldots, q \text{ and } j = 1, 2, \ldots, n$$

It seems to be hard to prove that the iteration of (4.22) converge to a fixed point or the solution of (4.15). However, simulations results show that the iterative solution, indeed, performs better than Koetter-Vardy assignment for decoding of Reed-Solomon codes $\mathbb{C}(255, 239)$ over a BAWGN channel (cf. Figure 4.5).

In the iterative algorithm, at each iteration we compute $\mu^{(s)}$, $\sigma^{(s)}$ and $\mu_j^{(s)}$. That takes $3qn$ addition and multiplication over reals. In addition, the normalization part of iteration needs $2qn$ operation. To get a stable results for multiplicities usually more that 5 iteration is needed, so totally the complexity of the algorithm is $30qn$ operation over reals. This is at least 30 times more than Koetter-Vardy algorithm. In the next section we solve the optimization analytically and hence drive a much faster algorithm for multiplicity assignment.

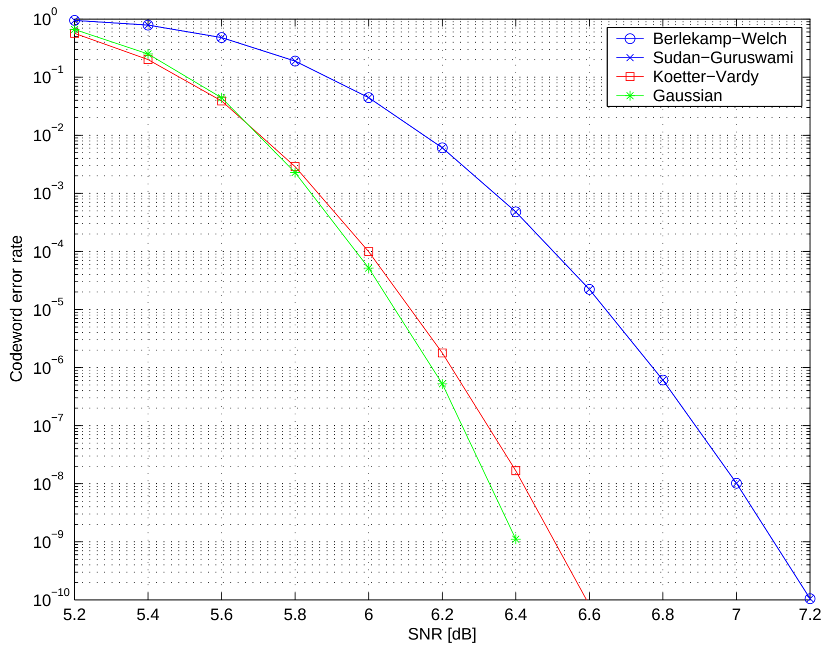


Figure 4.4: Performance of soft decoding of $\mathbb{C}(468, 420)$ over AWGN channel, when multiplicity of the interpolation tends to infinity.

4.4.2 Analytical solution

In order to solve the optimization problem of (4.15) in Section 4.4.1 we end up with a system of nonlinear equations (4.19). Here, by representing (4.18) in a matrix form, we are able to transform the system of equations to a diagonal form and so decouple the equations. That helps us to solve the equations analytically.

Although we have the solutions analytically but still they are hard to interpret. So, we use further approximations to simplify the solution. At high SNR, we know that the received symbol with high probability is either the most reliable symbol or second most reliable symbol. Thus, we just keep track of most reliable and second most reliable symbol for each received symbol at decoder and we assume that the reliability of the rest is zero. That reduces the dimension of reliability matrix from $q \times n$ to $2 \times n$ and so reduces the complexity by a factor of q. In addition, this assumption helps us to simplify the final solution.

To represent (4.18) in a matrix form it is more convenient to think of the reliability matrix and multiplicity matrix as vectors instead of matrices. So we

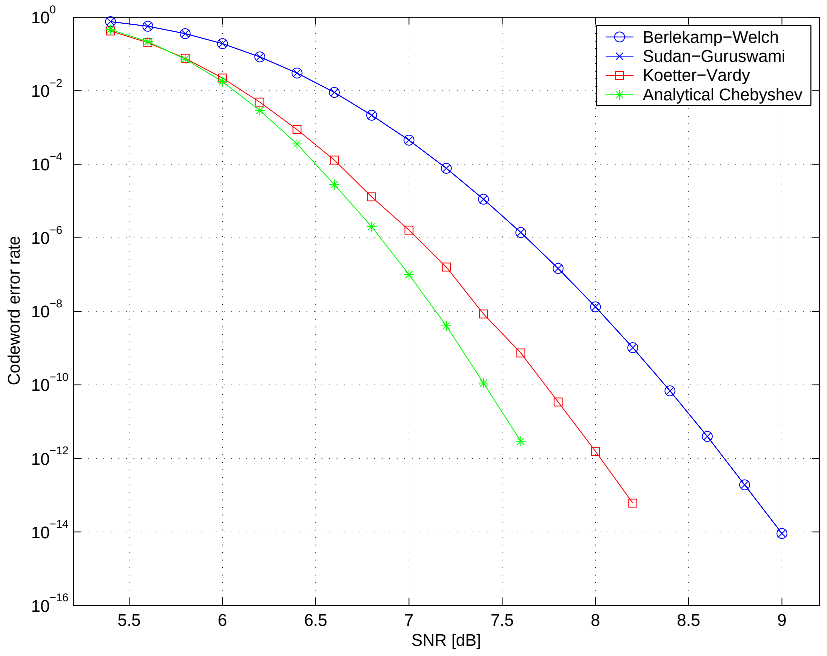


Figure 4.5: Performance of soft decoding of $\mathbb{C}(255, 239)$ over AWGN channel, when multiplicity of the interpolation tends to infinity.

define the vector forms as:

$$\Pi_{\text{vec}} \stackrel{\text{def}}{=} [\pi_{1,1}, \ldots, \pi_{q,1}, \pi_{1,2}, \ldots, \pi_{q,2}, \ldots, \pi_{q,n}]^t$$
$$M_{\text{vec}} \stackrel{\text{def}}{=} [m_{1,1}, \ldots, m_{q,1}, m_{1,2}, \ldots, m_{q,2}, \ldots, m_{q,n}]^t$$

Also, let P_j denotes the covariance matrix of the $\mathcal{S}_j(M)$, where $\mathcal{S}_j(M)$ is the random variable with the distribution of $\Pr\{\mathcal{S}_j(M) = m_{i,j}\} = \pi_{i,j}$. Notice that under this notation $\mathcal{S}_M = \mathcal{S}_1(M) + \mathcal{S}_2(M) + \cdots + \mathcal{S}_n(M)$. The covariance matrix for $\mathcal{S}_j$ is equal to

$$P_j \stackrel{\text{def}}{=} \begin{pmatrix} \pi_{1,j}(1-\pi_1,j) & -\pi_{1,j}\pi_{2,j} & \ldots & -\pi_{1,j}\pi_{q,j} \\ -\pi_{1,j}\pi_{2,j} & \pi_{2,j}(1-\pi_{2,j}) & \ldots & -\pi_{2,j}\pi_{q,j} \\ \vdots & \vdots & \ddots & \vdots \\ -\pi_{1,j}\pi_{q,j} & \ldots & & \pi_{q,j}(1-\pi_{q,j}) \end{pmatrix} \tag{4.23}$$

In addition, we the covariance matrix of $\mathcal{S}_M$ becomes

$$P \overset{\text{def}}{=} \begin{pmatrix} P_1 & & & & \\ & P_2 & & 0 & \\ & & \ddots & & \\ & 0 & & \ddots & \\ & & & & P_n \end{pmatrix}. \tag{4.24}$$

Notice that the random variable $\mathcal{S}_i$ is independent of $\mathcal{S}_j$ for any i and j that $i \neq j$. Then for the mean and variance of the score $\mathcal{S}_M$ we have

$$\mu(\mathcal{S}_M) = M_{\text{vec}}\Pi^t_{\text{vec}} \;,\; \text{Var}(\mathcal{S}_M) = M^t_{\text{vec}} P M_{\text{vec}} \tag{4.25}$$

and $L(M)$ defined in (4.18) turns into the form of

$$L(M) = \frac{M^t_{\text{vec}}\Pi_{\text{vec}} - \sqrt{(k-1)M^t_{\text{vec}}M_{\text{vec}}}}{\sqrt{M^t_{\text{vec}}PM_{\text{vec}}}} \tag{4.26}$$

Matrix P is symmetric along its main diagonal, so there exists a unitary matrix V such that $P = V^t D V$, where D is a diagonal matrix and the diagonal elements of D are the eigen values of P. we map our multiplicity vector and reliability vector by V to the domain that the covariance matrix P has the diagonal form of D:

$$M'_{\text{vec}} \leftarrow VM_{\text{vec}} \;, \quad \Pi'_{\text{vec}} \leftarrow V\Pi_{\text{vec}} \tag{4.27}$$

Hence (4.18) becomes

$$L(M'_{\text{vec}}) = \frac{M'^t{}_{\text{vec}}\Pi'_{\text{vec}} - \sqrt{(k-1)M'^t{}_{\text{vec}}M'_{\text{vec}}}}{\sqrt{M'^t{}_{\text{vec}}DM'_{\text{vec}}}} \tag{4.28}$$

Similar to Section 4.4.1, by using Lagrange multipliers, we optimize (4.28) under the conditions that $M'^t{}_{\text{vec}}M'_{\text{vec}} = 1$ and $M'^t{}_{\text{vec}}DM'_{\text{vec}} = \sigma^2$. The solution to the optimization is

$$M'_{\text{vec}} = (I + \lambda D)^{-1}\Pi'_{\text{vec}} \tag{4.29}$$

where λ is an arbitrarily constant. To find λ we replace M'_{vec} in (4.28) by RHS of (4.29). This gives us an optimization over λ:

$$L(\lambda) = \frac{g(\lambda) - \sqrt{(k-1)\left(g(\lambda) + \lambda g'(\lambda)\right)}}{\sqrt{-g'(\lambda)}} \tag{4.30}$$

where

$$g(\lambda) \overset{\text{def}}{=} \Pi'^{t}{}_{\text{vec}}(I+\lambda D)^{-1}\Pi'{}_{\text{vec}} \quad \text{and} \quad g'(\lambda) = \frac{d}{d\lambda}g(\lambda) \tag{4.31}$$

The derivative of $L(\lambda)$ respect to λ is zero whenever $g(\lambda) + \lambda g'(\lambda) = k-1$:

$$\Pi'^{t}{}_{\text{vec}}\left(I+\lambda D\right)^{-2}\Pi'_{\text{vec}} = k-1 \tag{4.32}$$

After simplification (4.32) gives a univariate polynomial of degree at most $2n$ over λ where the zeros of the polynomial are solution of (4.32). Any efficient root finding algorithm such as Newton method can be used to find zeros of the univariate polynomial. We set λ^* to be the root that maximizes (4.32). Then (4.29) computed at $\lambda = \lambda^*$ gives us the multiplicities.

Multiplicity assignment algorithm:

`Input`: **Reliability**

$$\Pi_{\text{vec}} := [\pi_{1,1}, \ldots, \pi_{q,1}, \pi_{1,2}, \ldots, \pi_{q,2}, \ldots, \pi_{q,n}]^t$$

`Output`: **Multiplicities**

$$M_{\text{vec}} := [m_{1,1}, \ldots, m_{q,1}, m_{1,2}, \ldots, m_{q,2}, \ldots, m_{q,n}]^t$$

$\boxed{1}$ Construct matrix P as defined in (4.24) and (4.23).

$\boxed{2}$ Find the unitary transformation V that $P = V^tDV$ for a diagonal matrix D; Set $D \leftarrow VPV^t$ and $\Pi'_{\text{vec}} \leftarrow V\,\Pi_{\text{vec}}$.

$\boxed{3}$ Find all the roots λ that satisfies the following equation

$$\Pi'^{t}{}_{\text{vec}}\left(I+\lambda D\right)^{-2}\Pi'_{\text{vec}} = k-1$$

$\boxed{4}$ Set λ^* to be the root among all the solution of step $\boxed{3}$ that maximizes the value of $L(\lambda)$ in (4.30).

$\boxed{5}$ Output $M_{\text{vec}} \leftarrow V^t\left(1+\lambda^* D\right)^{-1}\Pi'_{\text{vec}}$.

This algorithm is computationally more complicated than Koetter-Vardy multiplicity assignment. We find an approximation for the algorithm when the SNR is high; In those cases we are expecting that the received symbols to be reliable with high probability. In other words, at each column of reliability matrix we should have one or two elements with nonzero reliability and the reliability of the rest should be very close to zero. We only keep the first two most reliable symbol at each column of reliability matrix and consider the reliability of the rest to be zero. This reduces the size of the reliability matrix from $q \times n$ to $2 \times n$. Also, we assume that $\pi_{1,j} + \pi_{2,j} \simeq 1$ for any $j = 1, 2, \ldots, n$. Hence, without loss of generality $\pi_{1,j}$ and $\pi_{2,j}$ denote the reliability of the most reliable and second most reliable symbol of the jth received word, respectively. Then for covariance matrix and its diagonal form we have

$$P_j = \pi_{1,j}\pi_{2,j} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \; D_j = \pi_{1,j}\pi_{2,j} \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}, \; V_j = \frac{\sqrt{2}}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{4.33}$$

We also assume that $\pi_{1,j}\pi_{2,j}$ is small so we approximate the terms of the form $(\pi_{1,j}\pi_{2,j})(\pi_{i,j'}\pi_{2,j'})$ with zero. This simplifies (4.32) to:

$$\lambda^* = \frac{1}{2\sum_{j=1}^{n} \pi_{1,j}\pi_{2,j}} \left(\sqrt{\frac{\sum_{j=1}^{n} (\pi_{1,j} - \pi_{2,j})^2}{2(k-1) - n}} - 1 \right) \tag{4.34}$$

Using the result of (4.34) in (4.29) gives us the following multiplicity assignment algorithm:

Choose:

$$\lambda^* \simeq \frac{n}{2\sum_{j=1}^{n} \pi_{1,j}\pi_{2,j}} \left(\sqrt{\frac{\sum_{j=1}^{n} (\pi_{1,j} - \pi_{2,j})^2}{2(k-1) - n}} - 1 \right)$$

Assign Multiplicities:

$$m_{1,j} = 1 + \frac{\pi_{1,j} - \pi_{2,j}}{1 + 2\pi_{1,j}\pi_{2,j}\lambda^*}$$

$$m_{2,j} = 1 - \frac{\pi_{1,j} - \pi_{2,j}}{1 + 2\pi_{1,j}\pi_{2,j}\lambda^*}$$

The complexity of this algorithm is essentially the same as of Koetter-Vardy algorithm. In Figure 4.4 and Figure 4.5 , the performance of Koetter-Vardy algorithm, the Chebyshev algorithm and the analytical solution are plotted.

4.5 Fixed-cost multiplicity assignment

Interpolation multiplicities are integer numbers and in practice have small values close to one. The complexity of decoder increase with cost of multiplicity matrix; To be able to decode in practice the cost of multiplicity matrix should be below certain limit. Finding the optimum *integer* value multiparities for a given cost is a really hard problem. Instead, we relax the optimization problem to *real* value multiplicities but with fixed given cost. Using the Lemmas and Theorems we developed in Section 4.3 we transform the finite cost problem into an infinite cost problem and then we try to mimic the ideas we introduced in Section 4.4 to solve the new optimization problem. It turns out that the multiplicity assignment for the finite cost after transformation into infinite cost is essentially the same as multiplicity assignment for infinite cost with the extra condition on the summation of multiplicities; I.e., multiplicity vector lies on a certain cone in $\boldsymbol{\mathcal{V}}$.

The general formulation of the multiplicity assignment for fixed-cost $\mathcal{C}$ is the following:

$$M^* = \underset{M \in \mathbb{R}^{q \times n}}{\operatorname{argmin}} \ \Pr\{\mathcal{S}_M \leqslant \Delta(M)\} \tag{4.35}$$
$$\text{where} \quad \mathcal{C}(M) = \sum_{i=1}^{q} \sum_{j=1}^{q} {}^{1}\!/\!{}_{2} m_{i,j} \left(m_{i,j} + 1\right) = \mathcal{C}$$

For simplicity lets denote by $\Delta_{\mathcal{C}}$ the value of $\Delta(\mathcal{C})$. From Theorem 4.2, the solution of (4.35) lie on the cost sphere $\mathscr{S}_{\mathcal{C}}$ of radius $r_{\mathcal{C}} = \sqrt{2\mathcal{C} + \frac{qn}{4}}$ centered about the point $(-{}^{1}\!/\!{}_{2}, -{}^{1}\!/\!{}_{2}, \ldots, -{}^{1}\!/\!{}_{2}) \in \boldsymbol{\mathcal{V}}$. Also, from Theorem 4.10, M^* lies on a tangent line from ξ to $\mathscr{S}_{\mathcal{C}}$. Let $\mathscr{T}_{\xi}(\mathscr{S}_{\mathcal{C}})$ denotes the following set

$$\mathscr{T}_{\xi}(\mathscr{S}_{\mathcal{C}}) \overset{\text{def}}{=} \{M \in \mathscr{S}_{\mathcal{C}} \ : \ M \text{ belongs to tangent line from } \xi \text{ to } \mathscr{S}_{\mathcal{C}}\} \tag{4.36}$$

Then the optimization of (4.35) is equivalent to

$$M^* = \underset{M \in \mathscr{T}_\xi(\mathscr{S}_\mathcal{C})}{\arg\min} \Pr\{\mathcal{S}_M \leqslant \Delta_\mathcal{C}\} \tag{4.37}$$

Theorem 4.12. *Let $\mathscr{K}_\mathcal{C}$ be the following set*

$$\mathscr{K}_\mathcal{C} \overset{\text{def}}{=} \{\lambda M + (1-\lambda)\xi \ : \ M \in \mathscr{T}_\xi(\mathcal{C}) \text{ and } \lambda \in [0,\infty) \subset \mathbb{R}\} \tag{4.38}$$

then the solution to optimization problem of (4.37) is also the solution to optimization problem of

$$M^* = \underset{M \in \mathscr{K}_\mathcal{C}}{\arg\min}\, Pr\{\mathcal{S}_M \leqslant \Delta_\mathcal{C}\} \tag{4.39}$$

In addition, let M^* denotes the solution of (4.39). Then the solution of (4.37) and (4.35) has the form of $\lambda M^* + (1-\lambda)\xi$ for some $\lambda \in [0,\infty) \subset \mathbb{R}$.

Proof. From Lemma 4.7, $\varphi_\Delta(M) = \Pr\{\mathcal{S}_M > \Delta\}$ is invariant under the transformation $M \to \lambda M + (1-\lambda)\xi$, for $\lambda \in [0,\infty) \subset \mathbb{R}$. The set $\mathscr{K}_\mathcal{C}$ is the extension of the set $\mathscr{T}_\xi(\mathcal{C})$ under the transformation $M \to \lambda M + (1-\lambda)\xi$. Thus the solution to optimization of (4.37) and (4.39) are the same modulo the transformation $M \leftarrow \lambda M + (1-\lambda)\xi$. ∎

Lemma 4.13. *The set $\mathscr{K}_\mathcal{C}$ is a cone in $\boldsymbol{\mathcal{V}}$ defined by the set of ray in $\boldsymbol{\mathcal{V}}$ starting at ξ and are tangent to the sphere $\mathscr{S}_\mathcal{C}$; Explicitly, let θ_M denotes the angle between the vectors $\frac{\Delta_\mathcal{C}}{n}\mathbf{1}$ and $\frac{\Delta_\mathcal{C}}{n}\mathbf{1} - M$ in $\boldsymbol{\mathcal{V}}$.*

$$\mathscr{K}_\mathcal{C} = \left\{ M \in V \ : \ \cos\theta_M = \sqrt{1 - \frac{r_\mathcal{C}^2}{\left(\frac{\Delta_\mathcal{C}}{n} + 1/2\right)^2 qn}} \right\} \tag{4.40}$$

Proof. In the definition (4.38) we implicitly defined the set $\mathscr{K}_\mathcal{C}$

$$\mathscr{K}_\mathcal{C} = \{M \in V \ : \ \text{where the ray } \lambda M + (1-\lambda)\xi \quad \text{for } \lambda \in [0,\infty) \text{ is tangent to sphere } \mathscr{S}_\mathcal{C}.\}$$

The reset of the proof is just the result of the geometry of the cone $\mathscr{K}_\mathcal{C}$ in the space $\boldsymbol{\mathcal{V}}$. See Figure 4.6. ∎

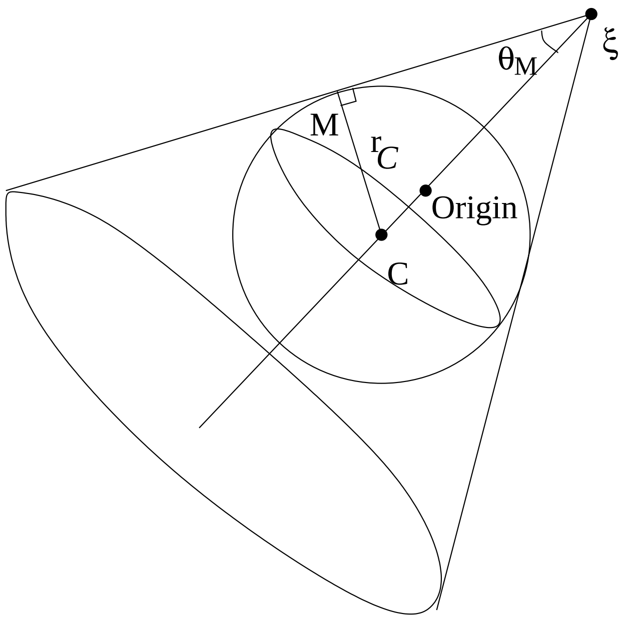


Figure 4.6: Geometrical framework; The cone and tangent rays

The definition of the cone becomes much simpler for the case that $\|M\|$ is large. In this case, we can approximate $\frac{\Delta_{\mathcal{C}}}{n}\mathbf{1} - M$ by $-M$ and therefore for the set $\mathscr{K}_{\mathcal{C}}$ we get:

$$\mathscr{K}_{\mathcal{C}} = \left\{ M \in V \; : \; \langle M, \mathbf{1} \rangle = -\|M\| \sqrt{qn - \left(\frac{r_{\mathcal{C}}}{\frac{\Delta_{\mathcal{C}}}{n} + 1/2} \right)^2} \right\}, \text{ as } \|M\| \to \infty \tag{4.41}$$

So we solve the optimization problem of (4.39) on the tail of the cone $\mathscr{K}_{\mathcal{C}}$ where $\|M\|$ is large and the simpler definition of the cone $\mathscr{K}_{\mathcal{C}}$ is applicable. After finding the solution of (4.39) for large $\|M\|$ we map it to finite cost multiplicity M by moving along the ray $\lambda M + (1 - \lambda)\xi$ toward ξ.

Similar to Section 4.4, we instead of optimizing directly the value of $\Pr\{\mathcal{S}_M \leqslant \Delta(M)\}$ we minimize the Chebyshev upper bound on the probability of failure of decoder. Because we have already considered that $\|M\|$ is large we can use (4.18) for the upper bound. We know value of $L(M)$ in (4.18) is independent of the norm of M and it only depends on the direction of M in the space $\boldsymbol{\mathcal{V}}$. In addition from (4.41) if M is in the set $\mathscr{K}_{\mathcal{C}}$ then also λM is in $\mathscr{K}_{\mathcal{C}}$ for any $\lambda \in \mathbb{R}^+$

. Thus, without loss of generality we assume that norm of M is ***one*** in our optimization. The optimization becomes:

$$M^*_\infty = \underset{M}{\operatorname{argmax}} \ L(M) = \frac{\sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}\pi_{i,j} - \sqrt{(k-1)}}{\sqrt{\sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}^2\pi_{i,j} - \sum_{j=1}^{n}\left(\sum_{i=1}^{q} m_{i,j}\pi_{i,j}\right)^2}} \tag{4.42}$$

under the conditions:

$$\sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j} = -\sqrt{qn - \left(\frac{r_{\mathcal{C}}}{\frac{\Delta_{\mathcal{C}}}{n} + 1/2}\right)^2}, \qquad \sum_{j=1}^{n}\sum_{i=1}^{q} m_{i,j}^2 = 1$$

The optimization of (4.42) can be solved using Lagrange multipliers. Similar to Section 4.4.2, we use vector notations for M and Π and transfer the optimization in the matrix form of

$$M^*_{\infty,\text{vec}} = \underset{M_{\text{vec}}}{\operatorname{argmax}} \ \frac{M_{\text{vec}}^t \Pi_{\text{vec}} - \sqrt{k-1}}{\sqrt{M_{\text{vec}}^t P M_{\text{vec}}}} \tag{4.43}$$

under the conditions:

$$M_{\text{vec}}^t J_{\text{vec}} = -\sqrt{qn - \left(\frac{r_{\mathcal{C}}}{\frac{\Delta_{\mathcal{C}}}{n} + 1/2}\right)^2}, \qquad M_{\text{vec}}^t M_{\text{vec}} = 1$$

where $J_{\text{vec}} = [1, 1, \ldots, 1]^t$ is a vector of ones. Let V be the unitary matrix that for P we have $P = V^t D V$ where D is a diagonal matrix. Also, let $M'_{\text{vec}} := VM_{\text{vec}}$, $\Pi'_{\text{vec}} := V\Pi_{\text{vec}}$ and $J'_{\text{vec}} := VJ_{\text{vec}}$. The solution to the optimization has the following form:

$$M'_{\infty,\text{vec}} = (\eta I + \lambda D)^{-1} \left(\Pi'_{\text{vec}} + \delta J'_{\text{vec}}\right) \tag{4.44}$$

where η, λ, and δ are constant, from Lagrange multipliers method. In general, two of the constants can be eliminated by using the two conditions of optimization (4.43). To eliminate the third constant we have to replace M in (4.43) by (4.44). This gives a function only, let say, in λ; Then we optimize the function one more time over λ to find the optimum λ. The computation is messy and we omit the results.

The final step of the algorithm is to transfer the optimal multiplicity at infinity to finite cost. As shown in Figure 4.6, the finite multiplicity vector is

$$M^* = \frac{\Delta_{\mathcal{C}}}{n}\mathbf{1} + M^*_\infty \sqrt{\left(\frac{\Delta_{\mathcal{C}}}{n} + \frac{1}{2}\right)^2 qn - r^2_{\mathcal{C}}} \tag{4.45}$$

assuming that $\|M^*_\infty\| = 1$.

Remark. The optimization of (4.43) is well known as *quadratic fractional optimization* in literature and there are efficient algorithms other that Lagrange multipliers for that. It remains as an open problem to find the best method that is well suited for our multiplicity assignment algorithm among the quadratic fractional programs.

We have presented the results of Chapter 4, in part, at the *2003 International Symposium on Information Theory (ISIT)*, Parvaresh, Farzad; Vardy, Alexander. The dissertation author was the primary investigator and author of this paper.

CHAPTER 5

Efficient interpolation via matrix-chain product

The main computational steps in algebraic soft-decoding [KV03a] and/or Sudan-type list decoding [GS99] of Reed-Solomon codes are interpolation and factorization. The interpolation consists of computing a certain bivariate polynomial $\mathcal{Q}(X,Y)$ that passes through a prescribed set of points with prescribed multiplicities. Using the iterative algorithm of Koetter [Köt96, Koe96, McE03b], this computation can be accomplished in time $O(N^2)$, where N is the number of linear equations that should be satisfied by the coefficients of $\mathcal{Q}(X,Y)$. Here, we recast the iterative interpolation procedure of [Köt96, Koe96] as a computation of the product of a certain chain of polynomial matrices. We then derive a dynamic-programming algorithm which optimizes the multiplication order in computing such polynomial matrix products. The resulting optimization reduces the number of finite-field operations needed to compute $\mathcal{Q}(X,Y)$ by a factor of about two. Our approach can be easily combined with other computational speed-ups, such as the re-encoding method developed in [KV03b, KMVA03].

5.1 Introduction

As we have seen in Chapter 1 and Chapter 4 list decoding and algebraic soft-decision decoding use interpolation and factorization of bivariate polynomials, which is much more computationally intensive than hard-decision decoding.

Bivariate interpolation consists of computing a polynomial $\mathcal{Q}(X,Y)$ of minimum weighted-degree that passes through a prescribed set of points $\mathcal{P} = \{(x_1,y_1),(x_2,y_2),\ldots,(x_s,y_s)\} \subseteq \mathbb{F}_q^2$ with prescribed multiplicities $m_{x_1,y_1}, m_{x_2,y_2}, \ldots, m_{x_s,y_s} \in \mathbb{N}$. This is, by far, the most time-consuming task in Sudan-type decoding: it is widely recognized as the computational bottleneck of the decoder. Several efficient algorithms for such bivariate interpolation have been proposed in the literature [AKS03, FG01, KV03b, McE03b, NH98, OS99, RR00]. Most of these are based on the iterative algorithm, originally developed by Koetter [Köt96, Koe96], for computing a Gröbner basis for the ideal of polynomials that pass through the points in $\mathcal{P}$ with the prescribed multiplicities. We will briefly review Koetter's algorithm in the next section. This algorithm takes about $O(N^2)$ finite-field operations to compute $\mathcal{Q}(X,Y)$, where N is given by

$$N \overset{\text{def}}{=} \frac{1}{2}\sum_{i=1}^{s} m_{x_i,y_i}(m_{x_i,y_i}+1)$$

We note that this fundamental iterative algorithm is used in one form or another in all of [AKS03, FG01, KV03b, McE03b, NH98, RR00] and serves as the basis for all existing implementations of Sudan-type decoders [AKS03, GKKG02].

In the next section, we observe that the computation performed in Koetter's algorithm [Köt96, Koe96] is tantamount to computing the matrix-chain product $A_1A_2\cdots A_N$, where $A_1, A_2, \ldots, A_N$ are square *polynomial* matrices over $\mathbb{F}_q$. Even though all these matrices are of the same dimension $\ell\times\ell$, the fact that the entries in A_i are polynomials in X implies that the total number of finite-field operations required to compute $A_1A_2\cdots A_N$ depends on the *order* in which the multiplication is carried out. In all of [AKS03, FG01, KV03b, McE03b, NH98, RR00], this multiplication is carried out in the natural order

$$\Big(\Big(\Big((A_1A_2)A_3\Big)\,A_4\Big)\cdots A_N\Big) \tag{5.1}$$

Instead, we derive in Section 5.3, a dynamic-programming algorithm which optimizes the multiplication order in computing $A_1A_2\cdots A_N$. The resulting optimization reduces the number of finite-field operations required to compute $A_1A_2\cdots A_N$ — and thus construct the interpolation polynomial $\mathcal{Q}(X,Y)$ — by a *factor of about two* as compared to (5.1).

We can further reduce the interpolation complexity by yet another factor of two, by observing that we only need to compute *one element* — the one with the least weighted degree — from a Gröbner basis for the relevant ideal. In Section 5.5, we present an optimum and suboptimum algorithms for this case.

The problem with the optimal dynamic-programming algorithm of Section 5.3 is that it requires $O(N^3)$ integer additions and comparisons, and therefore might take more time than computing the product $A_1A_2\cdots A_N$ itself, which takes only $O(N^2)$ finite-field operations. Thus in Section 5.4, we develop a suboptimal version of this algorithm which runs in time $O(dN^2)$, where d is a small parameter, and achieves almost the same reduction in complexity. The constants in $O(\cdot)$ work-out in our favor. For example, for $N = 1000$ and $\ell = 16$, the sequential computation in (5.1) takes 53×10^6 finite-field operations. With $d = 8$, we pay only about 8×10^6 integer additions to reduce the number of finite-field operations required to compute $A_1A_2\cdots A_N$ to 33×10^6.

We point out that several other methods for reducing the complexity of interpolation in Sudan-type decoding have been recently proposed, most notably [FG01] and [KV03b]. Indeed, the savings obtained here pale in comparison with the re-encoding idea of [KMVA03, KV03b], which reduces the complexity of interpolation by a factor of at least $n^2/(n-k)^2$ for a Reed-Solomon code of length n and dimension k. The point is that our results can be *easily combined* with the methods of [FG01, KMVA03, KV03b] and other authors. Whenever there is iterative interpolation, it can be recast as a polynomial matrix-chain product computation. Whenever the multiplication of the matrices in this chain can be performed in an arbitrary order (as is often the case), our methods will apply. Thus the savings we obtain in are *on top* and not instead of previous reductions in interpolation complexity.

5.2 Interpolation as a matrix-chain product

The central idea underlying all of the Sudan-type list decoding algorithms is to construct a polynomial $\mathcal{Q}(X,Y)$ of minimal $(1,k-1)$-weighted degree that passes through a prescribed set of points $\mathcal{P} = \{(x_1,y_1),(x_2,y_2),\ldots,(x_s,y_s)\}$ with prescribed multiplicities $m_{x_1,y_1}, m_{x_2,y_2},\ldots,m_{x_s,y_s} \in \mathbb{N}$. More specifically, let $\mathcal{D}_{u,v}\, P(X,Y)$ denote the (u,v)-th partial Hasse derivative of a bivariate polynomial $\mathcal{P}(X,Y)$. Then the *interpolation polynomial* $\mathcal{Q}(X,Y)$ can be defined as the least $(1,k-1)$ weighted-degree polynomial in $\mathbb{F}_q[X,Y]$ that satisfies the following set of linear equations:

$$\forall (x,y)\in\mathcal{P},\ \forall u,v\in\mathbb{Z} \text{ with } 0 \leqslant u+v < m_{x,y}\ :\ \mathcal{D}_{u,v}\, \mathcal{Q}(X,Y)\Big|_{(x,y)} \;=\; 0 \quad (5.2)$$

Note that the total number of equations in (5.2) is precisely N. Koetter's algorithm [Köt96, Koe96] proceeds by iteratively constructing the minimal (in the $(1,k-1)$-weighted-degree monomial order) Gröbner basis for the ideal of polynomials that satisfy the first i equations in (5.2), for $i = 0,1,\ldots,N$. We will often think of this Gröbner basis as a vector of bivariate polynomials $(Q_0,\ldots,Q_{\ell-1})$, where ℓ is the smallest integer such that $2N < \ell(\ell+1)(k-1)$. For $i = 0$, this vector is initialized to

$$\left(Q_0^{(0)}, Q_1^{(0)}, Q_2^{(0)},\ldots, Q_{\ell-1}^{(0)}\right) \;=\; \left(1, Y, Y^2,\ldots, Y^{\ell-1}\right)$$

Now suppose that at iteration i of the algorithm, we are dealing with the (a,b)-th Hasse derivative at the point $(x_r,y_r)\in\mathcal{P}$ — that is, $(x,y) = (x_r,y_r)$ and $(u,v) = (a,b)$ in (5.2). Then Koetter's algorithm proceeds as follows. Given the current basis $Q_0^{(i-1)}(X,Y), Q_1^{(i-1)}(X,Y),\ldots,Q_{\ell-1}^{(i-1)}(X,Y)$:

$\boxed{1}$ For all $j = 0,1,\ldots,\ell-1$:
Compute the *discrepancy* $\Delta_j = \mathcal{D}_{a,b}\, Q_j^{(i-1)}(X,Y)\Big|_{(x_r,y_r)}$.

$\boxed{2}$ Among $Q_0^{(i-1)}(X,Y), Q_1^{(i-1)}(X,Y),\ldots,Q_{\ell-1}^{(i-1)}(X,Y)$, find the least $(1,k-1)$-weighted-degree polynomial $Q_t^{(i-1)}(X,Y)$ such that its discrepancy Δ_t is nonzero. We call this the pivot polynomial and say that $\tau(i) \overset{\text{def}}{=} t$ is the *pivot index* at iteration i.

$\boxed{3}$ For all $j = 0, 1, \dots, \ell{-}1$, except $j = t$, compute

$$Q_j^{(i)}(X,Y) \leftarrow Q_j^{(i-1)}(X,Y) - \frac{\Delta_j}{\Delta_t} Q_t^{(i-1)}(X,Y)$$

Then set $Q_t^{(i)}(X,Y) \leftarrow (X - x_r) Q_t^{(i-1)}(X,Y)$.

In fact, the only "essential" computation is the one performed in Step 3 of the above algorithm. The discrepancies $\Delta_1, \Delta_2, \dots, \Delta_{\ell-1}$ and the pivot index $t = \tau(i)$ can be pre-computed using the so-called discrepancy polynomials, as suggested in [AKS03]. Moreover, if the divide-and-conquer methods of [FG01, MTV04] are used, these discrepancies would often be already available to the decoder. The computation in Step 3 of the algorithm can be represented in matrix form as follows:

$$\begin{bmatrix} Q_0^{(i)}(X,Y) \\ Q_1^{(i)}(X,Y) \\ \vdots \\ Q_{\ell-1}^{(i)}(X,Y) \end{bmatrix} = \begin{bmatrix} 1 & & \beta_{i,0} & & \\ & 1 & \beta_{i,1} & & \\ & & \ddots \;\vdots & & \\ & & X - x_r & & \\ & & \vdots & \ddots & \\ & & \beta_{i,\ell-1} & & 1 \end{bmatrix} \begin{bmatrix} Q_0^{(i-1)}(X,Y) \\ Q_1^{(i-1)}(X,Y) \\ \vdots \\ Q_{\ell-1}^{(i-1)}(X,Y) \end{bmatrix} \tag{5.3}$$

where, in the $\ell \times \ell$ matrix above, $\beta_{i,j} = -\Delta_j/\Delta_t$ for all $j \in \{0, 1, \dots, \ell{-}1\} \setminus \{t\}$, the column containing $X - x_r$ is in the pivot position t, and all the other columns are of the form $(00\cdots010\cdots0)^t$. Let us denote the $\ell \times \ell$ polynomial matrix in (5.3) by A_i. Then it should be obvious that

$$\begin{bmatrix} Q_0^{(N)}(X,Y) \\ Q_1^{(N)}(X,Y) \\ \vdots \\ Q_{\ell-1}^{(N)}(X,Y) \end{bmatrix} = A_N A_{N-1} \cdots A_2 A_1 \begin{bmatrix} 1 \\ Y \\ Y^2 \\ \vdots \\ Y^{\ell-1} \end{bmatrix} \tag{5.4}$$

This is the desired formulation of the iterative interpolation algorithm as a polynomial matrix-chain product. Since matrix product is associative, the computation in (5.4) can be carried out *in any order*, and not necessarily in the sequential iterative order implied by (5.3). Finding the "best" multiplication order for computing $A_N A_{N-1} \cdots A_1$ in (5.4) is the subject of the next section.

Remark. In the context of (5.4), the algorithm proposed by Feng and Giraud in [FG01] essentially becomes a recursive divide-and-conquer parenthesization of the matrix chain $A_N A_{N-1} \cdots A_1$.

5.3 Optimal multiplication order

It is well known that an appropriate parenthesization of a chain of matrices can have a dramatic impact on the cost of evaluating their product. Given a matrix-chain product $A_1 A_2 \cdots A_N$, there is an exponential number of ways to parenthesize this product, given by the Catalan number

$$\mathcal{C}_{N-1} \overset{\text{def}}{=} \frac{1}{N}\binom{2(N-1)}{N-1} = \frac{(N+1)(N+2)\cdots 2(N-1)}{1{\cdot}2\cdots(N-1)}$$

The standard matrix-chain multiplication problem, where the dimensions of the matrices are *different* while the matrix elements are *scalars*, was first solved using dynamic-programming by Godbole [God73] in 1973. Godbole's algorithm runs in time $O(N^3)$. A more efficient algorithm, that runs in time $O(N \log N)$, was given by Hu and Shing [CT84] in 1984. Variations of this problem are still a subject of active research interest — see the recent survey in [LKHL03].

Here, we are interested in parenthesizing the chain $A_N A_{N-1} \cdots A_1$ in (5.4), where A_i is the matrix defined in (5.3). For convenience, we shall henceforth reverse the indexing order and write this chain as $A_1 A_2 \cdots A_N$, with the convention that $i \leftarrow (N{+}1) - i$ with respect to (5.4).

We first need an expression for the cost of multiplying two univariate polynomials and for the cost of multiplying two polynomial matrices. To this end, we make several simplifying assumptions. Given $p(X) \in \mathbb{F}_q[X]$, we define

$$\operatorname{Deg} p(X) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } p(X) = 0 \text{ or } p(X) = 1 \\ 1 + \deg p(X) & \text{otherwise} \end{cases}$$

where $\deg p(X)$ is defined as usual. We then assume that the cost of multiplying two polynomials $p(X), q(X) \in \mathbb{F}_q[X]$ is $\operatorname{Deg} p(X) \operatorname{Deg} q(X)$. Indeed, if $p(X)$ and $q(X)$ are both nonzero and non-sparse, then computing $p(X)q(X)$ (in a straightforward fashion, see the remark below) takes about $(\deg p(X) + 1)(\deg q(X) + 1)$ finite-field additions/multiplications. Now let $U = [u_{i,j}(X)]$ and $V = [v_{i,j}(X)]$ be $\ell \times \ell$ polynomial matrices. Then we assume that the cost of computing UV is

$$\boldsymbol{C}(U,V) \overset{\text{def}}{=} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \sum_{k=1}^{\ell} \operatorname{Deg} u_{i,j}(X) \operatorname{Deg} v_{j,k}(X) \tag{5.5}$$

Remark. There are fast polynomial-multiplication algorithms [Gan89] that compute $p(X)q(X)$ with only $O(D \log D)$ finite-field multiplications, providing $\deg p(X), \deg q(X) < 2^D - 1$. We can account for such algorithms by modifying the expression in (5.5) — the analysis remains virtually unchanged. There are also fast matrix-multiplication algorithms [Pan92] that compute the product of two $\ell \times \ell$ matrices in time $O(\ell^{2.496})$. However, such algorithms involve significant overhead. They are therefore unlikely to be advantageous in our application where ℓ is small (usually $4 \leqslant \ell \leqslant 16$).

We now return to the original problem of parenthesizing the matrix-chain $A_1 A_2 \cdots A_N$ in (5.4). Given integers i and j such that $1 \leqslant i < j \leqslant N$, let $\mathscr{C}(i,j)$ denote the optimal cost of computing the subchain $A_i A_{i+1} \cdots A_j$. Thus what we are after is $\mathscr{C}(1,N)$. Clearly, an optimal parenthesization of $A_i A_{i+1} \cdots A_j$ splits this product between A_k and A_{k+1} for *some* k in the range $i \leqslant k < j$. Thus

$$\mathscr{C}(i,j) = \min_{i \leqslant k < j} \Big\{ \mathscr{C}(i,k) + \mathscr{C}(k{+}1,j) + \boldsymbol{C}\Big((A_i A_{i+1} \cdots A_k), (A_{k+1} A_{k+2} \cdots A_j)\Big) \Big\} \tag{5.6}$$

To complete the recursion, we must compute $\boldsymbol{C}\Big((A_i \cdots A_k), (A_{k+1} \cdots A_j)\Big)$. To this end, it will be convenient to introduce *degree matrices*, defined as follows. Given an $\ell \times \ell$ polynomial matrix $U = [u_{i,j}(X)]$, we define the corresponding $\ell \times \ell$ integer matrix $\operatorname{Deg} U = [\operatorname{Deg} u_{i,j}(X)]$ and call it the *degree matrix of* U. With this notation, the following lemma shows that the expression for $\boldsymbol{C}(U,V)$ in (5.5) can be computed with only ℓ rather than ℓ^3 integer multiplications.

Lemma 5.1. *Let $\boldsymbol{C}(U,V)$ be the cost of multiplying the $\ell \times \ell$ polynomial matrices U and V, as defined in* (5.5). *Then*

$$\boldsymbol{C}(U,V) \;=\; (\overbrace{1,1,\ldots,1}^{\ell})\,\mathrm{Deg}\,U\;\mathrm{Deg}\,V\,(\overbrace{1,1,\ldots,1}^{\ell})^t \tag{5.7}$$

Proof. The expression on the right-hand side of (5.5) can be re-written as

$$\sum_{i=1}^{\ell}\sum_{j=1}^{\ell}\sum_{k=1}^{\ell}\mathrm{Deg}\,u_{i,j}(X)\,\mathrm{Deg}\,v_{j,k}(X) \;=\; \textstyle\sum_{j=1}^{\ell}\Big(\sum_{i=1}^{\ell}\mathrm{Deg}\,u_{i,j}(X)\Big)\Big(\sum_{k=1}^{\ell}\mathrm{Deg}\,v_{j,k}(X)\Big) \quad \blacksquare$$

It follows from Lemma 5.1 that if $\mathrm{Deg}(A_iA_{i+1}\cdots A_k)$ and $\mathrm{Deg}(A_{k+1}\cdots A_j)$ are known, then $\boldsymbol{C}\Big((A_iA_{i+1}\cdots A_k),(A_{k+1}A_{k+2}\cdots A_j)\Big)$ can be computed with $\ell(2\ell+1)$ integer additions and ℓ integer multiplications. Thus our remaining task is to compute $\mathrm{Deg}(A_jA_{j+1}\cdots A_i)$ for all i and j such that $1 \leqslant j < i \leqslant N$. This can be done in time $O(N^2)$ using the following lemma.

Lemma 5.2. *Let $U = A_jA_{j+1}\cdots A_{i-1}$ and $V = UA_i$. Let $\tau(i)$ and $\tau(i-1)$ denote the pivot indices at iterations i and $i-1$, respectively, of the Koetter algorithm* [Köt96, Koe96] *(cf. Section 5.3). Then*

$$(\mathrm{Deg}\,V)_{k,l} \;=\; \begin{cases} \max\Big\{(\mathrm{Deg}\,U)_{k,\tau(i-1)},(\mathrm{Deg}\,U)_{k,\tau(i)}+1\Big\} & \\ & \text{if } l=\tau(i) \text{ and } (\mathrm{Deg}\,U)_{k,\tau(i)}\neq 0 \\ (\mathrm{Deg}\,U)_{k,\tau(i-1)} & \text{if } l=\tau(i) \text{ and } (\mathrm{Deg}\,U)_{k,\tau(i)}=0 \\ (\mathrm{Deg}\,U)_{k,l} & \text{if } l\neq\tau(i) \end{cases}$$

Proof. We finally use the structure of the matrix A_i defined in (5.3). Write $U=[u_{k,l}(X)]$ and $V=[v_{k,l}(X)]$. Since $V=UA_i$, it is easy to see from (5.3) that $v_{k,l}(X)=u_{k,l}(X)$ unless $l=\tau(i)$. This shows that $(\mathrm{Deg}\,V)_{k,l}=(\mathrm{Deg}\,U)_{k,l}$ for all $l\neq\tau(i)$. For $l=\tau(i)$, we have

$$v_{k,\tau(i)}(X) \;=\; (X-x_r)u_{k,\tau(i)}(X) \;+\; \sum_{\substack{s=0\\ s\neq\tau(i)}}^{\ell-1}\beta_{i,s}\,u_{k,s}(X)$$

Thus if $u_{k,\tau(i)}(X)$ is equal to zero, then $(\operatorname{Deg} V)_{k,\tau(i)} = \max_{s\neq\tau(i)}\{(\operatorname{Deg} U)_{k,s}\}$ while if $u_{k,\tau(i)}(X) \neq 0$, then $(\operatorname{Deg} V)_{k,\tau(i)}$ is the maximum of $(\operatorname{Deg} U)_{k,\tau(i)} + 1$ and $\max_{s\neq\tau(i)}\{(\operatorname{Deg} U)_{k,s}\}$. It remains to show that $\max_{s\neq\tau(i)}\{(\operatorname{Deg} U)_{k,s}\} = (\operatorname{Deg} U)_{k,\tau(i-1)}$. That is basically inherited in the recursion. ▮

Lemma 5.2 shows that $\operatorname{Deg}(A_j A_{j+1} \cdots A_i)$ can be computed iteratively as follows. Start with $U = A_j$, then $\operatorname{Deg} U$ follows trivially from (5.3). Once $\operatorname{Deg} U$ is known for $U = A_j A_{j+1} \cdots A_{i-1}$, compute $\operatorname{Deg}(UA_i)$, and so forth. By Lemma 5.2, at each iteration we need to update only the $\tau(i)$ column of the degree matrix, which takes at most ℓ operations. Computing $\operatorname{Deg}(A_j A_{j+1} \cdots A_i)$ for all $i = j, j{+}1, \ldots, N$ in this manner thus takes at most $\ell(N - j)$ operations. Hence, the overall complexity of computing $\operatorname{Deg}(A_j A_{j+1} \cdots A_i)$ for all i, j such that $1 \leqslant j < i \leqslant N$ is $O(\ell N^2)$. Once the degree matrices are known, all the relevant multiplication costs in (5.6) readily follow by (5.7).

This finally closes the recursion in (5.6). What then remains is classical dynamic programming problem. This can be easily solved using, for example, the $O(N^3)$ algorithm of Godbole [God73]. We still need to verify whether the more efficient $O(N \log N)$ algorithm of Hu and Shing [CT84] would apply in our case. However, in the next section, we propose a simple variation to Godbole's algorithm, which ensures a running time of $O(N^2)$ with negligible degradation in performance — that is, almost the same savings in the complexity of multiplying $A_1 A_2 \cdots A_N$ (see Figures 5.1 and 5.2).

5.4 Suboptimal dynamic-programming algorithm

We now use empirical observations to derive a suboptimal version of Godbole's dynamic-programming algorithm [God73] that is well-suited to the problem of parenthesizing the chain $A_1 A_2 \cdots A_N$ in (5.4). This version runs substantially faster and performs almost as well as the optimal algorithm.

First, we observe that $\mathscr{C}(i, j)$ seems to depend on i and j only through their *difference* $j - i$. This is to be expected: since all the matrices in (5.4) have the same structure, the chain $A_1 A_2 \cdots A_N$ exhibits a "stationary process" behavior. Next we observe that the best way to split $A_iA_{i+1}\cdots A_j$ into $(A_i\cdots A_k)(A_{k+1}\cdots A_j)$ usually occurs for those values of k that are close to either i or j. Thus we propose to fix a small parameter d — the *depth* of the search — and restrict the minimization in (5.6) to either $i \leqslant k < i+d$ or $j-d \leqslant k < j$. This produces the following recursion

$$\widetilde{\mathscr{C}_d}(i,j) = \min_{\substack{i\leqslant k<\min\{j,i+d\}\\ \max\{j-d,i\}\leqslant k<j}} \Big\{\widetilde{\mathscr{C}_d}(i,k) + \widetilde{\mathscr{C}_d}(k+1,j) + \boldsymbol{C}\Big((A_iA_{i+1}\cdots A_k),(A_{k+1}A_{k+2}\cdots A_j)\Big)\Big\} \tag{5.8}$$

Since the depth of the search in (5.8) is fixed to at most $2d$ positions, where d is independent of the size $j-i$ of $A_iA_{i+1}\cdots, A_j$, the complexity of solving the recursion in (5.8) reduces to $O(dN^2)$.

5.5 Optimization over the interpolation polynomial

So far we have tried to find the best parenthesization of (5.1) in order to minimize the number of finite field operations that is needed to carry out the multiplication. However, our main goal is to find the interpolation polynomial. From (5.3) we observe that interpolation polynomial is the smallest $(1,k-1)$-weighted degree polynomial among $\left(Q_0^{(N)}, Q_1^{(N)}, \ldots, Q_{\ell-1}^{(N)}\right)$. Thus, in order to find the interpolation polynomial we do not actually need to compute the whole matrix of (5.1) and we only have to compute the row with smallest weighted-degree.

From Lemma 5.2, degree of $(A_1A_2\cdots A_N)$ can be computed in $O(N)$. Therefore, the row of $A_1A_2\cdots A_N$ that leads to the interpolation polynomial with smallest weighted-degree can be determined without doing any multiplication. We denote the row with smallest weighted-degree by s. Let $\mathscr{R}_s(i,j)$ denotes the optimal cost of computing the s-th row of the sub-chain $A_iA_{i+1}\cdots A_j$. We optimize the value of $\mathscr{R}_s(i,j)$ under the following scenario:

> In order to find the s-th row of $A_iA_{i+1}\cdots A_j$ we first split the chain into $A_iA_{i+1}\cdots A_k$ and $A_{k+1}A_{k+2}\cdots A_j$ for some k in the range $i \leqslant$

$k < j$. Then we compute the s-th row of $A_i A_{i+1} \cdots A_k$ and the *whole* matrix of $A_{k+1} A_{k+2} \cdots A_j$. Finally we multiply the s-th row of $A_i A_{i+1} \cdots A_k$ to $A_{k+1} A_{k+2} \cdots A_j$.

Notice that we can compute s-th row of $A_i A_{i+1} \cdots A_k$ using the same scenario. Thus we have

$$\begin{aligned} \mathscr{R}_s(i,j) &= \min_{i \leqslant k < j} \{ \mathscr{R}_s(i,k) + \mathscr{C}(k+1,j) + \\ &\qquad \boldsymbol{C}_s\Big((A_i A_{i+1} \cdots A_k), (A_{k+1} A_{k+2} \cdots A_j) \Big) \Big\} \end{aligned} \tag{5.9}$$

where $\boldsymbol{C}_s\Big((A_i A_{i+1} \cdots A_k), (A_{k+1} A_{k+2} \cdots A_j) \Big)$ is the cost of multiplying the s-th row of $A_i A_{i+1} \cdots A_k$ to $A_{k+1} A_{k+2} \cdots A_j$. Similarly, we can solve the recursion of (5.9) using dynamic-programming in $O(N^3)$.

The dynamic-programming solution to the recursion (5.9) needs $O(N^3)$ finite field multiplication and/or addition. The complexity of the algorithm is more than the straightforward sequential multiplication of the matrices in the chain of (5.1). Similar to Section 5.4, by looking at the empirical solution of the recursion (5.9) we find a suboptimal solution that has better complexity but good performance.

The optimum solution of (5.9) is usually the following series of multiplications: The s-th row of A_1 multiplies to A_2 the result multiplies to A_3 and iteratively this goes up to A_N. Based on this observation, we propose the following suboptimal algorithm:

$\boxed{1}$ Find $\mathrm{Deg}(A_1 A_2 \cdots A_N)$ the degree-matrix of the chain of matrices. From Lemma 5.2, we know that it takes $O(N)$.

$\boxed{2}$ From $\mathrm{Deg}(A_1 A_2 \cdots A_N)$ pick the row of $A_1 A_2 \cdots A_N$ with smallest $(1, k-1)$-weighted degree. Denote that row by s.

$\boxed{3}$ Set $\boldsymbol{v}$ to be the s-th row of A_1 and $i := 2$.

$\boxed{4}$ While $i \leqslant N$ update $\boldsymbol{v} \leftarrow \boldsymbol{v} A_i$ and $i \leftarrow i+1$.

$\boxed{5}$ Return $\boldsymbol{v}$.

The performance of the suboptimal algorithm and optimal algorithm are plotted in Figure 5.1 and Figure 5.2. In these figures, “Sequential” stands for the straightforward multiplication of (5.1), “Suboptimal with different depths” are the solution of (5.8) with different d’s, the “Optimum” is the solution of (5.6), the “Least weighted-degree Optimum” is the solution of (5.9), and the “Least weighted-degree Suboptimum” is the complexity of suboptimal algorithm presented in Section 5.5.

The results of Chapter 5 are published, in part, in proceedings of *2004 International Symposium on Information Theory (ISIT)*, Parvaresh, Farzad; Vardy, Alexander. The dissertation author was the primary investigator and author of this paper.

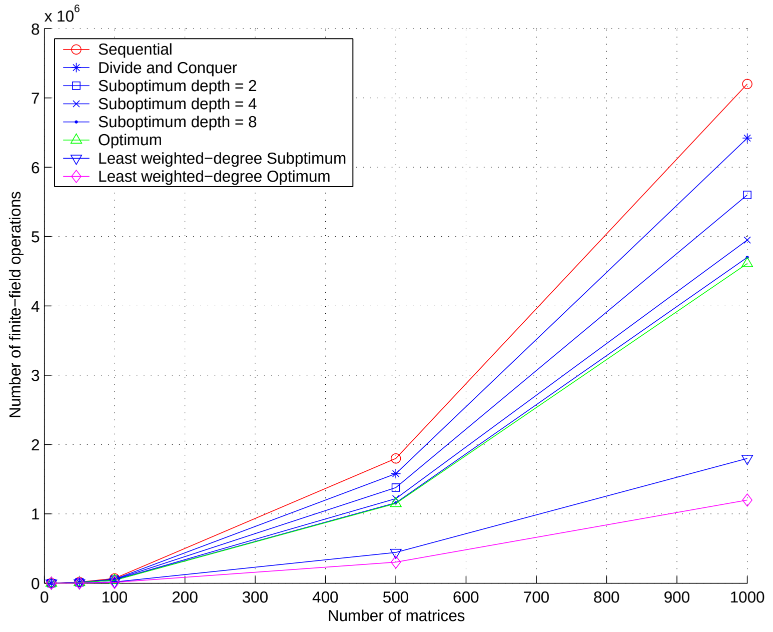


Figure 5.1: Average cost of multiplying a 4×4 polynomial matrix-chain.

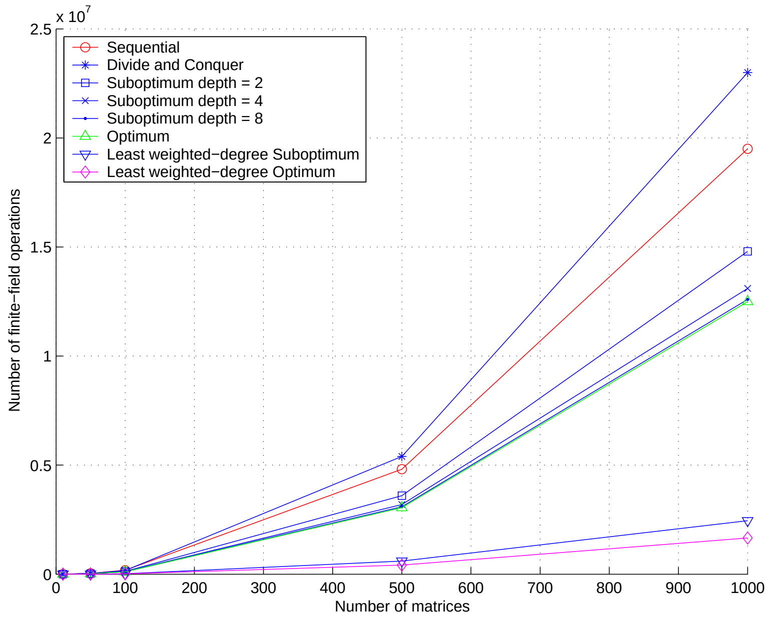


Figure 5.2: Average cost of multiplying an 8×8 polynomial matrix-chain.

CHAPTER 6

Conclusion and open problems

Before the pioneering work of Sudan [Sud97] and Guruswami-Sudan [GS99], complexity of most of the algorithm that decode beyond the half-the-distance bound for many family of error-correcting codes would grow exponentially. Sudan came up with an algebraic list decoding algorithm that can correct up to $1 - \sqrt{R}$ fraction of errors for Reed-Solomon codes (and many other algebraic codes). The Sudan-type list decoding algorithms basically consist of three parts: *multiplicity assignment, interpolation and factorization*. The emphasis of this work was to study, understand and improve these class of algebraic list-decoding algorithms.

We first looked at the multiplicity assignment part of the algorithm. Koetter and Vardy in [KV03a] present a novel way of assigning the multiplicities. The goal of the multiplicity assignment scheme of [KV03a] is to maximize the *probability* of correct decoding. They show that, for a given multiplicity assignment this probability is given by $\Pr\{\mathcal{S}_M > \Delta(M)\}$, where $\mathcal{S}_M$ is a random variable whose distribution depends on the channel output. Here, $\Delta(M)$ is a known deterministic function of M. The goal then is to compute M in such way that maximizes $\Pr\{\mathcal{S}_M > \Delta(M)\}$.

We are able to partially solve the optimization problem. We first relax the to be real numbers instead of integers. Then, we recast the problem into a geometrical framework. While we are yet unable to solve the corresponding geometric problem, we establish several key properties of the resulting geometric structure which later become handy in solving the optimization problem for finite multiplicities.

A different approach is taken by upper bounding the probability of failure using the Chebyshev bound. So, instead of minimizing the actual probability of failure we minimize the upper bound on the probability of failure. This leads to an iterative and analytical solutions for the multiplicity assignment when the multiplicities are large.

Finally, we use both the geometrical framework and the analytical solution for large multiplicities to derive the solution for finite multiplicities. We are able to get up to 0.20dB coding gain in decoding of Reed-Solomon codes of length 255 and up to 0.75dB coding gain for decoding of Reed-Solomon codes of length 15, as compared to the Koetter-Vardy algorithm over BAWGN cannel.

Many open problems remain to investigate. Can we solve the problem for integer multiparities? Currently, for a given reliability matrix and multiplicity matrix we can compute the distribution of $\mathcal{S}_M$ efficiently and so $\Pr\{\mathcal{S}_M > \Delta(M)\}$. Can this somehow be used to optimize the probability of failure for integer multiplicities? A better upper bound on the probability of failure such as Chernoff bound can be used instead of Chebyshev bound for the optimization. There have been some research in this direction presented at [EMH04, EM05]. However, the complexity of these algorithms are high comparing to Koetter-Vardy algorithm. Is it possible to simplify these algorithms for high SNR when the Chernoff-type optimizations usually show some coding gain over the Chebyshev-type multiplicity assignment. How far are the current multiplicity assignment techniques from the optimum? Is the gap between the performance of the optimum multiplicity assignment and the current approximations reasonably large for practical purposes and further research?

The other part the of the decoder is the interpolation part. The interpolation part is the bottleneck of the decoder for the implementation. Many research already has been done on reducing the complexity of the interpolation part which leaded to many clever ideas presented at [AKS03, FG01, KV03b, McE03b, NH98, RR00]. Most of these results are based on the iterative algorithm originally developed by Koetter [Köt96, Koe96], for computing a Gröbner basis of the ideal of polynomials that pass through the received points with the prescribed multiparities. The complexity of these algorithms are in general $O(n^2)$ where n is length of the code. Recently, Alekhnovich [Ale02] showed that the interpolation can be done in $O((\frac{w}{r})^{0(1)} n \log^2 n \log\log n)$ where, r is rate of the code and w is the maximal multiplicity assigned to a vertical line.

We recast the interpolation problem into a polynomial matrix-chain product. The parenthesization of the matrices changes complexity of computing the product. We show that using a dynamic-programming algorithm one can finds the best parenthesization. The complexity of the dynamic-programming algorithm is in order of $O(n^3)$. Next, we derive a suboptimal dynamic-programming algorithm, presented in Chapter 5, which has complexity of $O(n^2)$. With the suboptimal algorithm we can reduce the number of finite field operation for computing the Gröbner basis of the ideal by a factor of at least two.

Can we find algorithms for polynomial matrix-chain parenthesization with better complexity? For scaler matrix multiplication problem there are algorithms that solve the parenthesization problem in $O(n \log n)$. A study of these algorithms for polynomial matrix-chain product problem may lead to much faster algorithms.

The rest of the work is based on generalization of Sudan-type decoders to construct codes with better decoding radius. Our first attempt is to decode $M-1$ transmitted Reed-Solomon codewords together, using M-variate polynomial interpolation. It is shown that if the channel errors are *synchronized* – occur in the same positions in all the $M-1$ codewords – this algorithm can often correct up to $n\big(1-R^{(M-1)/M}\big)$ errors in a Reed-Solomon code of length n and rate R, which is significantly higher than the Guruswami-Sudan decoding radius of $n\big(1-R^{1/2}\big)$. When number of errors is smaller than Guruswami-Sudan bound the algorithm for sure recovers the codeword but when number of errors is between $n\big(1-R^{(M-1)/M}\big)$ and $n\big(1-R^{1/2}\big)$ then for some error patterns the new decoder fails. We are able to analyze the performance of the decoder for a special case of $M = 2$ where multiplicity of the interpolation is one. In this case, the MID algorithm attempts to correct up to $n\tau_{2,1}$ errors, where $\tau_{2,1} = 1 - \sqrt[3]{6R^2}$. We consider the situation where symbol values received from the channel at the erroneous positions are distributed uniformly at random (a version of the q-ary symmetric channel). We show that, with high probability, the performance of the MID algorithm is very close to $\tau_{2,1}$ in this case. Specifically, we prove that if the fraction of positions in error is at most $\tau_{2,1} - O(R^{5/3})$, then the probability of failure of the MID algorithm is at most $n^{-\Omega(n)}$. Thus the probability of failure is, indeed, negligible for large n in this case. Analyzing the performance of the algorithm for a general case of M-variate decoding with arbitrary multiplicity m remains open.

The interpolation part of the M-variate decoding algorithm has complexity of $O(n^2)$. However, we are expecting that by using similar techniques presented in [Ale02] we should be able to reduced the complexity further to $O(n \log n)$.

One of the main results of the dissertation is a consequence of deviation from multivariate interpolation algorithm of Chapter 2. We introduce a new family of error-correcting codes that have a polynomial-time encoder and a polynomial-time list-decoder, where these codes correct a fraction of adversarial errors up to $\tau_M = 1 - \sqrt[M+1]{M^M R^M}$. Here, R is the rate of the code and $M \geqslant 1$ is an arbitrary integer parameter. This bound is beyond the Guruswami-Sudan radius of $1 - \sqrt{R}$ for all rates less than $1/16$. Notably, this error-correction is achieved in the worst-case against adversarial errors: a probabilistic model for the error distribution is neither needed nor assumed. The best results at the time of publication of the paper required a rate of $O(\varepsilon^2)$ to achieve the correction radius of $1-\varepsilon$. Later, Guruswami and Rudra [GR06c] show that a specific class of the codes we define here combined by a *compression/decompression* technique can correct up to a fraction of $1 - \sqrt[M+1]{R^M}$ errors. The list-size of these codes grows polynomially with length of the code. Fixing the list-size of Guruswami-Rudra code remain open. Guruswami-Rudra code construction gets close to capacity of adversarial channel when alphabet size of the codes is not bounded. Is it possible to modify these class of codes to get codes with constant alphabet size? Can we concatenate these algebraic list-decodable codes with well-studied constant size binary codes, such as Hadamard codes, to achieve the capacity of adversarial binary channel?

References

[ABKR00] John Abbott, Anna M. Bigatti, Martin Kreuzer, and Lorenzo Robbiano, *Computing ideals of points*, Journal of Symbolic Computing **30** (2000), 341–356.

[AKS03] Arshad Ahmed, Ralf Koetter, and Naresh R. Shanbhag, *VLSI Architectures for soft-decision decoding of Reed-Solomon Codes*, IEEE Transactions on VLSI Systems (2003), submitted for publication.

[AKS04] Ahmed Arshad, Ralf Koetter, and Naresh R. Shanbhag, *VLSI architectures for soft-decision decoding of Reed-Solomon codes*, IEEE International Conference on Communications, vol. 5, June 2004, pp. 2584–2590.

[Ale02] Michael Alekhnovich, *Linear Diophantine Equations over Polynomials and Soft Decoding of Reed-Solomon Codes*, FOCS '02: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 2002, pp. 439–448.

[Ber68] Elwyn R. Berlekamp, *Algebraic Coding Theory*, McGrow-Hill, New York, 1968.

[Ber70] ______, *Factoring polynomials over large finite fields*, Mathematics of Computation **24** (1970), 713–735.

[BKR05] Eli Ben-Sasson, Swastik Kopparty, and Jaikumar Radhkrishnan, *The list-decoding radius for Reed-Solomon codes*, AMS Sectional Meeting (Lincoln, NE), October 2005.

[BKY03] Daneil Bleichenbacher, Aggelos Kiayias, and Moti Yung, *Decoding of interleaved Reed-Solomon codes over noisy data*, Lecture Notes in Computer Science **2719** (2003), 97–108.

[BM82] Bruno Buchberger and Hans Michael Möller, *The construction of multivariate polynomials with preasigned zeros*, Lecture Notes in Computer Science **144** (1982), 24–31.

[BMS05] Andrew Brown, Lorenz Minder, and Amin Shokrollahi, *Improved decoding of interleaved AG codes*, Cryptography and Coding: 10th IMA International Conference, Cirencester (Nigel P. Smart, ed.), Lecture Notes in Computer Science, vol. 3796, November 2005, pp. 37–46.

[CLO96] David A. Cox, John B. Little, and Don O'Shea, *Ideals, Varieties, and Algorithms*, Springer-Verlag, Berlin, 1996.

[CS03] Don Coppersmith and Madhu Sudan, *Reconstructing curves in three (and higher) dimensional space from noisy data*, STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 2003, pp. 136–142.

[CT84] Hu T. C. and Shing M. T., *Computation of matrix chain products, Part I and Part II*, SIAM Journal of Computing **11, 13** (1982, 1984), 362–373, 228–251.

[Eli57] Peter Elias, *List decoding for noisy channels*, Technical Report 355, Research Laboratory of Electronics, MIT (1957).

[EM05] Mostafa El-Khamy and Robert J. McEliece, *Interpolation Multiplicity Assignment Algorithms for Algebraic Soft-Decision Decoding of Reed Solomon Codes*, AMS-DIMACS volume on Algebraic Coding Theory and Information Theory, vol. 68, 2005.

[EMH04] Mostafa El-Khamy, Robert J. McEliece, and Jonathan Harel, *Performance Enhancements for Algebraic Soft-Decision Decoding of Reed Solomon Codes*, IEEE International Symposium on Information Theory (Chicago, IL), June-July 2004.

[FG01] Gui-Liang Feng and Xavier Giraud, *Fast algorithms in Sudan decoding procedure for Reed-Solomon codes*, 2001.

[FO02] Patrick Fitzpatrick and Henry O'Keeffe, *Gröbner basis solutions of constrained interpolation problems*, Linear Algebra and Applications, vol. 351/352, 2002, pp. 533–551.

[Gan89] David G. Gantor, *On arithmetical algorithms over finite fields*, Journal of Combinatorial Theory Series A **50** (1989), no. 2, 285–300.

[GI01] Venkatesan Guruswami and Piotr Indyk, *Expander-based constructions of efficiently decodable codes*, IEEE Symposium on Foundations of Computer Science, October 2001, pp. 658–667.

[GI02] ______, *Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets*, STOC '02: Proceedings of the thirty-fourth annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 2002, pp. 812–821.

[GKKG02] Warren J. Gross, Frank R. Kschischang, Ralf Koetter, and P. Glenn Gulak, *A VLSI architecture for interpolation in soft-decision list decoding of Reed-Solomon codes*, Proceedings of IEEE Workshop on Signal Processing Systems (San Diego, CA), October 2002, pp. 39–44.

[GN97] Joachim von zur Gathen and Michael Nöcker, *Exponentiation in finite fields: Theory and practice*, Lecture Notes in Computer Science **1255** (1997), 88–113.

[God73] S. S. Godbole, *On efficient computation of matrix chain products*, IEEE Transactions on Computers **C–22** (1973), 864–866.

[Gol49] Marcel J. E. Golay, *Notes on digital coding*, Proceedings of IRE, vol. 37, 1949, p. 657.

[GP06] Venkatesan Guruswami and Anindya C. Patthak, *Correlated Algebraic-Geometry codes : Improved List-decoding over Bounded Alphabets*, FOCS 2006: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, vol. 46th, 2006, to appear.

[GR05] Venkatesan Guruswami and Atri Rudra, *Limits to list decoding Reed-Solomon codes*, STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 2005, pp. 602–609.

[GR06a] Philippe Gaborit and Olivier Ruatta, *Improved Hermite multivariable polynomial interpolation*, IEEE International Symposium on Information Theory, 2006.

[GR06b] Venkatesan Guruswami and Atri Rudra, *Achieving List Decoding Capacity Using Folded Reed-Solomon Codes*, Proceedings of 44-th Annual Allerton Conference on Communications, Control and Computing, September 2006.

[GR06c] ______, *Explicit capacity-achieving list-decodable codes*, STOC '06: Proceedings of the thirty-eighth annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 2006, pp. 1–10.

[GS99] Venkatesan Guruswami and Madhu Sudan, *Improved decoding of Reed-Solomon and algebraic-geometric codes*, IEEE Transactions on Information Theory **45** (1999), 1757–1767.

[Gur01] Venkatesan Guruswami, *List decoding of error-correcting codes*, Ph.D. thesis, MIT, 2001.

[Gur04] ______, *Better extractors for better codes?*, Proceedings of 36-th ACM Symposium on Theory of Computing (STOC) (2004), 436—444.

[Gur05a] ______, *Algebraic-geometric generalizations of parvaresh-vardy codes*, July 2005.

[Gur05b] ______, *List decoding of error-correcting codes*, Lecture Notes in Computer Science, vol. 3282, ch. 5, Springer, 2005.

[GV05] Venkatesan Guruswami and Alexander Vardy, *Maximum-likelihood decoding of Reed-Solomon codes is NP-hard*, IEEE Transactions on Information Theory **51** (2005), no. 7, 2249–2256.

[Ham50] Richard W. Hamming, *Error detecting and error correcting codes*, Bell System technical Journal **29** (1950), 147–160.

[Har28] Ralph Vinton Lyon Hartley, *Transmission of information*, Bell Technical Journal (1928), 535.

[Has49] Helmut Hasse, *Number theory*, Springer, Berlin, 1949.

[Imm91] Kees A. SSchouhamer Immink, *Coding Techniques for Digital Recorders*, Prentice–Hall, 1991.

[JH01] Jørn Justesen and Tom Høholdt, *Bounds on list decoding of MDS codes*, IEEE Transaction of Information Theory **47** (2001), 1604–1609.

[JTH04] Jørn Justesen, Chritian Thommesen, and Tom Høholdt, *Decoding of concatenated codes with interleaved outer codes*, Proceedings of IEEE Symposium on Information Theory (ISIT) (Chicago, IL), July 2004.

[KMVA03] Ralf Koetter, Jun Ma, Alexander Vardy, and Ahmed Arshad, *Efficient interpolation and factorization in algebraic soft-decision decoding of Reed-Solomon codes*, Proceedings of IEEE Symposium on Information Theory (Yokohama, Japan), 2003, p. 365.

[Koe96] Ralf Koetter, *Fast generalized minimum distance decoding of algebraic geometric and Reed-Solomon codes*, IEEE Transactions on Information Theory **42** (1996), 721–738.

[Köt96] Ralf Kötter, *On Algebraic Decoding of Algebraic-Geometric and Cyclic codes*, Ph.D. thesis, University of Linköping, Sweden, 1996.

[KV02] Ralf Koetter and Alexander Vardy, *Decoding of Reed-Solomon codes for additive cost functions*, IEEE Symposium on Information Theory (ISIT) (Lausanne, Switzerland), July 2002.

[KV03a] ______, *Algebraic soft-decision decoding of Reed-Solomon codes*, IEEE Transactions on Information Theory **49** (2003), no. 11, 2809–2825.

[KV03b] ______, *A complexity reducing transformation in algebraic list decoding of Reed-Solomon codes*, Proceedings of IEEE Information Theory Workshop (Paris, France), April 2003, pp. 10–13.

[KY02] Aggelos Kiayias and Moti Yung, *Cryptographic hardness based on the decoding of Reed-Solomon codes with applications*, Tech. Report TR02-017, Electronic Colloquium Computational Complexity ECCC, March 2002.

[LKHL03] Heejo Lee, Jong Kim, Sung Je Hong, and Sunggu Lee, *Processor allocation and task scheduling of matrix chain products on parallel systems*, IEEE Transactions on Parallel and Distributed Systems **14** (2003), no. 4, 394–407.

[LM88] Edward A. Lee and David G. Messerschmitt, *Digital communication*, Kluwer, Boston, 1988.

[LN86] Rudolf Lidl and Harald Niederreiter, *Introduction to Finite Fields and their applications*, Cambridge University Press, Cambridge, MA,, 1986.

[Mas69] James L. Massey, *Shift-register synthesis and BCH decoding*, IEEE Transaction on Information Theory **15** (1969), 122–127.

[McE03a] Robert J. McEliece, *The Guruswami-Sudan decoding algorithm for Reed-Solomon codes*, Interplanetary Network Progress Report (NASA), May 2003, pp. 42–135.

[McE03b] ______, *On the average list size for the Guruswami-Sudan decoder*, 7th International Symposium on Communication Theory and Applications, July 2003, pp. 2–6.

[MMM93] M. G. Marinari, Hans Michael Möller, and T. Mora, *Gröbner bases of ideals defined by functionals with and application to ideals of projective points*, Applied Algebra in Engineering Communication Computer **4** (1993), 103–145.

[MS81] F. Jessie MacWilliams and Neil J. A. Sloane, *The theory of error-correcting codes*, Elsevier/North-Holland, Amsterdam, 1981.

[MS94] Robert J. McEliece and Laif Swanson, *Reed-Solomon Codes and their Applications*, ch. Reed-Solomon codes and the exploration of the solar system, pp. 25–40, IEEE Press, New York, 1994.

[MS05] V.N. Muralindhara and S. Sen, *On the tightness of the Johnson bound for Reed-Solomon codes*, 16-th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2005, submitted for publication.

[MTV04] Jun Ma, Peter Trifonov, and Alexander Vardy, *Divide-and-conquer interpolation for list decoding of Reed-Solomon codes*, Proceedings of IEEE Symposium on Information Theory (ISIT) (2004).

[NH98] R. Refslund Nielsen and Tom Høholdt, *Decoding Reed-Solomon codes beyond half the minimum distance*, Proceedings of International Conference on Coding Theory, Cryptography, and Related Areas (Gaunajuato, Mexico), 1998, pp. 221–236.

[Nie03] R. Refslund Nielsen, *Decoding concatenated codes with Sudan's algorithm*, submitted for publication, 2003.

[Nyq24] Harry Nyquist, *Certain factors affecting telegraph speed*, Bell System Technical Journal (1924), 324.

[Nyq28] ______, *Certain topics in telegraph transmission theory*, A.I.E.E. Transaction **47** (1928), 617.

[OS99] Vadim Olshevsky and M. Amin Shokrollahi, *A displacement approach to efficient decoding of algebraic-geometric codes*, STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing (New York, NY, USA), ACM Press, 1999, pp. 235–244.

[Pan92] Victor Y. Pan, *Complexity of computations with matrices and polynomials*, SIAM Review **34** (1992), 225–262.

[Pec01] Lancelot Pecquet, *List decoding of algebraic-geometric codes*, IEEE Transaction on Information Theory (2001), submitted for publication.

[Pec02] ______, *Décodage en liste des codes géométriques*, Ph.D. thesis, Universit Pierre et Marie Curie, Paris, France, 2002.

[Pet60] W. Wesley Peterson, *Encoding and error-correcting procedures for Bose-Chaudhuri codes*, IEEE Transactions on Information Theory **6** (1960), 459–470.

[PV03] Farzad Parvaresh and Alexander Vardy, *Multiplicity assignments for algebraic soft-decision decoding of Reed-Solomon codes*, Proceedings of IEEE Symposium on Information Theory (ISIT) (Yokohama, Japan), July 2003.

[PV04a] ______, *Multivariate interpolation decoding beyond the Guruswami-Sudan radius*, Proceedings of 42nd Annual Allerton Conference on Communications, Control and Computing, October 2004.

[PV04b] ______, *Polynomial matrix-chain interpolation in Sudan-type Reed-Solomon decoders*, Proceedings of IEEE Symposium on Information Theory (ISIT) (Chicago, IL), July 2004.

[PV05] ______, *Correcting Errors Beyond the Guruswami-Sudan Radius in Polynomial Time*, FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 2005, pp. 285–294.

[PV06] ______, *Multivariate interpolation decoding of Reed-Solomon codes*, May 2006.

[RK05] Niranjan Ratnakar and Ralf Koetter, *Exponential Error Bounds for Algebraic Soft-Decision Decoding of Reed-Solomon Codes*, IEEE Transactions on Information Theory **51** (2005), 3899–3917.

[RR00] Ron M. Roth and Gitit Ruckenstein, *Efficient decoding of Reed-Solomon codes beyond half the minimum distance*, IEEE Transaction on Information Theory **46** (2000), no. 1, 246–257.

[RR03] Gitit Ruckenstein and Ronny M. Roth, *Bounds on the list-decoding radius of reed-solomon codes*, SIAM Journal on Discrete Mathematics **17** (2003), 171–195.

[RS60] Irving S. Reed and Gustave Solomon, *Polynomial codes over certain finite fields*, SIAM: Journal of the Society for Industrial and Applied Mathematics **8** (1960), no. 2, 300–304.

[RU05] Thomas J. Richardson and Rüdiger L. Urbanke, *Modern Coding Theory*, http://lthcwww.epfl.ch/mct/, August 2005.

[Sha48] Claude Elwood Shannon, *A mathematical theory of communication*, Bell System Technical Journal **423** (1948), 623–656.

[Sho91] Victor Shoup, *A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic*, Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC) (Bonn,Germany), July 1991, pp. 14–21.

[Sud97] Madhu Sudan, *Decoding of Reed-Solomon codes beyond the error-correction bound*, Journal of Complexity **13** (1997), no. 1, 180–193.

[WB86] Lloyd R. Welch and Elwyn R. Berlekamp, *Error correction for algebraic block codes*, US Patent No. 4,633,470, 1986.

[Woz58] John M. Wozencraft, *List decoding*, Quarterly Progress Report, Research Laboratory of Electronics, MIT **48** (1958), 90–95.

[WS01] Xin-Wen Wu and Paul H. Siegel, *Efficient Root-Finding Algorithm With Application to List Decoding of Algebraic-Geometric Codes*, IEEE Transactions on Information Theory **47** (2001), 2579–2587.